UNIVERSITY OF CALIFORNIA SAN DIEGO

Understanding Expressivity and Trustworthy Aspects of Deep Generative Models

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Zhifeng Kong

2023

University of California San Diego

2023

EPIGRAPH

Believe in yourself and all that you are.
Know that there is something inside you that is greater than any obstacle.

Christian D. Larson

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Kamalika Chaudhuri. I have learned many valuable things from her during these five years of Ph.D. study and research. She not only taught me how to solve hard, technical problems, but also guided me how to find and define good research problems, instructed me how to make good hypothesis and design experiments, and taught me good writing and presentation skills. Besides these, she also gave me numerous valuable career and life advices. She has always been supportive in all aspects, and without her help, I would not have accomplished the research outcomes that I made.

I would like to thank the rest of my committee members, Professor Sanjoy Dasgupta, Professor Berk Ustun, and Professor Rose Yu, for their helpful feedback that helped me complete my dissertation.

I would like to thank my co-author, Professor Scott Alfeld. Without your help, our paper would not have been successful and appeared in my dissertation. I would like to thank all my other co-authors, a partial list including Professor Dahua Lin, Professor Ngai Wong, Professor Luca Daniel, Dr. Amrita Roy Chowdhury, Dr. Wei Ping, Dr. Bryan Catanzaro, Dr. Zhaoyang Lyu, Dr. Jiaji Huang, Ching-Yun Ko, Xudong Xu, Liang Pan, Kexin Zhao, Ambrish Dantrey. Although our papers did not make it to this dissertation, many of them would not have been successful without your help.

I would like to thank my parents, Yuan Wang and Weiguang Kong, for their unconditional support in my life.

I would like to thank my friends who accompanied me during my years of study.

I would like to express my special thanks to Dr. Catherine Mackinnon Robertson, Dr. Michael Warren Gibbs, Dr. Kenneth Todd Rutkowski, Dr. Elliott Caponetti, and Dr. Laicee Kimberly Grahek. I had a serious sport injury in my last year of my Ph.D. Without your operation and physical therapy I would not be able to stand and walk again.

Chapter 1, in full, is a reprint of the paper as it appears in *International Conference on Artificial Intelligence and Statistics, 2020,* by Zhifeng Kong and Kamalika Chaudhuri [Kong and Chaudhuri, 2020]. The dissertation author was the primary researcher and author of this paper.

Chapter 2, in full, is a reprint of the paper as it appears in *International Conference on Machine Learning INNF+ Workshop, 2021,* by Zhifeng Kong and Kamalika Chaudhuri [Kong and Chaudhuri, 2021b]. The dissertation author was the primary researcher and author of this paper.

Chapter 3, in full, is a reprint of the paper as it appears in *Advances in Neural Information Processing Systems, 2021,* by Zhifeng Kong and Kamalika Chaudhuri [Kong and Chaudhuri, 2021a]. The dissertation author was the primary researcher and author of this paper.

Chapter 4, in full, is a reprint of the paper as it appears in *IEEE Conference on Secure and Trustworthy Machine Learning, 2023,* by Zhifeng Kong and Kamalika Chaudhuri [Kong and Chaudhuri, 2023b]. The dissertation author was the primary researcher and author of this paper.

Chapter 5, in full, is currently being prepared for submission for publication of the material, by Zhifeng Kong and Kamalika Chaudhuri [Kong and Chaudhuri, 2023a]. The dissertation author was the primary researcher and author of this material.

Chapter 6, is based on the manuscript as it appears in *arXiv preprint arXiv:2206.14439, 2023,* by Zhifeng Kong and Scott Alfeld [Kong and Alfeld, 2022]. The dissertation author was the primary researcher and author of this paper.

VITA

2014–2018	B.S. in Mathematics and Applied Mathematics, Xi'an Jiaotong University, China
2018–2023	Ph.D. in Computer Science, University of California San Diego

PUBLICATIONS

Zhifeng Kong and Kamalika Chaudhuri. Data Redaction from Pre-trained GANs, IEEE SaTML, 2023.

Zhifeng Kong and Scott Alfeld. Approximate Data Deletion in Generative Models, arXiv preprint arXiv:2206.14439, 2023.

Zhifeng Kong*, Amrita Roy Chowdhury*, and Kamalika Chaudhuri. Can Membership Inferencing be Refuted? ArXiv preprint arXiv:2303.03648, 2023.

Zhifeng Kong, Wei Ping, Ambrish Dantrey, and Bryan Catanzaro. Speech Denoising in the Waveform Domain with Self-Attention, ICASSP, 2022.

Zhaoyang Lyu*, **Zhifeng Kong***, Xudong Xu, Liang Pan, and Dahua Lin. A conditional point diffusion-refinement paradigm for 3d point cloud completion, ICLR, 2022.

Zhifeng Kong and Kamalika Chaudhuri. Understanding Instance-based Interpretability of Variational Auto-Encoders, NeurIPS, 2021.

Zhifeng Kong and Kamalika Chaudhuri. Universal Approximation of Residual Flows in Maximum Mean Discrepancy, ICML INNF+ Workshop, 2021.

Zhifeng Kong and Wei Ping. On Fast Sampling of Diffusion Probabilistic Models, ICML INNF+ Workshop, 2021.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A Versatile Diffusion Model for Audio Synthesis, ICLR, 2021.

Zhifeng Kong and Kamalika Chaudhuri. The Expressive Power of a Class of Normalizing Flow Models, AISTATS, 2020.

Zhaoyang Lyu, Ching-Yun Ko, **Zhifeng Kong**, Ngai Wong, Dahua Lin, and Luca Daniel. Fastened CROWN: Tightened Neural Network Robustness Certificates, AAAI, 2020.

ABSTRACT OF THE DISSERTATION

Understanding Expressivity and Trustworthy Aspects of Deep Generative Models

by

Zhifeng Kong

Doctor of Philosophy in Computer Science

University of California San Diego, 2023

Professor Kamalika Chaudhuri, Chair

Deep Generative Models are a kind of unsupervised deep learning methods that learn the data distribution from samples and then generate unseen, high-quality samples from the learned distributions. These models have achieved tremendous success in different domains and tasks. However, many questions are not well-understood for these models. In order to better understand these models, in this dissertation, we investigate the following questions: (*i*) what is the representation power of deep generative models, and (*ii*) how to identify and mitigate trustworthy concerns in deep generative models.

We study the representation power of deep generative models by looking at which distributions they can approximate arbitrarily well. we study normalizing flows and rigorously establish bounds on their expressive power. Our results indicate that some basic flows are highly expressive in one dimension, but in higher dimensions their representation power may be limited, especially when the flows have moderate depth. We then prove residual flows are universal approximators in maximum mean discrepancy and provide upper bounds on the depths under different assumptions.

We next investigate three different trustworthy concerns. The first is how to explain the black box neural networks in these models. We introduce VAE-TracIn, a computationally efficient and theoretically sound interpretability solution, for VAEs. We evaluate VAE-TracIn on real world datasets with extensive quantitative and qualitative analysis.

The second is how to mitigate privacy issues in learned generative models. We propose a density-ratio-based framework for efficient approximate data deletion in generative models, which avoids expensive re-training. We provide theoretical guarantees under various learner assumptions and empirically demonstrate our methods across a variety of generative methods.

The third is how to prevent undesirable outputs from deep generative models. We take a compute-friendly approach and investigate how to post-edit a pre-trained model to *redact* certain samples. We consider several unconditional and conditional generative models and various types of descriptions of redacted samples. Extensive evaluations on real-world datasets show our algorithms outperform baseline methods in redaction quality as well as robustness while retaining high generation quality.

Introduction

Deep generative models are a type of unsupervised learning methods that aim to learn the data distribution from samples and then generate unseen, high-quality samples from the learned distributions. There are many different ways to design and train these models, which can be roughly categorized as generative adversarial networks (GANs), variational auto-encoders (VAEs), normalizing flows (NFs), diffusion models, energy-based models, and autoregressive models. These models have shown tremendous success in a wide range of tasks such as image, audio, text, molecular data, and many domains as well as cross-domain tasks.

Despite the great success of these models, many questions for these models remain not well-understood. In this dissertation, we investigate the following two questions in order to better understand deep generative models. The first question is what is the representation power of these models. Formally, are these models able to express or approximate arbitrary distributions? The second question is how to identify and mitigate trustworthy concerns of these models? This includes, how to interpret the black box neural networks, how to make these models satisfy privacy regulations, and how to prevent these models from producing undesirable samples (such as offensive or biased contents). It is of great significance to understand these problems because these can ensure expressivity and reliability of deep generative models at deployment, as the motivation of the present dissertation.

0.1 Deep Generative Models

Deep Generative Models are trained to learn probability distributions and produce new samples. There are two fundamental ways that a deep generative model produces a sample. The first is **unconditional** generation. An unconditional model is usually trained on a set of samples (without labels or context) and generates different samples randomly. The input is often a latent code from a latent distribution such as a Gaussian, which leads to the randomness of generation. The second is **conditional** generation. A condition model is trained on a set of sample-context pairs, such as image-caption pairs in text-to-image models. At the generation phase, the model takes two inputs, the context and the latent code, and outputs a sample based on the context. In these models, the context controls the content and style of the outputs, whereas the latent code still controls the randomness.

There are many different types of deep generative models based on how they are trained. Generative Adversarial Networks [Goodfellow et al., 2014] jointly train a discriminator and a generator with a minimax game, which trains the discriminator to distinguish real and generated samples, and trains the generator to fool the discriminator. Variation Auto-Encoders [Kingma and Welling, 2013] jointly train an encoder and a decoder with variational inference, where the encoder is trained to encode samples into codes and the decoder is trained to decode codes to samples. Normalizing Flows [Rezende and Mohamed, 2015a] train a series of invertible transformations between a normal distribution and the data distribution with maximum likelihood. Denoising Diffusion Probabilistic Models [Ho et al., 2020] (often abbreviated as Diffusion models) define a forward process that gradually adds noise to data, and then train a series of reverse blocks that gradually remove the noise from a high-dimensional Gaussian with score matching. There are many other generative models including energy-based models, auto-regressive models, and extension as well as combinations of these techniques.

0.2 Expressivity of Deep Generative Models

The theoretical analysis of expressivity (approximation capability) of neural networks can be traced back to Cybenko [1989] and Hornik et al. [1989], who proved the universal approximation theorem that neural networks with certain activations can approximate continuous functions arbitrarily well. In generative models, the neural networks transform a latent distribution such as a Gaussian to a generative distribution. The expressivity is usually defined as how well the generative distributions can approximate (or converge to) well-defined probability distributions in some probability metric. This is a very different problem compared to the expressivity question in the function space, and often needs very different mathematical tools to analyze the theoretical properties. There are also many different metrics to study, for example, stronger metrics such as L_1 distance, or weaker metrics such as weak convergence, and we may obtain very different results for different metrics even if we look at the same class of neural networks. Some neural networks are restricted by their architecture – for example, normalizing flows are invertible, and some other models change the data dimension – which make the expressivity problem non-trivial even if the neural networks themselves are universal approximators.

In Chapter 1, we look at the expressivity of a class of basic normalizing flows, with a focus on the L_1 distance metric. We show that planar flows [Rezende and Mohamed, 2015a] can be universal approximators on one dimension. We then show several basic flow models may have limited expressivity in higher dimensions, especially when the model has moderate depth.

In Chapter 2, we look at the expressivity of a more general class of normalizing flows called residual flows [Chen et al., 2019], with a focus on the maximum mean discrepancy (MMD) metric [Gretton et al., 2012]. We show that even though the residual flow blocks are restricted by the invertability and Lipschitz conditions, they are universal approximators in certain MMD metrics.

0.3 Trustworthiness of Deep Generative Models

There are many trustworthiness questions that are not well studied for deep generative models. In this dissertation, we focus on three different kinds of problems: interpretability, privacy issues, and undesirable outputs of these models.

Interpretability. Modern deep neural networks are often considered as black boxes as the computation inside these networks are often too complicated to track. Therefore, many methods look at how to interpret the computation of the neural networks. One class of methods is called instance-based interpretability, which studies the influence of training samples on the prediction on test samples [Koh and Liang, 2017, Yeh et al., 2018, Pruthi et al., 2020]. These methods aim to approximate the score which measures the difference of test loss if a particular training sample is removed from the training set.

However, in the literature of unsupervised learning especially generative models, instance-based interpretations are much less understood. In Chapter 3, we study instance-based interpretability of variational auto-encoders (VAEs) [Kingma and Welling, 2013]. We formally frame the counter-factual question answered by influence functions in this setting. We then introduce VAE-TracIn, a computationally efficient and theoretically sound solution, for VAEs. Finally, we evaluate VAE-TracIn on several real world datasets with extensive quantitative and qualitative analysis.

Undesirable outputs. In certain situations, deep generative models produce undesirable outputs. For example, with text-to-image models, one may craft a prompt that contains offensive, biased, malignant, or fabricated content, and generate a high-resolution image that visualizes the prompt [Nichol et al., 2021, Birhane et al., 2021, Schuhmann et al., 2022, Ramesh et al., 2022, Rando et al., 2022, Bedapudi, 2022, Laborde, 2022]. With speech synthesis models, one may easily turn text into celebrity voices [Betker, 2022, Wang et al., 2023, Zhang et al., 2023]. Text generation models can emit offensive, biased, or toxic content [Pitsilis et al., 2018, Wallace et al., 2019, McGuffie and Newhouse, 2020, Gehman et al., 2020, Abid et al., 2021, Perez et al., 2022, Schramowski et al., 2022b].

One plausible solution to mitigate this problem is to remove all undesirable samples from the training set and re-train the model. This is too computationally heavy for modern, large models. Another solution is to apply a classifier that filters out undesirable conditionals or outputs [Rando et al., 2022, Bedapudi, 2022, Laborde, 2022], or to edit the outputs and remove the undesirable content after generation [Schramowski et al., 2022a]. However, in cases where the model owners share the model weights with third parties, they do not have control over whether the filters or editing methods will be used. In order to prevent undesirable outputs more efficiently and reliably, we propose to post-edit the weights of a pre-trained model, which we call data redaction. We show that redaction is a fundamentally different task from data deletion, and data deletion may not always lead to redaction.

In Chapter 4, we frame the data redaction framework for unconditional generative models especially GANs. We provide three different algorithms for data redaction that differ on how the samples to be redacted are described. Extensive evaluations on real-world image datasets show that our algorithms out-perform data deletion baselines, and are capable of redacting data while retaining high generation quality at a fraction of the cost of full retraining.

In Chapter 5, we frame the data redaction framework for a broad class of conditional generative models. Our goal is to redact certain conditionals that will, with high probability, lead to undesirable content. This is done by distilling the conditioning network in the models, giving a solution that is effective, efficient, controllable, and universal for a class of deep generative models. We conduct experiments on redacting prompts in text-to-image models and redacting voices in text-to-speech models. Our method is computationally light, leads to better redaction quality and robustness than baseline methods while still retaining high generation quality.

Privacy issues. In recent years there are growing concerns in academia, government, and the private sector about user privacy and responsible data management. Several recent regulations (e.g., GDPR and CCPA) have introduced a right to erasure whereby a user may request that their data is deleted from a database. While it is straightforward to delete user data from a simple database, a savvy attacker might still be able to reverse-engineer the data by examining a machine learning model trained on it [Balle et al., 2021]. Re-training a model from scratch (after deleting the requested data) is computationally expensive, especially for modern large deep learning methods. This has motivated machine unlearning [Cao and Yang, 2015] where learned models are altered in a computationally cheap way to emulate the re-training process.

Prior work in supervised learning proposed approximate data deletion to approximate the re-trained model without actually performing the re-training [Guo et al., 2019, Neel et al., 2021, Sekhari et al., 2021, Izzo et al., 2021]. While these methods have achieved great success, approximate data deletion for unsupervised learning largely remains an open question. In Chapter 6, we propose a density-ratio-based framework for generative models. Using this framework, we introduce a fast method for approximate data deletion and a statistical test for estimating whether or not training points have been deleted. We provide theoretical guarantees under various learner assumptions and empirically demonstrate our methods across a variety of generative methods.

0.4 Overview

This dissertation is organized as follows.

In Chapter 1, we investigate the representation power of a basic kind of normalizing flows and present theoretical results on approximating arbitrary distributions in ℓ_1 distance. This joint work with Kamalika Chaudhuri has been published in AISTATS 2020 [Kong and Chaudhuri, 2020].

In Chapter 2, we investigate the representation power of residual flows and present theoretical results on approximating arbitrary distributions in maximum mean discrepancy. This joint work with Kamalika Chaudhuri has been published in ICML INNF+ Workshop 2021 [Kong and Chaudhuri, 2021b].

In Chapter 3, we study instance-based interpretability of VAEs. This joint work with Kamalika Chaudhuri has been published in NeurIPS 2021 [Kong and Chaudhuri, 2021a].

In Chapter 4, we study how to prevent undesirable outputs from learned unconditional generative models. This joint work with Kamalika Chaudhuri has been published in IEEE SaTML 2023 [Kong and Chaudhuri, 2023b].

In Chapter 5, we study how to prevent undesirable outputs from learned conditional generative models. This joint work with Kamalika Chaudhuri is currently being prepared for submission for publication [Kong and Chaudhuri, 2023a].

In Chapter 6, we look at privacy issues of generative models by studying a framework that is able to identify and mitigate these issues. This joint work with Scott Alfeld is currently being prepared for submission for publication [Kong and Alfeld, 2022].

Finally, we conclude the dissertation in Chapter 7.

Chapter 1

The Expressive Power of a Class of Normalizing Flow Models

1.1 Introduction

Normalizing flows are a class of deep generative models that aspire to learn an invertible transformation to convert a pre-specified distribution, such as a Gaussian, to the distribution of the input data. These models offer flexible generative modeling – as the invertible transformation can be implemented by deep neural networks – and easy likelihood computation in equation (1.3) that follows from the invertibility of the transformation [Rezende and Mohamed, 2015b].

Due to these advantages and their empirical success, a number of flow models have been proposed [Dinh et al., 2014, Germain et al., 2015, Uria et al., 2016, Kingma et al., 2016, Tomczak and Welling, 2016, Dinh et al., 2016, Papamakarios et al., 2017, Huang et al., 2018, Berg et al., 2018, Grathwohl et al., 2018, Behrmann et al., 2018, Jaini et al., 2019, Ho et al., 2019]. However, the expressive power offered by different kinds of flow models – what kind of distributions they can map between, and with what complexity – remains not well-understood, which makes it challenging to select the right flow model for specific tasks. Obviously, due to their invertible nature, a normalizing flow can only transform a distribution to one with a homeomorphic support [Armstrong, 2013]. However, even within such distributions, it remains unclear whether a simple distribution supported on $\mathbb{R}^d$ could be transformed or approximated via a normalizing flow from a Gaussian.

In this work, we carry out a rigorous analysis of the expressive power of planar flows, Sylvester flows, and Householder flows – the most basic classes of normalizing flows. The main challenge in analyzing the expressive power of any flow model class is *invertibility*. There is a body of prior work that analyzes the universal approximation properties of standard neural networks; however, analyzing the approximation properties of *invertible* mappings between distributions is a completely different problem. Just because a function class $\mathscr{F}$ is a universal approximator does not mean that the set of all its invertible functions can transform between arbitrary distributions; dually, even if functions in $\mathscr{F}$ have limited expressivity, it is possible that its invertible subset is an universal approximator in transforming between distributions [Villani, 2008]. Additionally, universal approximation properties are often proved by construction via non-invertible functions [Lu et al., 2017, Lin and Jegelka, 2018] and hence these constructions cannot to be used to establish properties of the corresponding flows.

This work gets around this challenge by studying properties of input-output distribution pairs directly, instead of considering the transformation class itself. In particular, we consider both a local and global analysis of properties of planar flows, their higher dimensional generalization – Sylvester flows, and Householder flows. First, we analyze the local topology – namely, the directional derivatives of the induced density. Second, we seek to bound the global total variation distance between the input and output distributions that can be achieved by each planar flow or Householder flow under certain conditions.

Using these two kinds of analysis, we make three main contributions in this chapter.

First, we show that in one dimension, even planar flows are highly expressive. In particular, they can transform a source distribution supported on $\mathbb{R}$ to an arbitrarily-accurate approximation of any target distribution supported on a finite union of intervals. The conclusion holds even if we restrict to planar flows with ReLU non-linearity and Gaussian source distributions. This indicates that planar flows in one dimension are universal approximators.

We next turn our attention to general d-dimensional spaces, and we look at what kinds of distributions may be expressed by a Sylvester flow model acting on a Gaussian, mixtures of Gaussian (MoG) distributions, or product (Prod) distributions. We show that when the non-linearity is a ReLU function, Sylvester flows of any depth cannot in general exactly transform between certain standard classes of distributions. In particular, ReLU Sylvester flows cannot exactly transform any mixture of k Gaussian distributions or product distributions into another one – no matter what the depth is – except under very special circumstances.

Finally, we consider the approximation capability of normalizing flow models in d-dimensional space. Here, we focus on local planar flows with a class of local non-linearities – including common non-linearities such as tanh, arctan and sigmoid – and Householder flows. We show that in these cases, provided certain conditions hold, transforming a source distribution into a target may require flows of inordinately large depth. In particular, if the target distribution $p(z)$ is constant in a ball centered at the origin and proportional to $\exp(-\|x\|_2^{1/\tau})$ outside the ball, then p may require local planar flows with depth $\Omega\left(d^{1/\tau-1}\right)$ to transform from an arbitrary source distribution (that is not too close). A similar conclusion holds for Householder flows when the target distribution is close to the standard Gaussian distribution. These results indicate that when local planar flows with certain non-linearities and Householder flows have moderate depth, they may have poor approximation power.

1.1.1 Related Work

There is a body of work on analyzing the approximation properties of neural networks [Cybenko, 1989, Hornik et al., 1989, Hornik, 1991, Montufar et al., 2014, Telgarsky, 2015, Lu et al., 2017, Hanin, 2017, Raghu et al., 2017]. Most of these results apply to feed-forward neural networks including non-invertible functions. Therefore, their universal approximation properties do not directly translate to normalizing flows.

The work most related to ours shows that a residual network (ResNet) in which each block is a single-neuron hidden layer with ReLU activation is a universal approximator in the space of Lebesgue integrable functions from $\mathbb{R}^d$ to $\mathbb{R}^d$ [Lin and Jegelka, 2018]. This is related to us because the set of all such ResNets with T invertible blocks is exactly T-layer ReLU planar flows. However, their construction that establishes this property is based on non-invertible mappings, consequently, their universal approximation result does not extend to planar flows.

There has also been some recent related work on the expressive power of generative networks. In particular, it was proved by construction that when the output dimension is equal to the input dimension, deep neural networks can approximately transform Gaussians to uniform distributions and vice versa [Bailey and Telgarsky, 2018]. However, their constructions are again based on non-invertible functions, and hence their results do not extend to normalizing flows.

Finally, there is also a body of empirical work on different kinds of normalizing flows; a more detailed discussion of these works is presented in Section 1.6.

1.2 Preliminaries

1.2.1 Definitions and Notation

Suppose d is the data dimension. Let $z \in \mathbb{R}^d$ be a random variable with density $q_z : \mathbb{R}^d \to \{0\} \cup \mathbb{R}^+$. Then, an invertible function $f : \mathbb{R}^d \to \mathbb{R}^d$ is called a *normalizing flow* if f is differentiable almost everywhere $(a.e.)$ and the determinant of the Jacobian matrix of f does not equal to zero:

$$\det J_f(z) \neq 0 \ (a.e.).$$

where $J_f(z)_{ij} = \frac{\partial f_i}{\partial z_j}$, $\forall i, j \in \{1, \cdots, d\}$. If we apply a flow f over z, we obtain a new random variable $y = f(z)$, whose density q_y can be written through the change-of-variable formula:

$$q_y(y) = \frac{q_z(z)}{|\det J_f(z)|}. \tag{1.1}$$

or

$$\log q_y(y) = \log q_z(z) - \log|\det J_f(z)|. \tag{1.2}$$

For conciseness, we write $q_y = f\#q_z$ in such context. In particular, if the flow f is composed of T *simple* flows $f_t, t = 1\cdots,T$:

$$f = f_T \circ f_{T-1} \circ \cdots \circ f_1.$$

then according to the chain rule of the Jacobian matrix, we have

$$\log q_y(y) = \log q_z(z) - \sum_{t=1}^{T} \log|\det J_{f_t}(z_{t-1})|. \tag{1.3}$$

where $z_0 = z,\ z_t = f_t(z_{t-1}),\ t = 1,\cdots,T$.

Two simple flows are defined below [Rezende and Mohamed, 2015b]:

Planar Flows. Given the scaling vector $u \in \mathbb{R}^d$, tangent vector $w \in \mathbb{R}^d$, shift $b \in \mathbb{R}$, and non-linearity $h : \mathbb{R} \to \mathbb{R}$, a planar flow f_{pf} on $\mathbb{R}^d$ is defined by

$$f_{\text{pf}}(z) = z + uh(w^\top z + b). \tag{1.4}$$

Radial Flows. Given the smoothing factor $a \in \mathbb{R}^+$, scaling factor $b \in \mathbb{R}$, and center $z_0 \in \mathbb{R}^d$, a radial flow f_{rf} on $\mathbb{R}^d$ is defined by

$$f_{\text{rf}}(z) = z + \frac{b}{a + \|z - z_0\|_2}(z - z_0). \tag{1.5}$$

A geometric intuition between planar and radial flows is shown in Section A.1. Planar flows can be generalized to a higher dimension below [Berg et al., 2018]:

Sylvester Flows. Given the flow dimension $m < d$, scaling matrix $A \in \mathbb{R}^{d\times m}$, tangent matrix $B \in \mathbb{R}^{d\times m}$, shift vector $b \in \mathbb{R}^d$, and non-linearity $h : \mathbb{R} \to \mathbb{R}$, a Sylvester flow f_{syl} on $\mathbb{R}^d$ is defined by

$$f_{\text{syl}}(z) = z + Ah(B^\top z + b). \tag{1.6}$$

where h maps coordinate-wise.

In addition, Householder matrices can also be used to construct flows [Tomczak and

Welling, 2016]:

Householder Flows. Given a unit reflection vector $v \in \mathbb{R}^d$, a Householder flow f_{hh} on $\mathbb{R}^d$ is defined by

$$f_{\text{hh}}(z) = z - 2vv^\top z. \tag{1.7}$$

For conciseness, we denote these flows by *base* flows.

1.2.2 Problem Statement

In this chapter, we study the expressivity of base flows in Section 1.2.1: given an input distribution q, we hope to understand when a flow f composed of a finite number of base flows can transform q into any target distribution p or its approximation on $\mathbb{R}^d$. Formally, suppose f is composed of T base flows in the same class. We propose to answer the following two questions:

Q1 (Exact transformation): Under what conditions is it possible to *exactly* transform q into p with a finite number of base flows? That is, $f\#q = p,\ (a.e.)$.

Q2 (Approximation): Since sometimes it may not be possible to exactly transform q into p, when is it possible to *approximate* p in total variation distance (which is equal to half of the ℓ_1 distance)? How many layers of base flows do we need? That is, given $\varepsilon > 0$, is there a bound for T such that

$$\|f\#q - p\|_1 \leq \varepsilon.$$

1.2.3 Additional Definitions and Notations

The determinant of the Jacobian matrix of a planar flow f_{pf}, a Sylvester flow f_{syl}, and a Householder flow f_{hh} can be easily calculated by

$$\begin{aligned} \det J_{f_{\text{pf}}}(z) &= 1 + u^\top w h'(w^\top z + b) \\ \det J_{f_{\text{syl}}}(z) &= \det(I_m + diag(h'(B^\top z + b))B^\top A) \\ \det J_{f_{\text{hh}}}(z) &= -1. \end{aligned} \tag{1.8}$$

In this chapter, we consider three types of non-linearities h: $\mathrm{relu}(x) = \max(x, 0)$, general differentiable functions, and local non-linearities (see Section 1.5 for detail) including $\tanh(x)$, $\arctan(x)$ and $\mathrm{sigmoid}(x) = 1/(1+\exp(-x))$. Specifically, let $h = \mathrm{ReLU}$ and $1\{\cdot\}$ be the indicator function, then $\det J_{f_{\mathrm{pf}}}$ is equal to

$$\det J_{f_{\mathrm{pf}}}(z) = 1 + u^\top w \cdot 1\{w^\top z + b \geq 0\}. \tag{1.9}$$

A ReLU planar/Sylvester flow is invertible under certain bounds on its parameters as ReLU is Lipschitz.

We make a few additional definitions here. $\mathcal{N}$ denotes a Gaussian distribution on $\mathbb{R}^d$:

$$\mathcal{N}(x; \mu, \Sigma) = \frac{\exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)}{(2\pi)^{d/2}\sqrt{\det\Sigma}}.$$

The set **supp** p denotes the support of distribution p:

$$\textbf{supp } p = \{x \in \mathbb{R}^d : p(x) > 0\}.$$

For vectors $w_i \in \mathbb{R}^d, 1 \leq i \leq k$, the **span** of them denotes the subspace spanned by $\{w_i\}_{i=1}^k$:

$$\textbf{span}\{w_1, \cdots, w_k\} = \left\{\sum_{i=1}^k \alpha_i w_i : \ \alpha_i \in \mathbb{R}, 1 \leq i \leq k\right\}.$$

The **span** of a set of matrices is defined as the span of the union of their column vectors. For any differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ and direction $\delta \in \mathbb{R}^d \setminus \{0\}$, its corresponding directional derivative is defined by

$$\lim_{\alpha \to 0} \frac{g(x+\alpha\delta) - g(x)}{\alpha} = \nabla_x g(x)^\top \delta.$$

1.2.4 Challenges

The main challenge in analyzing whether a class of flows can universally approximate any target distribution when applied to a fixed source is *invertibility*. To understand this, suppose $\mathscr{F},\mathscr{C}$ are function classes and $\mathscr{I}$ is the set of all invertible functions.

Even if $\mathscr{F}$ can approximate any function in $\mathscr{C}$, it might not hold that the invertible functions in $\mathscr{F}$ can approximate any invertible function in $\mathscr{C}$. This is because the set of invertible functions $\mathscr{I}$ might have no interior in $\mathscr{C}$: for any invertible function, it is possible to modify it slightly to make it non-invertible – and hence the approximation to an invertible function $c \in \mathscr{C}$ may be a non-invertible function $f \in \mathscr{F}$ (see **Lemma** 4, [Mulansky and Neamtu, 1998]). For instance, it was shown that a certain ResNet ($\mathscr{F}$) is a universal approximator in $\mathscr{C} = \ell_1(\mathbb{R}^d)$ [Lin and Jegelka, 2018], and its invertible function subset ($\mathscr{F} \cap \mathscr{I}$) is exactly the set of transformations composed of finitely many ReLU planar flows. However, since the universal approximation property was proved by construction using the non-invertible trapezoid functions, this result does not translate to ReLU planar flows.

Dually, if $\mathscr{F}$ has limited expressivity, it might still happen that functions in $\mathscr{F} \cap \mathscr{I}$ can approximate or even express transformations between arbitrary pairs of distributions. This is because a small subset of functions $\mathscr{T}$ (for instance, increasing triangular maps [Villani, 2008]) is enough to transform between distributions. Therefore, if $\mathscr{F} \cap \mathscr{I}$ is dense in $\mathscr{T}$, then it is expressive. It is however challenging to find all such dense sets $\mathscr{T}$.

1.3 The $d = 1$ case

In this section, we discuss the universal approximation properties of Sylvester flows when the data dimension $d = 1$. In this case, a Sylvester flow is identical to a planar flow. However, the one-dimensional case is not trivial and requires delicate design. For both general and ReLU non-linearity cases, we demonstrate they are able to achieve universal approximation.

1.3.1 General Smooth Non-linearity

Suppose the flow f is a single planar flow with an arbitrary smooth non-linearity h. It is straightforward to show by construction that if **supp** $p =$ **supp** $q = \mathbb{R}$, then there exists a planar flow that exactly transforms q into p. (See **Lemma** A.2.2). Using these exact transformations, we can approximate any density supported on a finite union of intervals when the input distribution is supported on $\mathbb{R}$ (e.g. a Gaussian).

Theorem 1.3.1 (Universal Approximation)**.** *Let p, q be densities on $\mathbb{R}$ such that p is supported on a finite union of intervals and* **supp** $q = \mathbb{R}$*. Then, for any $\varepsilon > 0$, there exists a planar flow f_{pf} such that $\|f_{\text{pf}}\#q - p\|_1 \leq \varepsilon$.*

Since in **Theorem** 1.3.1, the support of p might not be $\mathbb{R}$, we are unable to achieve exact transformation between p and q. However, approximation is possible in that we can transform q into $\tilde{p}$, a distribution supported on $\mathbb{R}$ but approximates p in ℓ_1 norm. To achieve this, we construct such $\tilde{p}$ that satisfying $\tilde{p} \approx p$ on **supp** p and $\tilde{p} \approx 0$ on $\overline{\textbf{supp}\ p}$. An example is shown in Figure 1.1, where $p(x) = \frac{3}{4}\min((|x|-1)^2, (|x|-3)^2)$ for $1 \leq |x| \leq 3$ and $p(x) = 0$ elsewhere.

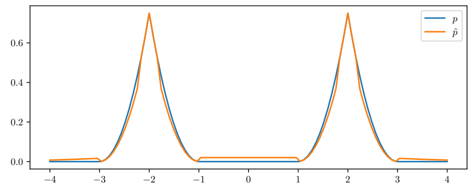


Figure 1.1. Target distribution p and its approximation $\tilde{p}$ with **supp** $\tilde{p} = \mathbb{R}$.

1.3.2 ReLU Non-linearity

Since the ReLU activation has been proven to be expressive and is popular in recent neural network models [He et al., 2016b, Lin and Jegelka, 2018], we provide a universal approximation result for planar flows with ReLU non-linearity.

Suppose the one-dimensional ReLU flow has the form $f(z) = f_{\text{pf}}(z) = z + uh(wz + b)$, where $h = \text{relu}$. Since ReLU is linear on both $\mathbb{R}^-$ and $\mathbb{R}^+$, we assign $u = \pm 1$ for concreteness. In addition, to ensure the transformation is strictly increasing, we require $uw > -1$. Different from the general non-linearity case, the determinant of $\det J_f$ in (1.9) indicates that a ReLU planar flow keeps a halfspace of $\mathbb{R}$ and applies linear scaling transformation to the other halfspace.

Given that the input distribution q is Gaussian, we prove it is possible to approximate any density supported on a finite union of intervals in ℓ_1 norm using a finite number of ReLU planar flows.

Theorem 1.3.2 (Universal Approximation)**.** *Let p be a density on $\mathbb{R}$ supported on a finite union of intervals. Then, for any $\varepsilon > 0$, there exists a flow f composed of finitely many ReLU planar flows and a Gaussian distribution $q_{\mathcal{N}}$ such that $\|f\#q_{\mathcal{N}} - p\|_1 \leq \varepsilon$.*

There are two steps in the proof. First, we show that Gaussian distributions can be exactly transformed to tail-consistent piecewise Gaussian distributions (see **Definition** A.3.2, **Definition** A.3.3 for formal definitions and **Lemma** A.3.5). An example of a tail-consistent piecewise Gaussian distribution of three pieces is shown in Figure 1.2: the distribution is composed of three Gaussian pieces in full lines of three colors, where the dashed lines are corresponding prolongations. Then, the area below yellow lines (—/- -) is equal to the area below the blue dashed line (- -), and the area below the green full line (—) is equal to the area below the yellow dashed line (- -).

In the second step, we show that tail-consistent piecewise distributions can approximate any piecewise constant distribution supported on a finite union of compact intervals (see **Lemma** A.3.6). Notice that piecewise constant functions supported on a finite union of compact intervals can approximate any Lebesgue-integrable function [Lin and Jegelka, 2018], so do densities supported on a finite union of intervals. Therefore, the universal approximation property of ReLU planar flows (**Theorem** 1.3.2) is obtained.

In Figure 1.3, two examples are presented on approximating the same target distribution p with different number of ReLU planar flows. As illustrated, the approximation almost reaches perfection when we choose a larger number of ReLU planar flows.

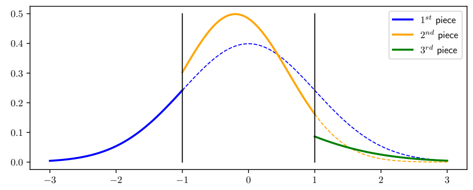


Figure 1.2. A tail-consistent piecewise Gaussian distribution in $\mathscr{PW}(3,\mathscr{G})$.

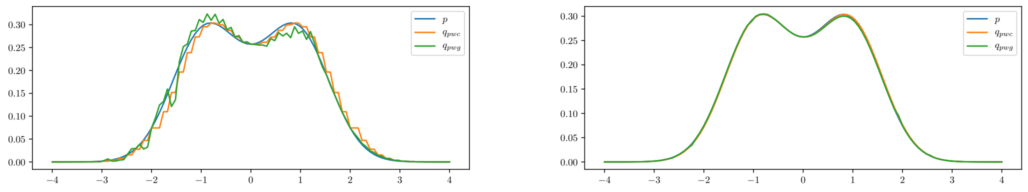


Figure 1.3. Target distribution p, its piecewise constant distribution approximation q_{pwc} of 50 (top)/300 (bottom) pieces, and its tail-consistent piecewise Gaussian distribution approximation q_{pwg} generated by 50 (top)/300 (bottom) ReLU planar flows over a Gaussian.

Remark 1.3.3. *Since we can transform the standard Gaussian distribution $\mathscr{N}(0,1)$ to any other Gaussian distribution using a scaling function, which can be achieved by two ReLU planar flows and a shift, we can further assign the input distribution $q_{\mathscr{N}}$ in* ***Theorem*** *1.3.2 to be the standard Gaussian distribution.*

1.4 Exact Transformation for $d > 1$

In this section, we consider the exact transformation question when the data dimension $d > 1$. We study two cases where the flow is composed of a finite number of Sylvester flows with (i) ReLU non-linearity and (ii) general non-linearity. We specifically show how the topology matching conditions yield negative results to the exact transformation question (that is, to show there does not exist such flow that can transform between certain distributions).

Our results are based on the following key observation for a flow $f : \mathbb{R}^d \to \mathbb{R}^d$. For almost every $z \in \mathbb{R}^d$ there exists a subspace $\mathscr{V}(z) \subset \mathbb{R}^d$ such that for any $v \in \mathscr{V}$ and small $\alpha > 0$, $\det J_f(z) = \det J_f(z + \alpha v)$. We call $\mathscr{V}$ the complementary subspace of f at z. This observation can be used to determine what class of distributions flows can transform between. By letting $\alpha \to 0$, we can focus on properties of small neighbourhoods around z, which we call *topology* *matching*.

1.4.1 ReLU Non-linearity

We begin with constructing a topology matching condition for ReLU Sylvester flows: $f(z) = f_{\text{syl}}(z) = Z + A\ \text{relu}(B^\top z + b)$. (1.8) shows that for a single ReLU Sylvester flow, if $B^\top z + b \neq 0$, then $\det J_f(z') = \det J_f(z)$ when z' is close to z. This statement can be further generalized: if f is a flow composed of a finite number of ReLU Sylvester flows, for almost every $z \in \mathbb{R}^d$, the determinant of the Jacobian of f is a constant near z. Based on this observation, we conclude that the complementary subspace $\mathscr{V}(z) = \mathbb{R}^d$, *a.e.* (see **Lemma** A.4.1). Using this property, we construct the topology matching condition in the following theorem.

Theorem 1.4.1 (Topology Matching for ReLU Sylvester flows)**.** *Suppose distribution q is defined on $\mathbb{R}^d$, and flow f is composed of finitely many ReLU Sylvester flows on $\mathbb{R}^d$. Let $p = f\#q$. Then, there exists a zero-measure closed set $\Omega \subset \mathbb{R}^d$ such that $\forall z \in \mathbb{R}^d \setminus \Omega$, we have*

$$J_f(z)^\top \nabla_z \log p(f(z)) = \nabla_z \log q(z).$$

Intuitively, the local directional derivatives of the logarithm of the density are preserved. As a special case, if z satisfies $\nabla_z q(z) = 0$ (which means that z is a local minima, local maxima, or saddle point of q), then $p(f(z))$ must also have zero gradient at z. For instance, suppose p is the standard Gaussian distribution on $\mathbb{R}^2$ and q is a mixture of two Gaussian distributions on $\mathbb{R}^2$ with two peaks. Since only at the origin does p have zero gradient, we conclude there does not exist a planar flow that transforms q to p. Additional examples are illustrated in Figure A.3 in the Appendix.

The proof of **Theorem** 1.4.1 follows from (1.2), the Taylor expansion of f, and the observation that $\mathscr{V}(z) = \mathbb{R}^d$ *a.e.*. Notably, the conclusion holds for any number of ReLU Sylvester flows. Using this condition, we show in the following corollaries that it is unlikely for finitely many ReLU Sylvester flows to transform between mixture of Gaussian (MoG) or product (Prod) distributions unless special conditions are satisfied.

Corollary 1.4.2 (MoG$\not\twoheadrightarrow$MoG)**.** *(See formal version in **Corollary** A.5.1) Suppose p, q are mixture of Gaussian distributions on $\mathbb{R}^d$ in the following form:*

$$p(z) = \sum_{i=1}^{r_p} w_p^i \mathcal{N}(z; \mu_p^i, \Sigma_p), \ q(z) = \sum_{j=1}^{r_q} w_q^j \mathcal{N}(z; \mu_q^j, \Sigma_q).$$

Then, there generally does not exist flow f composed of finitely many ReLU Sylvester flows such that $p = f\#q$.

Corollary 1.4.3 (Prod$\not\twoheadrightarrow$Prod)**.** *(See formal version in **Corollary** A.6.1) Suppose p and q are product distributions in the following form:*

$$p(z) \propto \prod_{i=1}^{d} g(z_i)^{r_p}; \ q(z) \propto \prod_{i=1}^{d} g(z_i)^{r_q}.$$

where $r_p, r_q > 0, r_p \neq r_q$, and g is a smooth function. Then, there generally does not exist flow f composed of finitely many ReLU Sylvester flows such that $p = f\#q$.

Given our negative results, the reader might wonder what distributions can be transformed by ReLU Sylvester flows. We show that certain linear transformations can be exactly expressed (see **Theorem** A.7.1, **Corollary** A.7.2 and **Corollary** A.7.3).

1.4.2 General Smooth Non-linearity

In this section, we construct a topology matching condition for Sylvester flows with general non-linearities. Suppose f is a Sylvester flow $f(z) = z + Ah(B^\top z + b)$ with flow dimension m, where h is an arbitrary smooth function. Analogous to **Theorem** 1.4.1, there exists a $d - m$ dimensional complementary subspace of f at every point $z \in \mathbb{R}^d$: $\mathcal{V}(z) = \textbf{span}\{B\}^\perp$. Using this property, we are able to establish the topology matching condition for a single Sylvester flow (see **Lemma** A.8.1). Then, we generalize this result to n layers of Sylvester flows in the following theorem.

Theorem 1.4.4 (Topology Matching for Sylvester flows)**.** *Suppose distribution q is defined on $\mathbb{R}^d$, and n Sylvester flows $\{f_i\}_{i=1}^n$ on $\mathbb{R}^d$ have flow dimensions $\{m_i\}_{i=1}^n$, tangent matrices $\{B_i\}_{i=1}^n$, and smooth non-linearities. Let $f = f_n \circ \cdots \circ f_1$ and $p = f\#q$. Then $\forall z \in \mathbb{R}^d$, we have*

$$\nabla_z \log p(f(z)) - \nabla_z \log q(z) \in \mathbf{span}\{B_1, B_2, \cdots, B_n\}.$$

When the sum of flow dimensions of $\{f_i\}_{i=1}^n$ is strictly less than the data dimension d, $\mathbf{span}\{B_1, B_2, \cdots, B_n\}$ is a strict subspace of $\mathbb{R}^d$. Under this situation, we show in the following corollary that transformation between Gaussian distributions might be impossible with a bounded number of Sylvester flows.

Corollary 1.4.5 ($\mathcal{N} \nrightarrow \mathcal{N}$)**.** *(See formal version in **Corollaries** A.9.1 and A.9.2) Let $p \sim \mathcal{N}(0, \Sigma_p), q \sim \mathcal{N}(0, \Sigma_q)$ be two Gaussian distributions on $\mathbb{R}^d$, and $\Sigma_q^{-1} - \Sigma_p^{-1}$ has high rank. Then, with a limited number of planar or Sylvester flows that have smooth non-linearities, it is impossible to transform q to p.*

Additional experiments are demonstrated in Figure A.4 in the Appendix. We also construct a topology matching condition for radial flows in **Theorem** A.10.1, and compare that result with **Theorem** 1.4.4.

1.5 Approximation Capacity for Large d

In this section, we provide a partially negative answer to the universal approximation question for certain normalizing flows by showing that approximations in these cases may require very deep flows. In particular, we study local planar flows and Householder flows with specific target distributions.

Given an input distribution q and a target distribution p on $\mathbb{R}^d$, our goal is to lower bound the depth T of a normalizing flow that can transform q to an approximation of p. This is formally defined below.

Definition 1.5.1. *Let p,q be two distributions on $\mathbb{R}^d$, $\varepsilon > 0$, and $\mathscr{F}$ be a set of normalizing flows. Then, the minimum number of flows in $\mathscr{F}$ required to transform q to an approximation of p to within ε is*

$$\begin{aligned} T_\varepsilon(p,q,\mathscr{F}) = \inf\{n: \ & \exists \{f_i\}_{i=1}^n \in \mathscr{F} \text{ such that} \\ & \|(f_1 \circ \cdots \circ f_n)\#q - p\|_1 \le \varepsilon\}. \end{aligned}$$

To achieve this goal, we look at the maximum ℓ_1 norm distance reduction of a normalizing flow f towards p:

$$\mathscr{L}(p,f) = \sup_{q' \text{ is a density on } \mathbb{R}^d} \|p - q'\|_1 - \|p - f\#q'\|_1.$$

We first show a surprisingly concise upper bound $\hat{\mathscr{L}}$ of $\mathscr{L}$. This bound is used in proving **Theorem** 1.5.5 and **Theorem** 1.5.6 in this section.

Lemma 1.5.2. *$\mathscr{L}(p,f) \le \hat{\mathscr{L}}(p,f)$, where*

$$\hat{\mathscr{L}}(p,f) = \int_{\mathbb{R}^d} \big| |\det J_f(z)| p(f(z)) - p(z) \big| dz.$$

Then, we naturally obtain a lower bound of T:

$$T_\varepsilon(p,q,\mathscr{F}) \ge \frac{\|p-q\|_1 - \varepsilon}{\sup_{f \in \mathscr{F}} \mathscr{L}(p,f)} \ge \frac{\|p-q\|_1 - \varepsilon}{\sup_{f \in \mathscr{F}} \hat{\mathscr{L}}(p,f)}.$$

Next, we make the following assumption on q:

Assumption 1.5.3. $\|p-q\|_1 = \Theta(1)$.

This assumption holds when the input distribution q is a random initialization (that is, q is chosen arbitrarily without any prior knowledge on p). Then, under **Assumption** 1.5.3, there exists $\varepsilon > 0$ (e.g. $\varepsilon = \frac{1}{2}\|p-q\|_1$) such that

$$T_\varepsilon(p,q,\mathscr{F}) = \Omega\left(\frac{1}{\sup_{f \in \mathscr{F}} \hat{\mathscr{L}}(p,f)}\right).$$

In the rest of this section, we use this lower bound on T to construct results for local planar flows and Householder flows with specific target distributions.

1.5.1 Local Planar Flows

In this section, we look at a specific group of planar flows, which we call the *local* planar flows. A c_h-local planar flow is defined below.

Definition 1.5.4. *A non-linearity h is called c_h-local if there is a constant $c_h \in \mathbb{R}$ satisfying for any $x \in \mathbb{R}$, (i) $|h(x)| \leq c_h$, and (ii) $|h'(x)| \leq c_h/(1+|x|)$. A planar flow $f(z) = z + uh(w^\top z + b)$ is called c_h-local if h is c_h-local, $\|u\|_2 \leq 1$, and $\|w\|_2 \leq 1$.*

Many popular non-linearities are c_h-local, such as tanh ($c_h = 2$), sigmoid ($c_h = 1$), and arctan ($c_h = \pi/2$).

Geometrically, a local planar flow applies non-linear scaling on the region near the $d-1$ dimensional subspace $\{z : w^\top z + b = 0\}$ in $\mathbb{R}^d$, while having little effect on regions far away from the subspace (almost a constant shift). This observation leads to the intuition that one layer of local planar flow can only affect a small volume of the whole space, so a large number of layers is needed to approximate the target distribution if **supp** p is a large region. In the following theorem, we show for certain p, T goes up polynomially in the data dimension d with adjustable degrees.

Theorem 1.5.5 (ℓ_1 norm approximation lower bound for local planar flows)**.** *Let p be a distribution on $\mathbb{R}^d$ ($d > 2$) such that for $\tau \in (0,1)$:*

- *$p = \mathscr{O}(p_1)$, where density p_1 satisfies*

$$p_1(z) \propto \exp(-\|z\|_2^{\tau}).$$

- $\|\nabla p\|_2 = \mathscr{O}(\|\nabla p_2\|_2)$, *where density* p_2 *satisfies*

$$p_2(z) \propto \begin{cases} \exp(-d) & \|z\|_2 \leq d^{\frac{1}{\tau}} \\ \exp(-\|z\|_2^{\tau}) & \|z\|_2 > d^{\frac{1}{\tau}} \end{cases}.$$

Suppose $\mathscr{F}$ *is the set of all* c_h*-local planar flows. Then, under* ***Assumption*** *1.5.3, there exists* $\varepsilon = \Theta(1)$ *such that*

$$T_\varepsilon(p,q,\mathscr{F}) = \Omega\left(\min\left((\log d)^{-\frac{1}{\tau}} d^{\left(\frac{1}{\tau}-\frac{1}{2}\right)}, d^{\left(\frac{1}{\tau}-1\right)}\right)\right).$$

This indicates that if the target distribution p has specifically bounded values and gradients, a large number of local planar flows is needed to approximate p starting with a distribution q that obeys **Assumption** 1.5.3. The number T is polynomial in d with adjustable degrees, so it can be incredibly large as d gets large.

A concrete example that satisfies the condition in **Theorem** 1.5.5 is when $p(z)$ is equal to the p_2 in the statement. This satisfies the first condition because $\exp(-d) \leq \exp(-\|z\|_2^{\tau})$ in the ball centered at the origin with radius $d^{1/\tau}$, and the integration of p_1 in this ball is $o(1)$ (see proof of **Lemma** A.12.1). Then, taking for instance $\tau = 0.2$, the lower bound on T becomes $\Omega(d^4)$, which is incredibly large in practical scenarios.

To prove **Theorem** 1.5.5, we first show that $\hat{\mathscr{L}}(p,f)$ is upper bounded by an integration of two terms. We then present **Lemma** A.12.1 and **Lemma** A.12.2 to bound these two terms separately.

1.5.2 Householder Flows

In this section, we look at Householder flows. Since a Householder matrix does not change the ℓ_2 norm of any vector, it is possible to upper bound $\mathscr{L}$ when the target distribution p is almost symmetric, according to **Lemma** 1.5.2. If p is a standard Gaussian distribution, we have $\mathscr{L} = 0$, indicating that Householder flows cannot transform any different distribution to a standard Gaussian distribution. In the following theorem, we provide a concise bound on T when p is very close to the standard Gaussian distribution, where there is only a small perturbation on its covariance matrix.

Theorem 1.5.6 (ℓ_1 norm approximation lower bound for Householder flows)**.** *Let p be a Gaussian distribution $\mathscr{N}(0, I+S)$ on $\mathbb{R}^d$ $(d>2)$, where $|S_{ij}| \le d^{-(2+\kappa)}$ for some $\kappa > 0$ and any $1 \le i, j \le d$. Suppose $\mathscr{F}$ is the set of all Householder flows. Then, under **Assumption** 1.5.3, there exists $\varepsilon = \Theta(1)$ such that*

$$T_\varepsilon(p, q, \mathscr{F}) = \Omega\left(d^\kappa\right).$$

This indicates that we need a large number of Householder flows to approximate a distribution close to the standard Gaussian distribution, starting with a distribution q that obeys **Assumption** 1.5.3. The number T is also polynomial in the data dimension d with adjustable degrees, so it could be large as well. The bound is computed from $\hat{\mathscr{L}}$, where $|\det J_f(z)| = 1$ for a Householder flow f.

1.6 Additional Related Work

1.6.1 Normalizing Flows

It was shown that transforming a simple distribution to a complicated one by composing many simple transformations can be used to solve density estimation problems [Tabak and Vanden-Eijnden, 2010, Tabak and Turner, 2013]. These transformations are called *normalizing flows*. Two basic normalizing flows (planar and radial flows) were introduced [Rezende and Mohamed, 2015b]. Due to their empirical success, there has been a growing body of work on other kinds of normalizing flows. Two categories of normalizing flows have been developed.

Triangular flows. It was proven that increasing triangular functions can transform between arbitrary distributions [Villani, 2008]. Therefore, triangular flows composed of fixed classes of increasing triangular functions are expected to enjoy good expressive power. In addition, the determinant of the Jacobian matrix of an increasing triangular function is easy to compute. These two benefits have led to the development of a large family of triangular flows [Dinh et al., 2014, Germain et al., 2015, Uria et al., 2016, Kingma et al., 2016, Dinh et al., 2016, Papamakarios et al., 2017, Huang et al., 2018, Jaini et al., 2019]. Among these flows, IAF [Kingma et al., 2016], NAF [Huang et al., 2018] and SOS flows [Jaini et al., 2019] were shown to have the universal approximation property.

Non-triangular flows. It is possible to calculate the determinant of the Jacobian matrix and the inverse of a well designed non-triangular function. Several flows parameterized by matrices were inspired by results from linear algebra and thus enjoy this property [Tomczak and Welling, 2016, Hasenclever et al., 2017, Ho et al., 2019, Berg et al., 2018], where the last one is a matrix-form generalization of the planar flow. Moreover, a recent non-triangular flow, the iResNet [Behrmann et al., 2018], in the form of residual networks (ResNet) [He et al., 2016b], was designed with an efficient log-det approximator. It was further improved in residual flows with an unbiased approximator [Chen et al., 2019]. However, the expressivity of these flows still remain unknown, even though the iResNet is expressed by powerful neural networks.

1.6.2 Continuous Time Flows

It is possible, from the infinitesimal point of view, to generalize the discrete update of finite flows to continuous update of infinite flows. Infinite flows are described by a differential equation instead of a sequence of transformations in the finite flow context [Chen et al., 2017, Grathwohl et al., 2018, Chen et al., 2018, Salman et al., 2018, Zhang and Wang, 2018]. The neural ODEs [Chen et al., 2018] is one significant work in this class, but its expressivity still lacks understanding. A counter-example was provided on the expressivity of the neural ODEs [Dupont et al., 2019]. However, this does not rigorously imply that neural ODEs are not universal approximators because (i) the failure in exact transformation does not imply the impossibility in approximation, and (ii) universal transformation does not necessarily need universal function representation.

To tackle the problem of such counter-example, additional p dimensions were introduced to ”augment” the neural ODEs [Dupont et al., 2019]. By solving a $d+p$ dimensional augmented ODE and extracting the first d dimensions, the expressivity of the neural ODEs is enhanced. It was further shown that the augmented neural ODEs is a universal approximator in the continuous function space when $p=1$ [Zhang et al., 2019a]. Nevertheless, in the context of normalizing flows, every transformation has to be invertible, so the change of dimension strategy, as well as its universal approximation property, does not apply to normalizing flows.

1.7 Conclusions

Normalizing flows are a class of deep generative models that offer flexible generative modeling as well as easy likelihood computation. While there has been a great deal of prior empirical work on different normalizing flow models, not much is (formally) known about their expressive power; we provide one of the first systematic studies on non-triangular flows. Our results demonstrate that one needs to be careful while designing normalizing flow models as well as their non-linearities in high dimensional space. In particular, we show that Sylvester flows, a universal approximator in one dimension, are unable to exactly transform between two (even simple) distributions unless rigorous conditions are satisfied. Additionally, a prohibitively large number of layers of planar or Householder flows are required to reduce the ℓ_1 distance between input and output distributions under certain conditions.

There are a large number of open problems. Some unresolved problems towards expressivity of simple flows include (i) are certain combinations of tangent matrices or non-linearities useful, (ii) can normalizing flows composed of finitely many $(\geq d)$ Sylvester flows with arbitrary non-linearities (or other simple flows) transform between any pair of input-output distributions in high dimensional space, (iii) are such normalizing flows universal approximators in converting distributions, and (iv) what class of distributions are easy or hard for normalizing flows composed of Sylvester flows or other simple flows to transform between. A final open problem is to look at other, more general classes of flows, and provide upper and lower bounds on their expressive power under different non-linearities.

1.8 Acknowledgements

We thank NSF under IIS 1617157 for research support.

This chapter is a reformatted version of the paper ”The Expressive Power of a Class of Normalizing Flow Models” published in *International Conference on Artificial Intelligence and Statistics, 2020,* by Zhifeng Kong and Kamalika Chaudhuri [Kong and Chaudhuri, 2020]. The dissertation author was the primary researcher and author of this paper.

Chapter 2

Universal Approximation of Residual Flows in Maximum Mean Discrepancy

2.1 Introduction

Normalizing flows are a class of generative models that learn an invertible function to transform a predefined source distribution into a complex target distribution [Tabak and Vanden-Eijnden, 2010, Tabak and Turner, 2013, Rezende and Mohamed, 2015b]. One category of normalizing flows called residual flows use residual networks [He et al., 2016b] to construct the transformation [Rezende and Mohamed, 2015b, Van Den Berg et al., 2018, Behrmann et al., 2019, Chen et al., 2019]. These models have shown great success in complicated real-world tasks.

However, to ensure invertibility, these models apply additional Lipschitz constraints to each residual block. Under these strong constraints, how expressive these models are remains an open question. Formally, can they approximate certain target distributions to within any small error?

In this chapter, we carry out a theoretical analysis on the expressive power of residual flows. We prove there exists a residual flow F that achieves universal approximation in the mean maximum discrepancy (MMD, [Gretton et al., 2012]) metric. Formally, given a target distribution, we provide upper bounds on the number of residual blocks in F such that applying F over the source distribution can approximate the target distribution in squared MMD (see

(2.4)).

Although residual networks are universal approximators [Lin and Jegelka, 2018], the proof of approximation uses a non-invertible construction and therefore does not apply to residual flows. This reflects the main difficulty in analyzing residual flows: under strong Lipschitz and invertibility constraints, they become a very restricted function class. As an illustration, take the set of piecewise constant functions. Classical real analysis shows that piecewise constant functions can approximate any Lebesgue-integrable function and therefore any probability density. However, the invertible subset of all piecewise constant functions is the empty set! Consequently, this universal approximation result does not apply to normalizing flows. This difficulty leads to many negative results for normalizing flows: they are either unable to express or find it hard to approximate certain functions [Zhang et al., 2019a, Koehler et al., 2020, Kong and Chaudhuri, 2020].

To tackle this problem, we adopt a new construction that satisfies the strong Lipschitz constraints in Behrmann et al. [2019]. Specifically, we construct the residual blocks by multiplying a small ε to a pre-specified Lipschitz function. Therefore, as long as ε is small enough, the strong Lipschitz constraints are satisfied. We then analyze the following quantity: how much can the MMD be reduced if a new residual block is appended? Since this quantity is a function of ε, we can analyze its Taylor expansion. With a first-order analysis and under mild conditions, we show there is an F with $\Theta\left(\frac{1}{\delta}\left(\log\frac{1}{\delta}\right)^2\right)$ residual blocks that achieves (2.4) (see **Theorem** 2.4.5), where δ is the ratio between the final squared MMD and the initial squared MMD. With a second-order analysis and under more conditions, we show there is a shallower F with only $\Theta\left(\log\frac{1}{\delta}\right)$ residual blocks that achieves (2.4) (see **Theorem** 2.5.2).

To sum up, we show residual flows are universal approximators in MMD under certain assumptions and provide explicit bounds on the number of residual blocks.

2.2 Related Work

The classic universal approximation theory for fully connected or residual neural networks in the function space are widely studied [Cybenko, 1989, Hornik et al., 1989, Hornik, 1991, Montufar et al., 2014, Telgarsky, 2015, Lu et al., 2017, Hanin, 2017, Raghu et al., 2017, Lin and Jegelka, 2018]. However, these results do not generalize to residual flows [Rezende and Mohamed, 2015b, Van Den Berg et al., 2018, Behrmann et al., 2019, Chen et al., 2019] for two reasons. First, the approximation theory for normalizing flows analyzes how well they can transform between *distributions*, rather than their ability to approximate a target function in the *function space*. Despite that L^p universality in the function space may lead to distributional universality for triangular flows [Teshima et al., 2020], there is no similar results for non-triangular flows including residual flows. Second, the classic results do not consider the invertibility or the Lipschitz constraints of the neural networks, which greatly restrict the expressive power.

There are also universal approximation results for Lipschitz networks [Anil et al., 2019, Cohen et al., 2019, Tanielian et al., 2020]. These results are related because in this work, we assume the expressive power of each Lipschitz residual block is large. However, these results only apply to functions defined on compact sets. Because compact sets are bounded, it is "easier" to satisfy the Lipschitz constraints. It is not trivial to extend their results to functions defined on $\mathbb{R}^d$.

Concerning the expressive power of generative networks, there are prior works showing feed-forward generator networks can approximate certain distributions [Lee et al., 2017, Bailey and Telgarsky, 2018, Lu and Lu, 2020, Perekrestenko et al., 2020]. However, the results are again based on non-invertible constructions, so they do not apply to normalizing flows.

In the literature of normalizing flows, there are universal approximation results for several models including autoregressive flows [Germain et al., 2015, Kingma et al., 2016, Papamakarios et al., 2017, Huang et al., 2018, Jaini et al., 2019], coupling flows [Teshima et al., 2020, Koehler et al., 2020], and augmented normalizing flows [Zhang et al., 2019a, Huang et al., 2020] [1]. There is also a continuous-time generalization of normalizing flows called neural ODEs [Chen et al., 2018, Dupont et al., 2019] with a universal approximation result [Zhang et al., 2019a]. We do not consider these flows in this chapter. In addition, Müller [2020] suggests residual networks can approximate neural ODEs, but the invertibility is again not considered in this case.

On the expressive power of residual flows, all existing theoretical analysis present negative results for these models [Zhang et al., 2019a, Koehler et al., 2020, Kong and Chaudhuri, 2020]. These results indicate residual flows are either unable to express certain functions, or unable to approximate certain distributions even with large depths. Compared to these results, our work presents positive results for standard residual flows: given a source distribution q, they can approximate a target distribution p in the MMD metric [Gretton et al., 2012] under certain conditions. We provide explicit upper bounds on the number of residual blocks (see **Theorem** 2.4.5 and **Theorem** 2.5.2).

2.3 Preliminaries

We first define the maximum mean discrepancy (MMD) metric between distributions below.

Definition 2.3.1 (MMD, [Gretton et al., 2012]). *Let q, p be two distributions on $\mathbb{R}^d$. Then,*

$$\begin{aligned} \mathrm{MMD}(q,p)^2 = \ & \mathbb{E}_{z,z'\sim q}K(z,z') + \mathbb{E}_{x,x'\sim p}K(x,x') \\ & -2\cdot\mathbb{E}_{z\sim q,x\sim p}K(z,x) \end{aligned} \tag{2.1}$$

for some kernel function $K(\cdot,\cdot)$. Let $\phi:\mathbb{R}^d\to\mathbb{R}^{d_\phi}$ be the feature map associated with K: $K(x,z)=\phi(x)^\top\phi(z)$, where we assume $d_\phi<\infty$. Then, the squared MMD can be simplified as

$$\mathrm{MMD}(q,p)^2 = \|\mathbb{E}_{z\sim q}\phi(z) - \mathbb{E}_{x\sim p}\phi(x)\|_2^2. \tag{2.2}$$

[1]In an augmented normalizing flow, there is an auxiliary random variable concatenated with the data, so the transformations operate on a higher dimensional space.

Next, we define a residual flow as a composition of invertible layers parameterized as $\mathbf{Id} + f$, where $\mathbf{Id}$ is the identity map and f is $\frac{1}{2}$-Lipschitz[2]. The class of residual flows include planar flows [Rezende and Mohamed, 2015b], Sylvester flows [Van Den Berg et al., 2018], and the more general invertible residual networks [Behrmann et al., 2019, Chen et al., 2019]. In these models every f_i is parameterized as a certain kind of fully-connected neural network. Since the expressive power of ($\frac{1}{2}$-)Lipschitz neural networks on $\mathbb{R}^d$ remains an open problem, in this chapter we assume every f_i can be selected as any $\frac{1}{2}$-Lipschitz function. Formally, we make the following definition.

Definition 2.3.2 (Residual flows)**.** *The set of N-block residual flows is defined as*

$$\mathscr{F}_N = \left\{ (\mathbf{Id} + f_N) \circ \cdots \circ (\mathbf{Id} + f_1) : \textit{ each } f_i \textit{ is } \frac{1}{2}\textit{-Lipschitz} \right\}. \tag{2.3}$$

Now we state the main problem. Let q_{source} and p_{target} be two distributions on $\mathbb{R}^d$, where q_{source} is the source distribution and p_{target} is the target distribution. We aim to answer the following problem in this chapter.

Problem Statement. *Let $\delta > 0$ be a small number. For any pair of distributions q_{source} and p_{target} on $\mathbb{R}^d$ satisfying $\mathrm{MMD}(q_{\text{source}}, p_{\text{target}}) < \infty$, does there exist an N and $F \in \mathscr{F}_N$ such that*

$$\mathrm{MMD}(F\#q_{\text{source}}, p_{\text{target}})^2 \leq \delta \cdot \mathrm{MMD}(q_{\text{source}}, p_{\text{target}})^2, \tag{2.4}$$

where $F\#q$ refers to the distribution obtained by applying F over q?

In this chapter, we prove existence of such F with a loose bound on N using first-order analysis under mild assumptions (see Section 2.4), and provide a tighter bound on N using second-order analysis under more assumptions (see Section 2.5).

[2]According to the fixed-point theorem, $\mathbf{Id} + f$ is invertible as long as the Lipschitz constant of f is strictly less than 1. For algebraic convenience, we restrict the Lipschitz constant to be at most $\frac{1}{2}$.

2.4 A Bound with First-Order Analysis

In this section, we show under mild conditions, there exists a residual flow F with $N = \Theta\left(\frac{1}{\delta}\left(\log\frac{1}{\delta}\right)^2\right)$ residual blocks that achieves (2.4). The idea is to show that a single residual block can reduce the squared MMD by a certain fraction, so F is obtained by stacking an enough number of these residual blocks. To begin with, we make the follow definition.

Definition 2.4.1. *For distributions q, p, and a $\frac{1}{2}$-Lipschitz function f, we define the improvement of the squared MMD by $\mathbf{Id}+f$ as*

$$\Delta(q,p;f) = \mathrm{MMD}(q,p)^2 - \mathrm{MMD}((\mathbf{Id}+f)\#q,p)^2. \tag{2.5}$$

Then, if $\Delta(q,p;f) > 0$, the residual block $\mathbf{Id}+f$ is helpful for reducing the squared MMD. It is straightforward to see that $\sup\{\Delta(q,p;f)\colon f \text{ is } \frac{1}{2}\text{-Lipschitz}\} \geq 0$. In order to construct an f that has a large $\Delta(q,p;f)$, we choose $f = \hat{f}_\varepsilon$ defined below.

Definition 2.4.2. *Define $\psi(p,q) = (\mathbb{E}_{x\sim p} - \mathbb{E}_{x\sim q})\phi(x)$, $g(z) = \psi(p,q)^\top\phi(z)$, and $\hat{f}_\varepsilon = \varepsilon\cdot\nabla g$, where $\varepsilon > 0$. Then, $\mathrm{MMD}(q,p) = \|\psi(p,q)\|$. In addition, $\hat{f}_\varepsilon(z) = \varepsilon J_\phi(z)\psi(p,q)$, where J_ϕ is the Jacobian matrix of ϕ.*

Then, according to (2.2) and (2.5),

$$\begin{aligned}\Delta(q,p;\hat{f}_\varepsilon) = \quad & \mathbb{E}_{z\sim q,x\sim q}\phi(z)^\top\phi(x) \\ & -\mathbb{E}_{z\sim q,x\sim q}\phi(z+\hat{f}_\varepsilon(z))^\top\phi(x+\hat{f}_\varepsilon(x)) \\ & +2\cdot\mathbb{E}_{z\sim q,x\sim p}\phi(z+\hat{f}_\varepsilon(z))^\top\phi(x) \\ & -2\cdot\mathbb{E}_{z\sim q,x\sim p}\phi(z)^\top\phi(x).\end{aligned} \tag{2.6}$$

Note that $\Delta(q,p;\hat{f}_\varepsilon)$ is a function of ε. We then analyze the first-order Taylor expansion of $\Delta(q,p;\hat{f}_\varepsilon)$ at $\varepsilon = 0^+$, denoted as $\Delta_1(q,p;\hat{f}_\varepsilon)$. Then, $\Delta(q,p;\hat{f}_\varepsilon) = \Delta_1(q,p;\hat{f}_\varepsilon) + \mathscr{O}\left(\varepsilon^2\right)$. With some arithmetic, we have

$$\Delta_1(q,p;\hat{f}_\varepsilon) = 2\psi(p,q)^\top \mathbb{E}_{z\sim q}\phi(z+\hat{f}_\varepsilon(z)). \tag{2.7}$$

We have the following bound on $\Delta_1(q,p;\hat{f}_\varepsilon)$.

Lemma 2.4.3. *If $d_\phi < \infty$, and the minimum singular value $\sigma_{\min}(J_\phi(z)) \geq \sqrt{b} > 0$ holds for any $z \in \mathbb{R}^d$, then*

$$\Delta_1(q,p;\hat{f}_\varepsilon) \geq 2\varepsilon b \cdot \mathrm{MMD}(q,p)^2. \tag{2.8}$$

Since $\Delta(q,p;\hat{f}_\varepsilon) = \Delta_1(q,p;\hat{f}_\varepsilon) + \mathscr{O}\left(\varepsilon^2\right)$, when ε is small, the residual block $\mathbf{Id} + \hat{f}_\varepsilon$ can indeed reduce the squared MMD by a certain fraction ($\geq 2\varepsilon b$). Next, as we require $f = \hat{f}_\varepsilon$ to be $\frac{1}{2}$-Lipschitz, we show under certain conditions the Lipschitz constant of $\hat{f}_\varepsilon$ is $\mathscr{O}(\varepsilon)$ in the following lemma.

Lemma 2.4.4. *If for any $z \in \mathbb{R}^d$, the Lipschitz constant of each element in $J_\phi(z)$ is no more than a universal constant L_{Jac}, then*

$$\mathrm{Lip}(\hat{f}_\varepsilon) \leq \sqrt{d \cdot d_\phi} L_{\mathrm{Jac}} \mathrm{MMD}(q,p) \cdot \varepsilon. \tag{2.9}$$

With these tools, we can construct an $F \in \mathscr{F}_N$ that achieves (2.4) in the following theorem.

Theorem 2.4.5. *Under the conditions of* **Lemma** *2.4.3 and* **Lemma** *2.4.4, there exists an $F \in \mathscr{F}_N$ with $N = \Theta\left(\frac{1}{\delta}\left(\log\frac{1}{\delta}\right)^2\right)$ that achieves* (2.4).

The proof is deferred to Section B.3. The main idea in the proof is to construct each f_i iteratively based on f_1 through f_{i-1}, so that adding this residual block can reduce the squared MMD by a certain fraction as indicated in **Lemma** 2.4.3. The bound is obtained by carefully balancing ε, δ, and N.

2.5 A Tighter Bound with Second-Order Analysis

In this section, we show under a few additional assumptions, there exists a much smaller $N = \mathscr{O}\left(\log \frac{1}{\delta}\right)$ and $F \in \mathscr{F}_N$ such that F achieves (2.4). The idea is to bound the second-order remainder of the Taylor expansion of $\Delta(q,p;\hat{f}_\varepsilon)$: $\Delta_2(q,p;\hat{f}_\varepsilon) = \Delta(q,p;\hat{f}_\varepsilon) - \Delta_1(q,p;\hat{f}_\varepsilon) = \mathscr{O}\left(\varepsilon^2\right)$. Once $\Delta_2(q,p;\hat{f}_\varepsilon)$ is explicitly bounded we can pick a small constant ε for every residual block [3] so $\Delta(q,p;\hat{f}_\varepsilon)$ is lower bounded by a universal constant times $\mathrm{MMD}(q,p)^2$. This then yields the $\mathscr{O}\left(\log \frac{1}{\delta}\right)$ bound for N. Now, we provide an explicit bound on $\Delta_2(q,p;\hat{f}_\varepsilon)$ in the following lemma.

Lemma 2.5.1. *Let B, C, L_{feat} be positive constants. If for any $z \in \mathbb{R}^d$, the maximum singular value $\sigma_{\max}(J_\phi(z)) \le \sqrt{B}$, the absolute value of any eigenvalue $|\lambda(\nabla^2 \phi_i(z))| \le C$ for any $1 \le i \le d_\phi$, and ϕ is L_{feat}-Lipschitz, then*

$$|\Delta_2(q,p;\hat{f}_\varepsilon)| \le \varepsilon^2 \cdot \mathrm{MMD}(q,p)^2 \cdot B \cdot \left(B + \|\psi(p,q)\| \sqrt{d_\phi} C (1 + \varepsilon L_{\text{feat}} \sqrt{B})\right). \tag{2.10}$$

Given this explicit bound on $\Delta_2(q,p;\hat{f}_\varepsilon)$, we can pick a small ε such that $|\Delta_2(q,p;\hat{f}_\varepsilon)| \le \frac{1}{2}\Delta_1(q,p;\hat{f}_\varepsilon)$ so that $\Delta(q,p;\hat{f}_\varepsilon) \ge \frac{1}{2}\Delta_1(q,p;\hat{f}_\varepsilon)$. Once this lower bound on $\Delta(q,p;\hat{f}_\varepsilon)$ is achieved, the squared MMD is multiplied by at most a universal constant less than 1 when the new residual block $\mathbf{Id} + \hat{f}_\varepsilon$ is added. We formalize the result in the following theorem.

Theorem 2.5.2. *Under the conditions of* **Lemma** *2.4.3,* **Lemma** *2.4.4, and* **Lemma** *2.5.1, there exists an $F \in \mathscr{F}_N$ with $N = \Theta\left(\log \frac{1}{\delta}\right)$ that achieves* (2.4).

The proof is deferred to Section B.5. The main idea in the proof is to construct each f_i in a similar way as in **Theorem** 2.4.5, but ε is selected as a universal constant according to **Lemma** 2.5.1.

[3] In **Theorem** 2.4.5, the ε for each residual block is related to δ in order to eliminate the effect by the unknown second-order terms. Here ε is independent with δ.

2.6 Conclusions

Normalizing flows are a class of flexible deep generative models that offers easy likelihood computation. Despite their empirical success, there is little theoretical understanding on whether they are universal approximators in transforming between probability distributions. In this work, we prove residual flows are indeed universal approximators in maximum mean discrepancy. Upper bounds on the number of residual blocks to achieve approximation are provided. Under mild conditions, we show $\Theta\left(\frac{1}{\delta}\left(\log\frac{1}{\delta}\right)^2\right)$ residual blocks can achieve (2.4) (see **Theorem** 2.4.5). Under more conditions, we show as few as $\Theta\left(\log\frac{1}{\delta}\right)$ residual blocks can achieve (2.4) (see **Theorem** 2.5.2).

There are a large number of open problems. One extension is to build universal approximation theory for residual flows in more general probability metrics such as the integral probability metrics [Müller, 1997] and the f-divergences [Csiszár and Shields, 2004]. Another direction is to extend the proposed universal approximation theory to other classes of normalizing flows such as autoregressive flows. A final open problem is to look at normalizing flows in real-world applications, and analyze their expressive power under practical assumptions.

2.7 Acknowledgements

We thank Songbai Yan for helpful feedback.

This chapter is a reformatted version of the paper ”Universal Approximation of Residual Flows in Maximum Mean Discrepancy” published in *International Conference on Machine Learning INNF+ Workshop, 2021,* by Zhifeng Kong and Kamalika Chaudhuri [Kong and Chaudhuri, 2021b]. The dissertation author was the primary researcher and author of this paper.

Chapter 3

Understanding Instance-based Interpretability of Variational Auto-Encoders

3.1 Introduction

Instance-based interpretation methods have been popular for supervised learning as they help explain why a model makes a certain prediction and hence have many applications [Barshan et al., 2020, Basu et al., 2020, Chen et al., 2020, Ghorbani and Zou, 2019, Hara et al., 2019, Harutyunyan et al., 2021, Koh and Liang, 2017, Koh et al., 2019, Pruthi et al., 2020, Yeh et al., 2018, Yoon et al., 2020]. For a classifier and a test sample z, an instance-based interpretation ranks all training samples x according to an interpretability score between x and z. Samples with high (low) scores are considered positively (negatively) important for the prediction of z.

However, in the literature of unsupervised learning especially generative models, instance-based interpretations are much less understood. In this work, we investigate instance-based interpretation methods for unsupervised learning based on influence functions [Cook and Weisberg, 1980, Koh and Liang, 2017]. In particular, we theoretically analyze certain classical non-parametric and parametric methods. Then, we look at a canonical deep generative model, variational auto-encoders (VAE, [Higgins et al., 2016, Kingma and Welling, 2013]), and explore some of the applications.

The first challenge is framing the counter-factual question for unsupervised learning. For instance-based interpretability in supervised learning, the counter-factual question is "which training samples are most responsible for the prediction of a test sample?" – which heavily relies on the label information. However, there is no label in unsupervised learning. In this work, we frame the counter-factual question for unsupervised learning as "which training samples are most responsible for increasing the likelihood (or reducing the loss when likelihood is not available) of a test sample?" We show that influence functions can answer this counter-factual question. Then, we examine influence functions for several classical unsupervised learning methods. We present theory and intuitions on how influence functions relate to likelihood and pairwise distances.

The second challenge is how to compute influence functions in VAE. The first difficulty here is that the VAE loss of a test sample involves an expectation over the encoder, so the actual influence function cannot be precisely computed. To deal with this problem, we use the empirical average of influence functions, and prove a concentration bound of the empirical average under mild conditions. Another difficulty is computation. The influence function involves inverting the Hessian of the loss with respect to all parameters, which involves massive computation for big neural networks with millions of parameters. To deal with this problem, we adapt a first-order estimate of the influence function called TracIn [Pruthi et al., 2020] to VAE. We call our method VAE-TracIn. It is fast because (*i*) it does not involve the Hessian, and (*ii*) it can accelerate computation with only a few checkpoints.

We begin with a sanity check that examines whether training samples have the highest influences over themselves, and show VAE-TracIn passes it. We then evaluate VAE-TracIn on several real world datasets. We find high (low) self influence training samples have large (small) losses. Intuitively, high self influence samples are hard to recognize or visually high-contrast, while low self influence ones share similar shapes or background. These findings lead to an application to unsupervised data cleaning, as high self influence samples are likely to be outside the data manifold. We then provide quantitative and visual analysis on influences over test data. We call high and low influence samples *proponents* and *opponents*, respectively. [1] We find in

[1]There are different names in the literature, such as helpful/harmful samples [Koh and Liang, 2017], excitatory/inhibitory points [Yeh et al., 2018], and proponents/opponents [Pruthi et al., 2020].

certain cases both proponents and opponents are similar samples from the same class, while in other cases proponents have large norms.

We consider VAE-TracIn as a general-purpose tool that can potentially help understand many aspects in the unsupervised setting, including (*i*) detecting underlying memorization, bias or bugs [Feldman and Zhang, 2020] in unsupervised learning, (*ii*) performing data deletion [Asokan and Seelamantula, 2020, Izzo et al., 2021] in generative models, and (*iii*) examining training data without label information.

We make the following contributions in this chapter.

- We formally frame instance-based interpretations for unsupervised learning.
- We examine influence functions for several classical unsupervised learning methods.
- We present VAE-TracIn, an instance-based interpretation method for VAE. We provide both theoretical and empirical justification to VAE-TracIn.
- We evaluate VAE-TracIn on several real world datasets. We provide extensive quantitative analysis and visualization, as well as an application to unsupervised data cleaning.

3.1.1 Related Work

There are two lines of research on instance-based interpretation methods for supervised learning.

The first line of research frames the following counter-factual question: which training samples are most responsible for the prediction of a particular test sample z? This is answered by designing an interpretability score that measures the importance of training samples over z and selecting those with the highest scores. Many scores and their approximations have been proposed [Barshan et al., 2020, Basu et al., 2020, Chen et al., 2020, Hara et al., 2019, Koh and Liang, 2017, Koh et al., 2019, Pruthi et al., 2020, Yeh et al., 2018]. Specifically, Koh and Liang [2017] introduce the influence function (IF) based on the terminology in robust statistics [Cook and Weisberg, 1980]. The intuition is removing an important training sample of z should result in a huge increase of its test loss. Because the IF is hard to compute, Pruthi et al. [2020] propose TracIn, a fast first-order approximation to IF.

Our work extends the counter-factual question to unsupervised learning where there is no label. We ask: which training samples are most responsible for increasing the likelihood (or reducing the loss) of a test sample? In this chapter, we propose VAE-TracIn, an instance-based interpretation method for VAE [Higgins et al., 2016, Kingma and Welling, 2013] based on TracIn and IF.

The second line of research considers a different counter-factual question: which training samples are most responsible for the overall performance of the model (e.g. accuracy)? This is answered by designing an interpretability score for each training sample. Again many scores have been proposed [Ghorbani and Zou, 2019, Harutyunyan et al., 2021, Yoon et al., 2020]. Terashita et al. [2021] extend this framework to a specific unsupervised model called generative adversarial networks [Goodfellow et al., 2014]. They measure influences of samples on several evaluation metrics, and discard samples that harm these metrics. Our work is orthogonal to these works.

The instance-based interpretation methods lead to many applications in various areas including adversarial learning, data cleaning, prototype selection, data summarization, and outlier detection [Barshan et al., 2020, Basu et al., 2020, Chen et al., 2020, Feldman and Zhang, 2020, Ghorbani and Zou, 2019, Hara et al., 2019, Harutyunyan et al., 2021, Khanna et al., 2019, Koh and Liang, 2017, Pruthi et al., 2020, Suzuki et al., 2021, Ting and Brochu, 2018, Ye et al., 2021, Yeh et al., 2018, Yoon et al., 2020]. In this chapter, we apply the proposed VAE-TracIn to an unsupervised data cleaning task.

Prior works on interpreting generative models analyze their latent space via measuring disentanglement, explaining and visualizing representations, or analysis in an interactive interface [Alvarez-Melis and Jaakkola, 2018, Bengio et al., 2013, Chen et al., 2016, Desjardins et al., 2012, Kim and Mnih, 2018, Olah et al., 2017, 2018, Ross et al., 2021]. These latent space analysis are complementary to the instance-based interpretation methods in this chapter.

3.2 Instance-based Interpretations

Let $X = \{x_i\}_{i=1}^N \in \mathbb{R}^d$ be the training set. Let $\mathscr{A}$ be an algorithm that takes X as input and outputs a model that describes the distribution of X. $\mathscr{A}$ can be a density estimator or a generative model. Let $L(X;\mathscr{A}) = \frac{1}{N}\sum_{i=1}^N \ell(x_i;\mathscr{A}(X))$ be a loss function. Then, the influence function of a training data x_i over a test data $z \in \mathbb{R}^d$ is the loss of z computed from the model trained without x_i minus that computed from the model trained with x_i. If the difference is large, then x_i should be very influential for z. Formally, the influence function is defined below.

Definition 3.2.1 (Influence functions [Koh and Liang, 2017])**.** *Let $X_{-i} = X \setminus \{x_i\}$. Then, the influence of x_i over z is defined as $\mathrm{IF}_{X,\mathscr{A}}(x_i,z) = \ell(z;\mathscr{A}(X_{-i})) - \ell(z;\mathscr{A}(X))$. If $\mathrm{IF}_{X,\mathscr{A}}(x_i,z) > 0$, we say x_i is a proponent of z; otherwise, we say x_i is an opponent of z.*

For big models $\mathscr{A}$ such as deep neural networks, doing retraining and obtaining $\mathscr{A}(X_{-i})$ can be expensive. The following TracIn score is a fast approximation to IF.

Definition 3.2.2 (TracIn scores [Pruthi et al., 2020])**.** *Suppose $\mathscr{A}(X)$ is obtained by minimizing $L(X;\mathscr{A})$ via stochastic gradient descent. Let $\{\theta_{[c]}\}_{c=1}^C$ be C checkpoints during the training procedure. Then, the estimated influence of x_i over z is defined as $\mathrm{TracIn}_{X,\mathscr{A}}(x_i,z) = \sum_{c=1}^C \nabla\ell(x_i;\theta_{[c]})^\top \nabla\ell(z;\theta_{[c]})$.*

We are also interested in the influence of a training sample over itself. Formally, we define this quantity as the self influence of x, or $\text{Self-IF}_{X,\mathscr{A}}(x) = \mathrm{IF}_{X,\mathscr{A}}(x,x)$. In supervised learning, self influences provide rich information about memorization properties of training samples. Intuitively, high self influence samples are atypical, ambiguous or mislabeled, while low self influence samples are typical [Feldman and Zhang, 2020].

3.3 Influence Functions for Classical Unsupervised Learning

In this section, we analyze influence functions for unsupervised learning. The goal is to provide intuition on what influence functions should tell us in the unsupervised setting. Specifically, we look at three classical methods: the non-parametric k-nearest-neighbor (k-NN) density estimator, the non-parametric kernel density estimator (KDE), and the parametric Gaussian mixture models (GMM). We let the loss function ℓ to be the negative log-likelihood: $\ell(z) = -\log p(z)$.

The k-Nearest-Neighbor (k-NN) density estimator. The k-NN density estimator is defined as $p_{k\text{NN}}(x;X) = k/(NV_dR_k(x;X)^d)$, where $R_k(x;X)$ is the distance between x and its k-th nearest neighbor in X and V_d is the volume of the unit ball in $\mathbb{R}^d$. Then, we have the following influence function for the k-NN density estimator:

$$\mathrm{IF}_{X,k\mathrm{NN}}(x_i,z) = \log\frac{N-1}{N} + \begin{cases} d\log\frac{R_{k+1}(z;X)}{R_k(z;X)} & \|x_i - z\| \le R_k(z;X) \\ 0 & \text{otherwise} \end{cases}. \tag{3.1}$$

See Appendix C.1.1 for proof. Note, when z is fixed, there are only two possible values for training data influences: $\log\frac{N-1}{N}$ and $\log\frac{N-1}{N} + d\log\frac{R_{k+1}(z;X)}{R_k(z;X)}$. As for Self-$\mathrm{IF}_{X,k\mathrm{NN}}(x_i)$, samples with the largest self influences are those with the largest $\frac{R_{k+1}(x_i;X)}{R_k(x_i;X)}$. Intuitively, these samples belong to a cluster of size exactly k, and the cluster is far away from other samples.

Kernel Density Estimators (KDE). The KDE is defined as $p_{\mathrm{KDE}}(x;X) = \frac{1}{N}\sum_{i=1}^N K_\sigma(x - x_i)$, where K_σ is the Gaussian $\mathcal{N}(0,\sigma^2 I)$. Then, we have the following influence function for KDE:

$$\mathrm{IF}_{X,\mathrm{KDE}}(x_i,z) = \log\frac{N-1}{N} + \log\left(1 + \frac{\frac{1}{N}K_\sigma(z-x_i)}{p_{\mathrm{KDE}}(z;X) - \frac{1}{N}K_\sigma(z-x_i)}\right). \tag{3.2}$$

See Appendix C.1.1 for proof. For a fixed z, an x_i with larger $\|z - x_i\|$ has a higher influence over z. Therefore, the strongest proponents of z are those closest to z in the ℓ_2 distance, and the strongest opponents are the farthest. As for Self-IF$_{X,\text{KDE}}(x_i)$, samples with the largest self influences are those with the least likelihood $p_{\text{KDE}}(x_i;X)$. Intuitively, these samples locate at very sparse regions and have few nearby samples. On the other hand, samples with the largest likelihood $p_{\text{KDE}}(x_i;X)$, or those in the high density area, have the least self influences.

Gaussian Mixture Models (GMM). As there is no closed-form expression for general GMM, we make the following well-separation assumption to simplify the problem.

Assumption 3.3.1. *$X = \bigcup_{k=0}^{K-1} X_k$ where each X_k is a cluster. We assume these clusters are well-separated:* $\min\{\|x-x'\| : x \in X_k, x' \in X_{k'}\} \gg \max\{\|x-x'\| : x,x' \in X_k\}$.

Let $|X_k| = N_k$ and $N = \sum_{k=0}^{K-1} N_k$. For $x \in \mathbb{R}^d$, let $k = \arg\min_i d(x, X_i)$. Then, we define the well-separated spherical GMM (WS-GMM) of K mixtures as $p_{\text{WS-GMM}}(x) = \frac{N_k}{N}\mathcal{N}(x;\mu_k,\sigma_k^2 I)$, where the parameters are given by the maximum likelihood estimates

$$\mu_k = \frac{1}{N_k}\sum_{x\in X_k} x,\ \sigma_k^2 = \frac{1}{N_k d}\sum_{x\in X_k}\|x-\mu_k\|^2 = \frac{1}{N_k d}\sum_{x\in X_k} x^\top x - \frac{1}{d}\mu_k^\top \mu_k. \tag{3.3}$$

For conciseness, we let test sample z from cluster zero: $z \in \text{conv}(X_0)$. Then, we have the following influence function for WS-GMM. If $x_i \notin X_0$, IF$_{X,\text{WS-GMM}}(x_i,z) = -\frac{1}{N} + \mathcal{O}\left(N^{-2}\right)$. Otherwise,

$$\text{IF}_{X,\text{WS-GMM}}(x_i,z) = \frac{d+2}{2N_0} + \frac{1}{2N_0\sigma_0^2}\left(\frac{\|z-\mu_0\|^2}{\sigma_0^2} - \|z-x_i\|^2\right) - \frac{1}{N} + \mathcal{O}\left(N_0^{-2}\right). \tag{3.4}$$

See Appendix C.1.1 for proof. A surprising finding is that some $x_i \in X_0$ may have very negative influences over z (i.e. strong opponents of z are from the same class). This happens with high probability if $\|z-x_i\|^2 \gtrapprox (1+\sigma_0^2)d + 2\sigma_0^2$ for large dimension d. Next, we compute the self influence of an $x_i \in X_k$. According to (3.4),

$$\text{Self-IF}_{X,\text{WS-GMM}}(x_i) = \frac{d+2}{2N_k} + \frac{\|x_i-\mu_k\|^2}{2N_k\sigma_k^4} - \frac{1}{N} + \mathcal{O}\left(N_k^{-2}\right). \tag{3.5}$$

Within each cluster X_k, samples far away to the cluster center μ_k have large self influences and vice versa. Across the entire dataset, samples in cluster X_k whose N_k or σ_k is small tend to have large self influences, which is very different from k-NN or KDE.

3.3.1 Summary

We summarize the intuitions of influence functions in classical unsupervised learning in Table 3.4. Among these methods, the strong proponents are all nearest samples, but self influences and strong opponents are quite different. We then visualize an example of six clusters of 2D points in Fig. C.1 in Appendix C.2.1. In Fig. C.2, We plot the self influences of these data points under different density estimators. For a test data point z, we plot influences of all data points over z in Fig. C.3.

3.4 Instance-based Interpretations for Variational Auto-encoders

In this section, we show how to compute influence functions for a class of deep generative models called variational auto-encoders (VAE). Specifically, we look at β-VAE [Higgins et al., 2016] defined below, which generalizes the original VAE by Kingma and Welling [2013].

Definition 3.4.1 (β-VAE [Higgins et al., 2016])**.** *Let d_{latent} be the latent dimension. Let $P_\phi : \mathbb{R}^{d_{\text{latent}}} \to \mathbb{R}^+$ be the decoder and $Q_\psi : \mathbb{R}^d \to \mathbb{R}^+$ be the encoder, where ϕ and ψ are the parameters of the networks. Let $\theta = [\phi, \psi]$. Let the latent distribution P_{latent} be $\mathcal{N}(0, I)$. For $\beta > 0$, the β-VAE model minimizes the following loss:*

$$L_\beta(X;\theta) = \mathbb{E}_{x\sim X}\ell_\beta(x;\theta) = \beta \cdot \mathbb{E}_{x\sim X}\mathrm{KL}\left(Q_\psi(\cdot|x)\|P_{\text{latent}}\right) - \mathbb{E}_{x\sim X}\mathbb{E}_{\xi\sim Q_\psi(\cdot|x)}\log P_\phi(x|\xi). \quad (3.6)$$

In practice, the encoder $Q = Q_\psi$ outputs two vectors, μ_Q and σ_Q, so that $Q(\cdot|x) = \mathcal{N}(\mu_Q(x), \mathrm{diag}(\sigma_Q(x))^2 I)$. The decoder $P = P_\phi$ outputs a vector μ_P so that $\log P(x|\xi)$ is a constant times $\|\mu_P(\xi) - x\|^2$ plus a constant.

Let $\mathscr{A}$ be the β-VAE that returns $\mathscr{A}(X) = \arg\min_\theta L_\beta(X;\theta)$. Let $\theta^* = \mathscr{A}(X)$ and $\theta^*_{-i} = \mathscr{A}(X_{-i})$. Then, the influence function of x_i over a test point z is $\ell_\beta(z;\theta^*_{-i}) - \ell_\beta(z;\theta^*)$, which equals to

$$\begin{aligned}\mathrm{IF}_{X,\mathrm{VAE}}(x_i,z) &= \beta\left(\mathrm{KL}\left(Q_{\psi^*_{-i}}(\cdot|z)\|P_{\mathrm{latent}}\right) - \mathrm{KL}\left(Q_{\psi^*}(\cdot|z)\|P_{\mathrm{latent}}\right)\right) \\ &\quad - \left(\mathbb{E}_{\xi\sim Q_{\psi^*_{-i}}(\cdot|z)}\log P_{\phi^*_{-i}}(z|\xi) - \mathbb{E}_{\xi\sim Q_{\psi^*}(\cdot|z)}\log P_{\phi^*}(z|\xi)\right).\end{aligned} \tag{3.7}$$

Challenge. The first challenge is that IF in (3.7) involves an expectation over the encoder, so it cannot be precisely computed. To solve the problem, we compute the empirical average of the influence function over m samples. In **Theorem** 3.4.2, we theoretically prove that the empirical influence function is close to the actual influence function with high probability when m is properly selected. The second challenge is that IF is hard to compute. To solve this problem, in Section 3.4.1, we propose VAE-TracIn, a computationally efficient solution to VAE.

A probabilistic bound on influence estimates. Let $\hat{\mathrm{IF}}^{(m)}_{X,\mathrm{VAE}}$ be the empirical average of the influence function over m i.i.d. samples. We have the following result.

Theorem 3.4.2 (Error bounds on influence estimates (informal, see formal statement in **Theorem** C.1.4))**.** *Under mild conditions, for any small $\varepsilon > 0$ and $\delta > 0$, there exists an $m = \Theta\left(\frac{1}{\varepsilon^2\delta}\right)$ such that*

$$\mathrm{Prob}\left(\left|\mathrm{IF}_{X,\mathrm{VAE}}(x_i,z) - \hat{\mathrm{IF}}^{(m)}_{X,\mathrm{VAE}}(x_i,z)\right| \geq \varepsilon\right) \leq \delta. \tag{3.8}$$

Formal statements and proofs are in Appendix C.1.2.

3.4.1 VAE-TracIn

In this section, we introduce VAE-TracIn, a computationally efficient interpretation method for VAE. VAE-TracIn is built based on TracIn (**Definition** 3.2.2). According to (3.6), the gradient of the loss $\ell_\beta(x;\theta)$ can be written as $\nabla_\theta \ell_\beta(x;\theta) = [\nabla_\phi \ell_\beta(x;\theta)^\top, \nabla_\psi \ell_\beta(x;\theta)^\top]^\top$, where

$$\begin{aligned}
\nabla_\phi \ell_\beta(x;\theta) &= \mathbb{E}_{\xi\sim Q_\psi(\cdot|x)}\left(-\nabla_\phi \log P_\phi(x|\xi)\right) =: \mathbb{E}_{\xi\sim Q_\psi(\cdot|x)} U(x,\xi,\phi,\psi), \text{ and} \\
\nabla_\psi \ell_\beta(x;\theta) &= \mathbb{E}_{\xi\sim Q_\psi(\cdot|x)} \nabla_\psi \log Q_\psi(\xi|x)\left(\beta \log \frac{Q_\psi(\xi|x)}{P_{\text{latent}}(\xi)} - \log P_\phi(x|\xi)\right) \\
&=: \mathbb{E}_{\xi\sim Q_\psi(\cdot|x)} V(x,\xi,\phi,\psi).
\end{aligned} \tag{3.9}$$

The derivations are based on the Stochastic Gradient Variational Bayes estimator [Kingma and Welling, 2013], which offers low variance [Rezende et al., 2014]. See Appendix C.1.3 for full details of the derivation. We estimate the expectation $\mathbb{E}_\xi$ by averaging over m i.i.d. samples. Then, for a training data x and test data z, the VAE-TracIn score of x over z is computed as

$$\begin{aligned}
\text{VAE-TracIn}(x,z) &= \sum_{c=1}^{C}\left(\frac{1}{m}\sum_{j=1}^{m} U(x,\xi_{j,[c]},\phi_{[c]},\psi_{[c]})\right)^\top \left(\frac{1}{m}\sum_{j=1}^{m} U(z,\xi'_{j,[c]},\phi_{[c]},\psi_{[c]})\right) \\
&+ \sum_{c=1}^{C}\left(\frac{1}{m}\sum_{j=1}^{m} V(x,\xi_{j,[c]},\phi_{[c]},\psi_{[c]})\right)^\top \left(\frac{1}{m}\sum_{j=1}^{m} V(z,\xi'_{j,[c]},\phi_{[c]},\psi_{[c]})\right),
\end{aligned} \tag{3.10}$$

where the notations U,V are from (3.9), $\theta_{[c]} = [\phi_{[c]},\psi_{[c]}]$ is the c-th checkpoint, $\{\xi_{j,[c]}\}_{j=1}^m$ are i.i.d. samples from $Q_{\psi_{[c]}}(\cdot|x)$, and $\{\xi'_{j,[c]}\}_{j=1}^m$ are i.i.d. samples from $Q_{\psi_{[c]}}(\cdot|z)$.

Connections between VAE-TracIn and influence functions [Koh and Liang, 2017].

Koh and Liang [2017] use the second-order (Hessian-based) approximation to the change of loss under the assumption that the loss function is convex. The TracIn algorithm [Pruthi et al., 2020] uses the first-order (gradient-based) approximation to the change of loss during the training process under the assumption that (stochastic) gradient descent is the optimizer. We expect these methods to give similar results in the ideal situation. However, we implemented the method by Koh and Liang [2017] and found it to be inaccurate for VAE. A possible reason is that the Hessian vector product used to approximate the second order term is unstable.

Complexity of VAE-TracIn.

The run-time complexity of VAE-TracIn is linear in the number of samples (N), checkpoints (C), and network parameters ($|\theta|$).

3.5 Experiments

In this section, we aim to answer the following questions.

- Does VAE-TracIn pass a sanity check for instance-based interpretations?
- Which training samples have the highest and lowest self influences, respectively?
- Which training samples have the highest influences over (i.e. are strong proponents of) a test sample? Which have the lowest influences over it (i.e. are its strong opponents)?

These questions are examined in experiments on the MNIST [LeCun et al., 2010] and CIFAR-10 [Krizhevsky and Hinton, 2009] datasets.

3.5.1 Sanity Check

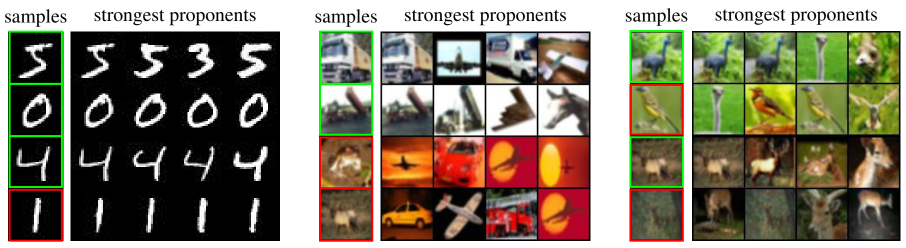
Question. Does VAE-TracIn find the most influential training samples? In a perfect instance-based interpretation for a good model, training samples should have large influences over themselves. As a sanity check, we examine if training samples are the strongest proponents over themselves. This is analogous to the identical subclass test by Hanawa et al. [2020].

Methodology. We train separate VAE models on MNIST, CIFAR, and each CIFAR subclass (the set of five thousand CIFAR samples in each class). For each model, we examine the frequency that a training sample is the most influential one among all samples over itself. Due to computational limits we examine the first 128 samples. The results for MNIST, CIFAR, and the averaged result for CIFAR subclasses are reported in Table 3.1. Detailed results for CIFAR subclasses are in Appendix C.2.3.

Results. The results indicate that VAE-TracIn can find the most influential training samples in MNIST and CIFAR subclasses. This is achieved even under the challenge that many training samples are very similar to each other. The results for CIFAR is possibly due

Table 3.1. Sanity check on the frequency that a training sample is the most influential one over itself. Results on MNIST, CIFAR, and the averaged result on CIFAR subclasses are reported.

MNIST		CIFAR		Averaged CIFAR subclass
$d_{\text{latent}} = 64$	$d_{\text{latent}} = 128$	$d_{\text{latent}} = 64$	$d_{\text{latent}} = 128$	$d_{\text{latent}} = 64$
0.992	1.000	0.609	0.602	0.998



(a) MNIST **(b)** CIFAR **(c)** CIFAR subclass

Figure 3.1. Certain training samples and their strongest proponents in the training set (sorted from left to right). A sample x_i is marked in green box if it is more influential than other samples over itself (i.e. it is the strongest proponent of itself) and otherwise in red box.

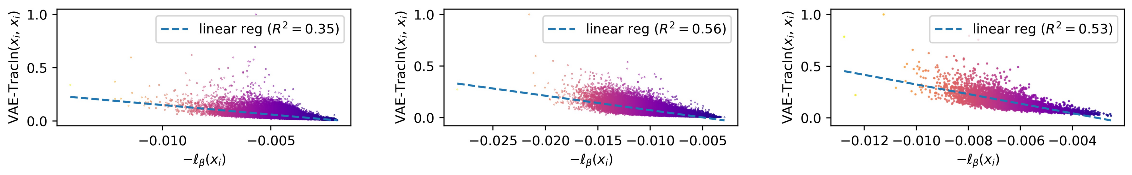
to underfitting as it is challenging to train a good VAE on this dataset. Note, the same VAE architecture is trained on CIFAR subclasses.

Visualization. We visualize some correctly and incorrectly identified strongest proponents in Fig. 3.1. On MNIST or CIFAR subclasses, even if a training sample is not exactly the strongest proponent of itself, it still ranks very high in the order of influences.

3.5.2 Self Influences

Question. Which training samples have the highest and lowest self influences, respectively? Self influences provide rich information about properties of training samples such as memorization. In supervised learning, high self influence samples can be atypical, ambiguous or mislabeled, while low self influence samples are typical [Feldman and Zhang, 2020]. We examine what self influences reveal in VAE.

Methodology. We train separate VAE models on MNIST, CIFAR, and each CIFAR subclass. We then compute the self influences and losses of each training sample. We show



(a) MNIST **(b)** CIFAR **(c)** CIFAR-Airplane

Figure 3.2. Scatter plots of self influences versus negative losses of all training samples in several datasets. The linear regressors show that high self influence samples have large losses.

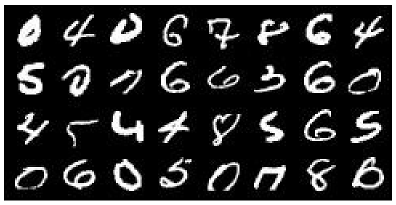
the scatter plots of self influences versus negative losses in Fig. 3.2. [2] We fit linear regression models to these points and report the R^2 scores of the regressors. More comparisons including the marginal distributions and the joint distributions can be found in Appendix C.2.4 and Appendix C.2.5.

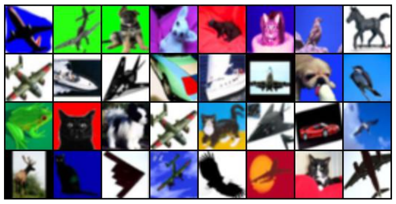
Results. We find the self influence of a training sample x_i tends to be large when its loss $\ell_\beta(x_i)$ is large. This finding in VAE is consistent with KDE and GMM (see Fig. C.2). In supervised learning, Pruthi et al. [2020] find high self influence samples come from densely populated areas while low self influence samples come from sparsely populated areas. Our findings indicate significant difference between supervised and unsupervised learning in terms of self influences under certain scenarios.

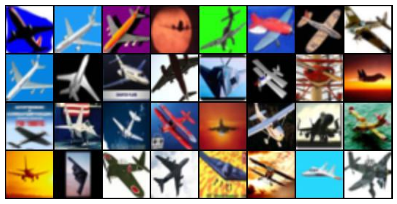
Visualization. We visualize high and low self influence samples in Fig. 3.3 (more visualizations in Appendix C.2.5). High self influence samples are either hard to recognize or visually high-contrast, while low self influence samples share similar shapes or background. These visualizations are consistent with the memorization analysis by Feldman and Zhang [2020] in the supervised setting. We also notice that there is a concurrent work connecting self influences on log-likelihood and memorization properties in VAE through cross validation and retraining [van den Burg and Williams, 2021]. Our quantitative and qualitative results are consistent with their results.

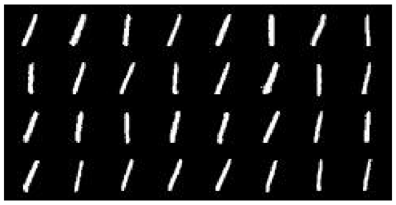
Application to unsupervised data cleaning. A potential application to unsupervised data cleaning is to use self influences to detect unlikely samples and let a human expert decide

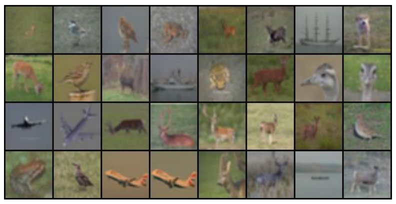
[2] We use the negative loss because it relates to the log-likelihood of x_i: when $\beta = 1$, $-\ell_\beta(x) \leq \log p(x)$.


(a) MNIST (highest self-inf)


(b) CIFAR (highest self-inf)


(c) CIFAR-Airplane (highest self-inf)


(d) MNIST (lowest self-inf)


(e) CIFAR (lowest self-inf)

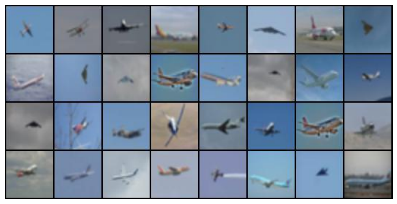

(f) CIFAR-Airplane (lowest self-inf)

Figure 3.3. High and low self influence samples from several datasets. High self influence samples are hard to recognize or high-contrast. Low self influence samples share similar shapes or background.

whether to discard them before training. The unlikely samples may include noisy samples, contaminated samples, or incorrectly collected samples due to bugs in the data collection process. For example, they could be unrecognizable handwritten digits in MNIST or objects in CIFAR. Similar approaches in supervised learning use self influences to detect mislabeled data [Koh and Liang, 2017, Pruthi et al., 2020, Yeh et al., 2018] or memorized samples [Feldman and Zhang, 2020]. We extend the application of self influences to scenarios where there are no labels.

To test this application, we design an experiment to see if self influences can find a small amount of extra samples added to the original dataset. The extra samples are from other datasets: 1000 EMNIST [Cohen et al., 2017] samples for MNIST, and 1000 CelebA [Liu et al., 2015] samples for CIFAR, respectively. In Fig. C.8, we plot the detection curves to show fraction of extra samples found when all samples are sorted in the self influence order. The area under these detection curves (AUC) are 0.887 in the MNIST experiment and 0.760 in the CIFAR experiment. [3] Full results and more comparisons can be found in Appendix C.2.6. The results indicate that extra samples generally have higher self influences than original samples, so it has much

[3]AUC ≈ 1 means the detection is near perfect, and AUC ≈ 0.5 means the detection is near random.

potential to apply VAE-TracIn to unsupervised data cleaning.

3.5.3 Influences over Test Data

Question. Which training samples are strong proponents or opponents of a test sample, respectively? Influences over a test sample z provide rich information about the relationship between training data and z. In supervised learning, strong proponents help the model correctly predict the label of z while strong opponents harm it. Empirically, strong proponents are visually similar samples from the same class, while strong opponents tend to confuse the model [Pruthi et al., 2020]. In unsupervised learning, we expect that strong proponents increase the likelihood of z and strong opponents reduce it. We examine which samples are strong proponents or opponents in VAE.

Methodology. We train separate VAE models on MNIST, CIFAR, and each CIFAR subclass. We then compute VAE-TracIn scores of all training samples over 128 test samples.

In MNIST experiments, we plot the distributions of influences according to whether training and test samples belong to the same class (See results on label zero in Fig. C.9). We then compare the influences of training over test samples to their distances in the latent space in Fig. C.10c. Quantitatively, we define samples that have the 0.1% highest/lowest influences as the strongest proponents/opponents. Then, we report the fraction of the strongest proponents/opponents that belong to the same class as the test sample and the statistics of pairwise distances in Table 3.2. Additional comparisons can be found in Appendix C.2.7,

In CIFAR and CIFAR subclass experiments, we compare influences of training over test samples to the norms of training samples in the latent space in Fig. C.11. Quantitatively, we report the statistics of the norms in Table 3.3. Additional comparisons can be found in Appendix C.2.8.

Results. In MNIST experiments, we find many strong proponents and opponents of a test sample are its similar samples from the same class. In terms of class information, many ($\sim 80\%$) strongest proponents and many ($\sim 40\%$) strongest opponents have the same label as test samples.

In terms of distances in the latent space, it is shown that the strongest proponents and opponents are close (thus similar) samples, while far away samples have small absolute influences. These findings are similar to GMM discussed in Section 3.3, where the strongest opponents may come from the same class (see Fig. C.3). The findings are also related to the supervised setting in the sense that dissimilar samples from a different class have small influences.

Results in CIFAR and CIFAR subclass experiments indicate strong proponents have large norms in the latent space. [4] This observation also happens to many instance-based interpretations in the supervised setting including classification methods [Hanawa et al., 2020] and logistic regression [Barshan et al., 2020], where large norm samples can impact a large region in the data space, so they are influential to many test samples.

Visualization. We visualize the strongest proponents and opponents in Fig. 3.4. More visualizations can be found in Appendix C.2.7 and Appendix C.2.8. In the MNIST experiment, the strongest proponents look very similar to test samples. The strongest opponents are often the same but visually different digits. For example, the opponents of the test ”two” have very different thickness and styles. In CIFAR and CIFAR subclass experiments, we find strong proponents seem to match the color of the images – including the background and the object – and they tend to have the same but brighter colors. Nevertheless, many proponents are from the same class. Strong opponents, on the other hand, tend to have very different colors as the test samples.

3.5.4 Discussion

VAE-TracIn provides rich information about instance-level interpretability in VAE. In terms of self influences, there is correlation between self influences and VAE losses. Visually, high self influence samples are ambiguous or high-contrast while low self influence samples are similar in shape or background. In terms of influences over test samples, for VAE trained on MNIST, many proponents and opponents are similar samples in the same class, and for

[4]Large norm samples can be outliers, high-contrast samples, or very bright samples.

Table 3.2. Statistics of influences, class information, and distances of train-test sample pairs in MNIST. "+" means top-0.1% strong proponents, "−" means top-0.1% strong opponents, and "all" means the train set.

d_{latent}	64	96	128
same class rate (+)	81.9%	84.0%	82.1%
same class rate (−)	37.3%	43.3%	40.3%
distances (+)	0.94 ± 0.53	0.94 ± 0.55	0.76 ± 0.51
distances (−)	1.78 ± 0.75	1.84 ± 0.78	1.29 ± 0.67
distances (all)	2.54 ± 0.90	2.57 ± 0.91	2.23 ± 0.92

Table 3.3. The means ± standard errors of latent space norms of training samples in CIFAR and CIFAR-Airplane. Notations follow Table 3.2. It is shown that strong proponents tend to have very large norms.

CIFAR	(+)	7.42 ± 1.10
	(−)	3.89 ± 1.26
	(all)	5.06 ± 1.18
CIFAR-Airplane	(+)	4.73 ± 0.78
	(−)	4.26 ± 0.91
	(all)	4.07 ± 0.83

Table 3.4. High level summary of influence functions in classical unsupervised learning methods (*k*-NN, KDE and GMM) and VAE.

Method	**high self influence samples**	**low self influence samples**
k-NN	in a cluster of size exactly *k*	–
KDE	in low density (sparse) region	in high density region
GMM	far away to cluster centers	near cluster centers
VAE	large loss complicated or high-contrast	small loss simple shapes, simple background
Method	**strong proponents**	**strong opponents**
k-NN	*k* nearest neighbours	other than *k* nearest neighbours
KDE	nearest neighbours	farthest samples
GMM	nearest neighbours	distant samples (same class)
$\text{VAE}_{(\text{MNIST})}$	nearest neighbors (same class)	distant samples (same class)
$\text{VAE}_{(\text{CIFAR})}$	large norms and similar colors	different colors

VAE trained on CIFAR, proponents have large norms in the latent space. We summarize these high level intuitions of influence functions in VAE in Table 3.4. We observe there are strong connections between these findings and influence functions in KDE, GMM, classification and

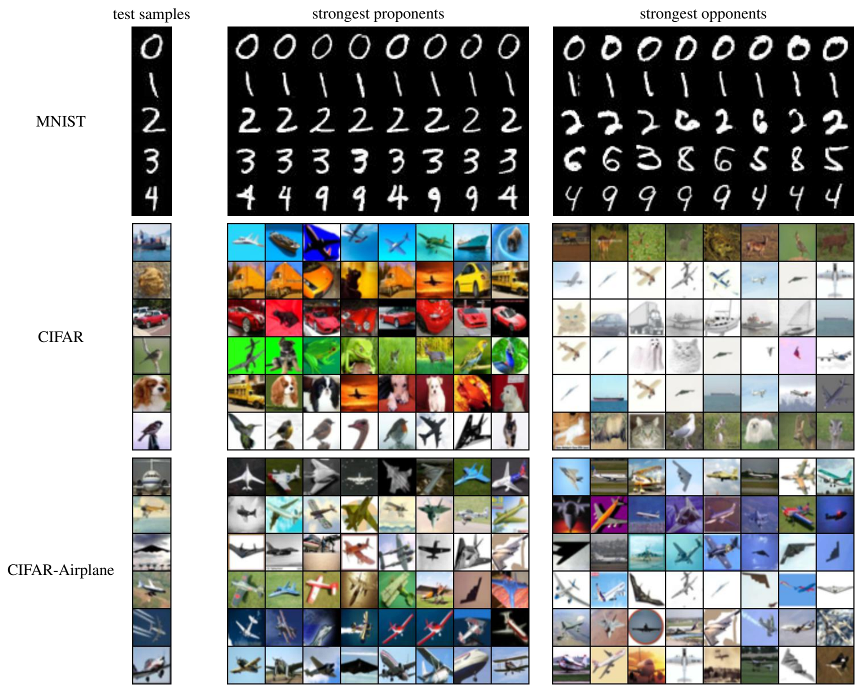


Figure 3.4. Test samples, their strongest proponents, and strongest opponents. In MNIST the strongest proponents are visually similar while the strongest opponents are often the same digit but are visually different. In CIFAR and CIFAR-Airplane the strongest proponents have similar but brighter colors.

simple regression models.

3.6 Conclusion

Influence functions in unsupervised learning can reveal the most responsible training samples that increase the likelihood (or reduce the loss) of a particular test sample. In this chapter, we investigate influence functions for several classical unsupervised learning methods and one deep generative model with extensive theoretical and empirical analysis. We present VAE-TracIn, a theoretical sound and computationally efficient algorithm that estimates influence functions for VAE, and evaluate it on real world datasets.

One limitation of our work is that it is still challenging to apply VAE-TracIn to modern, huge models trained on a large amount of data, which is an important future direction. There are several potential ways to scale up VAE-TracIn for large networks and datasets. First, we observe both positively and negatively influential samples (i.e. strong proponents and opponents) are similar to the test sample. Therefore, we could train an embedding space or a tree structure (such as the kd-tree) and then only compute VAE-TracIn values for similar samples. Second, because training at earlier epochs may be more effective than later epochs (as optimization is near convergence then), we could select a smaller but optimal subset of checkpoints to compute VAE-TracIn. Finally, we could use gradients of certain layers (e.g. the last fully-connected layer of the network as in Pruthi et al. [2020]).

Another important future direction is to investigate down-stream applications of VAE-TracIn such as detecting memorization or bias and performing data deletion or debugging.

3.7 Acknowledgements

We thank NSF under IIS 1719133 and CNS 1804829 for research support. We thank Casey Meehan, Yao-Yuan Yang, and Mary Anne Smart for helpful feedback.

This chapter is a reformatted version of the paper ”Understanding Instance-based Interpretability of Variational Auto-Encoders” published in *Advances in Neural Information Processing Systems, 2021,* by Zhifeng Kong and Kamalika Chaudhuri [Kong and Chaudhuri, 2021a]. The dissertation author was the primary researcher and author of this paper.

Chapter 4

Data Redaction from Pre-trained GANs

4.1 Introduction

Generative Adversarial Networks (GANs) are large neural generative models that learn a complicated probability distribution from data and then generate samples from it. These models have been immensely successful in many large scale tasks from multiple domains, such as images [Zhu et al., 2020, Karras et al., 2020, 2021], point clouds [Zhang et al., 2021], video [Tulyakov et al., 2018], text [de Masson d'Autume et al., 2019], and speech [Kong et al., 2020].

However, it is also well-known that many deep generative models frequently output undesirable samples, which makes them less reliable and trustworthy. Image models generate blurred samples [Kaneko and Harada, 2021] or checkerboard artifacts [Odena et al., 2016, Zhang et al., 2019b, Wang et al., 2020, Schwarz et al., 2021], speech models produce unnatural sound [Donahue et al., 2018, Thiem et al., 2020], and language models emit offensive text [Abid et al., 2021, Perez et al., 2022]. Thus, an important question is how to mitigate these artifacts, which would improve the trustworthiness of these models.

One way to mitigate undesirable samples is to re-design the entire training pipeline including data augmentation, model architecture and loss functions, and then re-train the entire model from scratch [Isola et al., 2017, Aitken et al., 2017, Kaneko and Harada, 2021] – a strategy that has been used in prior work. This approach is very compute-intensive as modern GANs can be extremely expensive to train. In addition, other problems may become apparent after training, and resolving them may require multiple re-trainings. To address this challenge, we consider *post-editing*, which means modifying a pre-trained model in a certain way rather than training it differently from scratch. This is a much more computationally efficient process that has shown empirical success in many supervised learning tasks [Frankle and Carbin, 2018, Zhou et al., 2021, Taha et al., 2021], but has not been studied much for unsupervised learning. In particular, we propose a post-editing framework to redact undesirable samples that might be generated by a GAN, which we call *data redaction*.

A second plausible solution for mitigating undesirable samples is to use a classifier to filter them out after generation. This approach, however, has several drawbacks. Classifiers can take a significant amount of space and time after deployment. Additionally, if the generative model is handed to a third party, then the model trainer has no control over whether the filter will ultimately be used. Data redaction via post-editing, on the other hand, offers a cleaner solution which does not suffer from these limitations.

A third plausible solution is data deletion or machine unlearning – post-edit the model to approximate a re-trained model that is obtained by re-training from scratch after removing the undesirable samples from the training data. However, this does not always work – as we show in Section 4.4.2, deletion does not necessarily lead to redaction in constrained models. Additionally, the undesirable samples may simply due to inductive biases of the neural generative model and may not exist in the training data; examples include unnatural sounds emitted by speech models and blurred images from image models. Data redaction, in contrast, can address all these challenges.

There are two major technical challenges that we need to resolve in order to do effective data redaction. The first is how to describe the samples to be redacted. This is important as data redaction algorithms need to be tailored to specific descriptions. The second challenge is that we need to carefully balance data redaction with retaining good generation quality, which means the latent space and the networks must be carefully manipulated.

In this work, we propose a systematic framework for redacting data from pre-trained generative models (see Section 4.2). We model data redaction as learning the data distribution restricted to the complement of a redaction set Ω. We then formalize three ways of describing redaction sets, namely data-based (where a pre-specified set is given), validity-based (where there is a validity checker), and classifier-based (where there is a differentiable classifier).

Then, we introduce three data redaction algorithms, one for each description (see Section 4.3). Prior works have looked at avoiding negative samples in the re-training setting with different descriptions and purposes [Sinha et al., 2020, Asokan and Seelamantula, 2020]. They introduce fake distributions to penalize the generation of negative samples. We extend this idea to data redaction by defining the fake distribution as a mixture of the generative distribution and a redaction distribution supported on Ω. We prove the optimal generator can recover the target distribution when label smoothing [Salimans et al., 2016, Szegedy et al., 2016, Warde-Farley and Goodfellow, 2016] is used.

Based on our theory, we introduce the data-based redaction algorithm (Alg. 1). We then combine this algorithm with an improper active learning algorithm by Hanneke et al. [2018] and introduce the validity-based redaction algorithm (Alg. 2). Finally, we propose to use a `guide` function to guide the discriminator via a classifier, and introduce the classifier-based redaction algorithm (Alg. 3).

Finally, we empirically evaluate these redaction algorithms via experiments on real-world image datasets (see Section 4.4). We show that these algorithms can redact quickly while keeping high generation quality. We then investigate applications of data redaction, and use our algorithms to remove different biases that may not exist in the training set but are learned by the pre-trained model. This demonstrates that data redaction can be used to reduce biases and improve generation quality, and hence improve the trustworthiness of generative models.

In summary, our contributions are as follows:

- We formalize the problem of post-editing generative models to prevent them from outputting undesirable samples as "data redaction" and establish its differences with data deletion.

- We propose three data augmentation-based algorithms for redacting data from pre-trained GANs as a function of how the inputs to be redacted are described.
- We theoretically prove that data redaction can be achieved by the proposed algorithms.
- We extensively evaluate our algorithms on real world image datasets. We show these algorithms can redact data quickly while retaining high generation quality. Moreover, we find data redaction performs better than data deletion in a de-biasing experiment.

4.2 A Formal Framework for Data Redaction

Let p_{data} be the data distribution on $\mathbb{R}^d$ and $X \sim p_{\text{data}}$ be i.i.d. training samples. Let $\mathscr{A}$ be the learning algorithm of generative modelling and $\mathscr{M} = \mathscr{A}(X)$ be the pre-trained model on X, which learns p_{data}. In this chapter, we consider $\mathscr{A}$ to be a GAN learning algorithm [Goodfellow et al., 2014], and $\mathscr{M}$ contains two networks, D (discriminator) and G (generator), which are jointly trained to optimize

$$\min_G \max_D \ \mathbb{E}_{x\sim p_{\text{data}}} \log D(x) + \mathbb{E}_{x\sim p_G} \log(1 - D(x)),$$

where p_G is the push-forward distribution $G\#\mathscr{N}(0,I)$ defined as the distribution of $G(Z)$ where $Z \sim \mathscr{N}(0,I)$.

4.2.1 Data Redaction Framework

Let the redaction set $\Omega \subset \mathbb{R}^d$ be the set of samples we would like the model to redact. Formally, the goal is to develop a redaction algorithm $\mathscr{D}$ such that $\mathscr{M}' = \mathscr{D}(\mathscr{M}, \Omega)$ learns the data distribution restricted to the complement $\bar{\Omega} = \mathbb{R}^d \setminus \Omega$, i.e. $p_{\text{data}}|_{\bar{\Omega}}$. Examples of Ω include inconsistent, blurred, unrealistic, or banned samples that are possibly generated by the model.

The redaction set Ω, in addition to the pre-trained model, is considered as an input to the redaction algorithm. We consider three kinds of Ω, namely data-based, validity-based, and classifier-based.

4.2.2 Redaction Set Descriptions

We propose three different descriptions for the redaction set Ω. First, the data-based Ω is a pre-defined set of samples in $\mathbb{R}^d$, such as a transformation applied on all training samples [Sinha et al., 2020]. Second, the validity-based Ω is defined as all invalid samples according to a validity function $\mathbf{v}:\mathbb{R}^d \to \{0,1\}$, where 0 means invalid and 1 means valid. This is similar to the setting in Hanneke et al. [2018]. Finally, let $\mathbf{f}:\mathbb{R}^d \to [0,1]$ be a soft classifier that outputs the probability that a sample belongs to a certain binary class, and $\tau \in (0,1)$ be a threshold. Then, the classifier-based Ω is defined as $\{x:\mathbf{f}(x) < \tau\}$. For example, $\mathbf{f}$ can be an offensive text classifier in language generation tasks [Pitsilis et al., 2018]. These descriptions are general and apply to any kind of generative models.

4.2.3 Data Deletion versus Data Redaction

Motivated by privacy laws such as the GDPR and the CCPA, there has been a recent body of work on data deletion or machine unlearning [Cao and Yang, 2015, Guo et al., 2019, Schelter, 2020, Neel et al., 2021, Sekhari et al., 2021, Izzo et al., 2021, Ullah et al., 2021]. In data deletion, we are given a subset set $X' \subset X$ of the training set to be deleted from an already-trained model, and the goal is to approximate the re-trained model $\mathscr{A}(X \setminus X')$. While there are some superficial similarities – in that the goal is to post-edit models in order to "remove" a few data points, there are two very important differences.

The first is that data redaction requires the model to assign zero likelihood to the redaction set Ω in order to avoid generating samples from this region; this is not the case in data deletion – in fact, we present an example below which shows that data deletion of a set X' may not cause a generative model to redact X'.

Specifically, in Fig. 4.1, the entire data distribution $p_{\text{data}} = \mathscr{N}(0,1)$ (blue line) is the standard Gaussian distribution on $\mathbb{R}$. We set the redaction set $\Omega = (-\infty, -1.5] \cup [1.5, \infty)$, so the blue samples fall in Ω and orange samples outside. The learning algorithm $\mathscr{A}$ is the maximum likelihood Gaussian learner that fits the mean and variance of the data. With $n = 80$ samples, the learnt density $\mathscr{A}(X)$ is shown in green. If the blue samples were **deleted**, and the model re-fitted, the newly learnt density $\mathscr{A}(X \setminus X')$ would be the red line. Notice that this red line has considerable density on the blue points – and so these points are not redacted. In contrast, the correct **redaction** solution that redacts samples in Ω would be the orange density. Thus deletion does not necessarily lead to redaction.

The second difference is that the redaction set Ω may have a *zero intersection* with the training data, but may appear in the generated data due to artifacts of the model. Examples include unnatural sounds emitted by speech models, and blurred images from image models. Data redaction, in contrast to data deletion, can address this challenge.

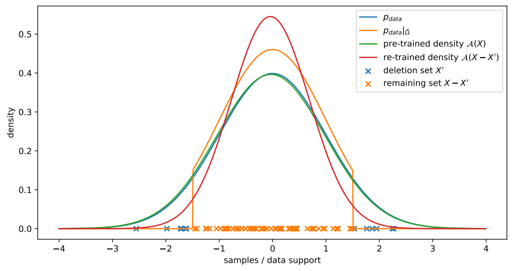


Figure 4.1. An example showing difference between data redaction and data deletion. The goal of **data deletion** is to approximate the re-trained model (red density), while the goal of **data redaction** is to approximate the restricted density (orange density).

4.3 Methods

In this section, we describe algorithms for each kind of redaction set described in Section 4.2. We also provide theory on the optimality of the generator and the discriminator. Finally, we generalize the algorithms to situations where we would like the model to redact the union of multiple redaction sets.

4.3.1 Data-based Redaction Set

The data-based redaction set Ω is a pre-defined set of samples we would like the model to redact. One example is a transformation function `NegAug` applied to all training samples, where `NegAug` makes realistic images unrealistic or inconsistent [Sinha et al., 2020]. Another example can be visually nice samples outside data manifold when the training set is small [Asokan and Seelamantula, 2020].

In our framework, the redaction set Ω can be any set of carefully designed or selected samples depending on the purpose of redacting them – which includes but does not limit to improving the generation quality of the model. For example, we expect the model to improve on fairness, bias, ethics or privacy when Ω is properly constructed with unfair, biased, unethical, or atypical samples.

To redact Ω, we regard both generated samples and Ω to be fake samples, and all training samples that are not in Ω to be real samples [Sinha et al., 2020, Asokan and Seelamantula, 2020]. Let p_Ω be a redaction distribution such that $\mathbf{supp}\,(p_\Omega) = \Omega$. Then, the fake data distribution p_{fake} is a mixture of the generative distribution p_G and the redaction distribution p_Ω:

$$p_{\text{fake}} = \lambda \cdot p_G + (1-\lambda) \cdot p_\Omega, \tag{4.1}$$

where $\lambda \in (0,1)$ is a hyperparameter. We also apply the common label smoothing [Salimans et al., 2016, Szegedy et al., 2016, Warde-Farley and Goodfellow, 2016] technique to the minimax loss function in order to improve robustness of the discriminator. Let $\alpha_+ \in (\frac{1}{2}, 1]$ be the positive target (such as 0.9) and $\alpha_- \in [0, \frac{1}{2})$ be the negative target (such as 0.1). Then, the loss function is

$$\begin{aligned} L(G,D) = \ & \mathbb{E}_{x\sim p_{\text{data}}|_{\bar{\Omega}}} \left[\alpha_+ \log D(x) + (1-\alpha_+) \log(1-D(x))\right] \\ + \ & \mathbb{E}_{x\sim p_{\text{fake}}} \left[\alpha_- \log D(x) + (1-\alpha_-) \log(1-D(x))\right]. \end{aligned} \tag{4.2}$$

Algorithm 1. Redaction Algorithm for Data-based Redaction Set

```
**Inputs**: Pre-trained model $\mathscr{M} = (G_0, D_0)$, train set $X$, redaction set $\Omega$.
Initialize $G = G_0$, $D = D_0$.
Define the fake data distribution $p_{\text{fake}}$ according to (4.1) with $p_\Omega = \mathscr{U}(\Omega)$.
Train $G, D$ to optimize (4.2): $\min_G \max_D L(G,D)$.
**return** $\mathscr{M}' = (G, D)$.
```

Algorithm 2. Redaction Algorithm for Validity-based Redaction Set

```
**Inputs**: Pre-trained model $\mathscr{M} = (G_0, D_0)$, train set $X$, validity function $\mathbf{v}$.
Initialize $\Omega' = \{x \in X : \mathbf{v}(x) = 0\}$, $\mathscr{M}_0 = \mathscr{M}$.
**for** $i = 0, \cdots, R-1$ **do**
    Initialize $G = G_i$, $D = D_i$. Draw $T$ samples $X_{\text{gen}}^{(i)}$ from $G_i$.
    Query $\mathbf{v}$ and add invalid samples to $\Omega'$: $\Omega' \leftarrow \Omega' \cup \{x \in X_{\text{gen}}^{(i)} : \mathbf{v}(x) = 0\}$.
    Define the fake data distribution $p_{\text{fake}}$ according to (4.1) with $p_\Omega = \mathscr{U}(\Omega')$.
    Let $\mathscr{M}_{i+1} = (G_{i+1}, D_{i+1})$ optimize (4.2): $\min_G \max_D L(G,D)$.
**end for**
**return** $\mathscr{M}' = (G_R, D_R)$
```

Theorem 4.3.1. *The optimal solution to* $\min_G \max_D L(G,D)$ *is*

$$\begin{aligned} D^* &= \frac{\alpha_+ p_{\text{data}}|_{\bar{\Omega}} + \alpha_-(\lambda p_G + (1-\lambda) p_\Omega)}{p_{\text{data}}|_{\bar{\Omega}} + \lambda p_G + (1-\lambda) p_\Omega} \\ p_{G^*} &= p_{\text{data}}|_{\bar{\Omega}} \end{aligned} \quad . \tag{4.3}$$

We provide the proof and theoretical extension to the more general f-GAN [Nowozin et al., 2016] setting in Appendix D.1. In the data-based setting, we let $p_\Omega = \mathscr{U}(\Omega)$, the uniform distribution on Ω. We assume Ω has positive, finite Lebesgue measure in $\mathbb{R}^d$ so that $\mathscr{U}(\Omega)$ is well-defined. The proposed method is summarized in Alg. 1.

Our objective function is connected to Sinha et al. [2020] and Asokan and Seelamantula [2020] in the sense that p_Ω is an instance of the negative distribution described in their frameworks. However, there are several significant differences between our method and theirs: (1) we start from a pre-trained model, (2) we aim to learn $p_{\text{data}}|_{\bar{\Omega}}$ rather than p_{data} and therefore do not require $\Omega \cap \mathbf{supp}\ (p_{\text{data}})$ to be the empty set, and (3) we consider the common label smoothing technique and provide theory for this setting. These differences are also true in the following sections.

Algorithm 3. Redaction Algorithm for Classifier-based Redaction Set

```
**Inputs**: Pre-trained model $\mathscr{M} = (G_0, D_0)$, train set $X$, differentiable classifier $\mathbf{f}$.
Initialize $G = G_0$, $D = D_0$.
Define the fake data distribution $p_{\text{fake}}$ according to (4.1) with $p_\Omega = \mathscr{U}(\{x \in X : \mathbf{f}(x) < \tau\})$.
Train $G, D$ to optimize (4.2): $\min_G \max_D L(G, \texttt{guide}(D, \mathbf{f}))$, where $\texttt{guide}(\cdot,\cdot)$ is defined in (4.5).
**return** $\mathscr{M}' = (G, D)$.
```

4.3.2 Validity-based Redaction Set

Let $\mathbf{v} : \mathbb{R}^d \to \{0,1\}$ be a validity function that indicates whether a sample is valid. Then, validity-based redaction set Ω is the set of all invalid samples $\{x : \mathbf{v}(x) = 0\}$. For example, $\mathscr{M}$ is a code generation model, and $\mathbf{v}$ is a compiler that indicates whether the code is free of syntax errors [Hanneke et al., 2018]. Different from the data-based setting, the validity-based Ω may have infinite Lebesgue measure, such as a halfspace, and consequently $\mathscr{U}(\Omega)$ may not be well-defined.

To redact Ω, we let p_Ω in (4.1) to be a mixture of $p_{\text{data}}|_\Omega$ and $p_G|_\Omega$. This corresponds to a simplified version of the improper active learning algorithm introduced by Hanneke et al. [2018] with our Alg. 1 as their optimization oracle. The idea is to apply Alg. 1 for R rounds. After each round, we query the validity of T newly generated samples and use invalid samples to form a data-based redaction set Ω'. In contrast to the data-based approach, this active algorithm focuses on invalid samples that are more likely to be generated, and therefore efficiently penalizes generation of invalid samples. The proposed method is summarized in Alg. 2.

The total number of queries to the validity function $\mathbf{v}$ is $|X| + T \times R$. In case $\mathbf{v}$ is expensive to run, we would like to achieve better data redaction within a limited number of queries. From the data-driven point of view, we hope to collect as many invalid samples as possible. This is done by setting $R = 1$ and T maximized if we assume less invalid samples are generated after each iteration. However, this may not be the case in practice. We hypothesis some samples are easier to redact while others harder. By setting $R > 1$, we expect an increasing fraction of invalid generated samples to be hard to redact after each iteration. Focusing on these hard samples can potentially help the generator redact them. Since it is hard to directly analyze neural networks, we leave the rigorous study to future work. In Appendix D.2, we study a much simplified dynamical system corresponding to Alg. 2, where we show the invalidity (the mass of p_G on Ω) converges to zero, and provide optimal T and R values.

4.3.3 Classifier-based Redaction Set

We would like the model to redact samples with certain (potentially undesirable) property. Let $\mathbf{f}: \mathbb{R}^d \to [0,1]$ be a soft binary classifier on the property (0 means having the property and 1 means not having it), and $\tau \in (0,1)$ be a threshold. The classifier-based redaction set Ω is then defined as $\{x : \mathbf{f}(x) < \tau\}$. For example, the property can be *being offensive* in language generation, *containing no speech* in speech synthesis, or *visual inconsistency* in image generation. We consider $\mathbf{f}$ to be a trained machine learning model that is fully accessible and differentiable.

To redact Ω, we let p_Ω be a mixture of $p_{\text{data}}|_\Omega$ and $p_G|_\Omega$, similar to the validity-based approach. We use $\mathbf{f}$ to guide the discriminator and make it able to easily detect samples from Ω. Let $\texttt{guide}(D,\mathbf{f})$ be a guided discriminator that assigns small values to x when $\mathbf{f}(x) < \tau$ or $D(x)$ is small (i.e. $x \sim p_{\text{fake}}$), and large values to x when $\mathbf{f}(x) > \tau$ and $D(x)$ is large (i.e. $x \sim p_{\text{data}}|_{\bar{\Omega}}$). Instead of optimizing $L(G,D)$ in (4.2), we optimize $L(G,\texttt{guide}(D,\mathbf{f}))$. This will effectively update G by preventing it from generating samples in Ω. According to **Theorem** 4.3.1, the optimal discriminator is the solution to

$$\texttt{guide}(D^*,\mathbf{f}) = \frac{\alpha_+ p_{\text{data}}|_{\bar{\Omega}} + \alpha_-(\lambda p_G + (1-\lambda)p_\Omega)}{p_{\text{data}}|_{\bar{\Omega}} + \lambda p_G + (1-\lambda)p_\Omega}. \tag{4.4}$$

Therefore, the design of the guide function must make (4.4) feasible. In this chapter, we let

$$\texttt{guide}(D,\mathbf{f})(x) = \begin{cases} D(x) & \text{if } \mathbf{f}(x) \geq \tau \\ \alpha_- + (D(x)-\alpha_-)\mathbf{f}(x) & \text{otherwise} \end{cases}. \tag{4.5}$$

The feasibility of (4.4) is discussed in Appendix D.3. The proposed method is summarized in Alg.

3. The classifier-based Ω generalizes the validity-based Ω. First, any validity-based Ω can be represented by a classifier-based Ω if we let $\mathbf{f} = \mathbf{v}$ and $\tau = \frac{1}{2}$. Next, we note there is a trivial way to deal with classifier-based Ω via the validity-based approach – by setting $\mathbf{v}(x) = 1\{\mathbf{f}(x) < \tau\}$. However, potentially useful information such as values and gradients of **f** are lost, and we will evaluate this effect in experiments. In addition, the classifier-based approach does not maintain the potentially large set of invalid generated samples, as this step is automatically done in the `guide` function.

4.3.4 Generalization to Multiple Redaction Sets

Let $\{\Omega_k\}_{k=1}^K$ be disjoint sets in $\mathbb{R}^d$, and we would like the model to redact $\Omega = \bigcup_{k=1}^K \Omega_k$. In the data-based setting, we let $p_\Omega = \mathscr{U}(\Omega) = \mathscr{U}(\bigcup_{k=1}^K \Omega_k)$. In the validity-based setting, each Ω_k is associated with a validity function $\mathbf{v}_k$. We let the overall validity function to be $\mathbf{v}(x) = \min_k \mathbf{v}_k(x)$. In the classifier-based setting, each Ω_k is associated with a classifier $\mathbf{f}_k$. Similar to the validity-based setting, we let the overall **f** to be $\mathbf{f}(x) = \min_k \mathbf{f}_k(x)$.

4.4 Experiments

In this section, we aim to answer the following questions.

- How well can the algorithms in Section 4.3 redact samples in practice?
- Can these algorithms be used to de-bias pre-trained models?
- Can these algorithms be used to understand training data?

In this section, we examine these questions by focusing on several real-world image datasets, including MNIST (28×28) [LeCun et al., 2010], CIFAR (32×32) [Krizhevsky and Hinton, 2009], CelebA (64×64) [Liu et al., 2015] and STL-10 (96×96) [Coates et al., 2011] datasets. In Section 4.4.1, we investigate how well these algorithms can redact samples with a specific label. In Section 4.4.2, we investigate how well these algorithms can de-bias pre-trained models

Table 4.1. Invalidity and generation quality of different redaction algorithms on redacting label zero within different datasets. The invalidity drops in magnitude after data redaction. Different redaction algorithms are highly comparable to each other.

Dataset	Evaluation	Pre-trained	Data-based	Validity-based	Classifier-based
MNIST	$\texttt{Inv}(\downarrow, \times 10^{-5})$	1.1×10^4	8.0 ± 2.2	6.4 ± 0.8	$\mathbf{5.2 \pm 3.7}$
(8 epochs)	$\texttt{IS}(\uparrow)$	7.82	$\mathbf{7.20 \pm 0.08}$	7.19 ± 0.04	7.16 ± 0.04
CIFAR-10	$\texttt{Inv}(\downarrow, \times 10^{-3})$	1.3×10^2	$\mathbf{7.5 \pm 1.1}$	7.6 ± 1.0	11.6 ± 1.0
(30 epochs)	$\texttt{FID}(\downarrow)$	36.2	34.8 ± 1.5	34.8 ± 1.4	$\mathbf{33.2 \pm 0.6}$
STL-10	$\texttt{Inv}(\downarrow, \times 10^{-4})$	6.2×10^2	8.8 ± 4.5	$\mathbf{7.7 \pm 1.3}$	11.6 ± 3.6
(40 epochs)	$\texttt{FID}(\downarrow)$	79.1	77.8 ± 2.2	$\mathbf{77.0 \pm 2.3}$	77.2 ± 1.5

and improve generation quality. In Section 4.4.3, we use these algorithms to understand training data through the lens of data redaction.

The pre-trained model for each dataset is a DCGAN [Radford et al., 2015] trained for 200 epochs (see details in Appendix D.4). We use one NVIDIA 3080 GPU to train these models and run experiments.

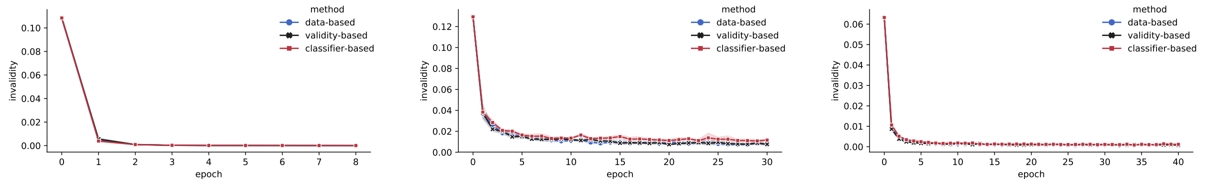
Evaluation Metrics: invalidity and generation quality. The invalidity is defined as the mass of the generation distribution on the redaction set Ω: $\texttt{Inv}(p_G) = \int_{x \in \Omega} p_G(x)dx$. In practice, we measure invalidity by generating 50K samples and computing the fraction of these samples that fall into Ω.

The generation quality is measured in Inception Score (`IS`) [Salimans et al., 2016] and Frechet Inception Distance (`FID`) [Heusel et al., 2017]. Higher `IS` or lower `FID` indicates better quality. We compute `IS` for grey-scale images and `FID` for RGB images. When measuring quality, we compute `IS` or `FID` between 50K generated samples and $X \cap \bar{\Omega}$. Therefore, this score is not comparable with the score w.r.t. the pre-trained model if the redaction set includes samples in the training set, such as samples with a specific label in Section 4.4.1. Detailed setup is in Appendix D.4.

4.4.1 Redacting Labels

Question. How well can the algorithms in Section 4.3 redact samples in practice?

Methodology. We investigate how well the proposed algorithms can redact samples



(a) MNIST **(b)** CIFAR-10 **(c)** STL-10

Figure 4.2. Invalidity during data redaction when redacting label zero. Mean and standard errors are plotted for five random seeds. Standard errors may be too small to spot. Invalidity drops quickly at the beginning of data redaction, and different algorithms are highly comparable to each other.

with a specific label y. In the data-based setting (Alg. 1), we express this as $\Omega = \{x \in X : \texttt{label}(x) = y\}$. In the validity-based setting (Alg. 2), we express this by setting $\mathbf{v}(x) = 1\{\arg\max_i \texttt{logit}(x)_i \neq y\}$, where $\texttt{logit}$ is the output of the softmax layer of a pre-trained label classifier [Chen, 2020]. In the classifier-based setting (Alg. 3), we set $\mathbf{f}(x) = 1 - \texttt{logit}(x)_y$.

In Table 4.1, we compare invalidity and generation quality among different algorithms and datasets when we redact label 0. We plot invalidity during data redaction in Fig. 4.2. We also compare invalidity after one epoch of data redaction in Appendix D.5.1. Mean and standard errors for 5 random runs are reported. Results for different hyper-parameters and redacting other labels are in Appendix D.5.

Results. We find all the algorithms in Section 4.3 work quite well with a much fewer number of epochs used for training the pre-trained model (which is 200). These algorithms are generally comparable. Therefore, we conclude that the simplest data-based algorithm is good enough to redact samples when those training samples to be redacted ($X \cap \Omega$) can characterize the redaction set (Ω) well.

We also find invalidity rapidly drops after only one epoch of data redaction, indicating these algorithms are very efficient in penalizing invalidity. While different algorithms perform better on different datasets, they are highly comparable with each other. The reason why the classifier-based algorithm performs the best on MNIST is possibly that the label classifier on MNIST is almost perfect so its gradient information is accurate.

Visualization. We sample latents $z \sim \mathcal{N}(0, I)$ and choose those corresponding to invalid

Table 4.2. Study on the effect of T in Alg. 2 when the total number of queries is fixed. R refers to the number of epochs of data redaction. A large T may lead to worse invalidity.

T	MNIST			CIFAR-10			STL-10		
	R	Inv(↓)	IS(↑)	R	Inv(↓)	FID(↓)	R	Inv(↓)	FID(↓)
400	20	$\mathbf{0.0 \times 10^{-4}}$	7.10	75	$\mathbf{0.45 \times 10^{-2}}$	35.1	100	1.0×10^{-3}	**75.1**
1000	8	0.6×10^{-4}	**7.19**	30	0.76×10^{-2}	34.8	40	$\mathbf{0.8 \times 10^{-3}}$	77.0
2000	4	2.8×10^{-4}	7.11	15	1.00×10^{-2}	**31.9**	20	1.0×10^{-3}	**75.1**

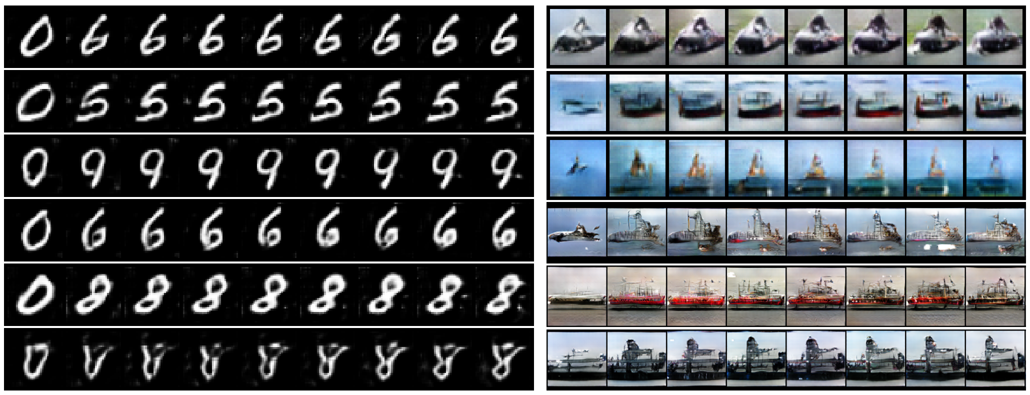


Figure 4.3. Visualization of the data redaction process of invalid samples when redacting label zero. The first column is from the pre-trained generator, and the i-th column is generated after $k \cdot (i-1)$ epochs of redaction. Left: MNIST with $k=1$. Right: top is CIFAR-10 and bottom is STL-10, both with $k=4$.

samples, i.e. $G_0(z) \in \Omega$ where G_0 is the pre-trained generator. We select visually good $G_0(z)$ for demonstration. We visualize $G(z)$ during data redaction in Fig. 4.3. This demonstrates how the latent space is manipulated: the label to be redacted is gradually pushed to other labels, and there is high-level visual similarity between the final $G(z)$ and the original $G_0(z)$.

Effects of other hyper-parameters. In Table 4.2, we compare different T (#queries after each epoch) in the validity-based redaction algorithm (Alg. 2). We fix the total number of queries by setting $T\times$#epochs to be a constant. Results indicate that a large T may lead to worse invalidity, and there is trade-off between invalidity and quality when setting T to be small or moderate.

In Appendix D.5.1, we compare different λ (hyperparameter in (4.1)) in the classifier-based redaction algorithm (Alg. 3). We find there exists a clear trade-off between invalidity and quality when alternating λ: a larger λ tends to produce better quality, and a smaller λ tends to

Table 4.3. Comparing (classifier-based) data redaction to correlated data deletion on CIFAR-10. Data redaction has better invalidity and generation quality than the data deletion baseline.

	Label	0	1	2	3	4	5
`Inv`(↓)	Data redaction (30 epochs)	1.1%	0.08%	1.6%	2.5%	1.6%	1.5%
	Data deletion (200 epochs)	6.2%	0.14%	5.6%	9.3%	10.1%	2.9%
`FID`(↓)	Data redaction (30 epochs)	33.2	33.4	28.3	28.1	29.7	31.4
	Data deletion (200 epochs)	40.0	40.5	40.0	39.5	40.0	39.3

Table 4.4. Invalidity and generation quality of different redaction algorithms on redacting a combination of attributes within CelebA. There is a significant drop of invalidity, indicating that different redaction algorithms can all generalize to multiple redaction sets.

Evaluation	Pre-trained	Epochs	Data-based	Validity-based	Classifier-based
`Inv`(↓)	1.66×10^{-3}	1	9.0×10^{-4}	7.6×10^{-4}	$\mathbf{7.0 \times 10^{-4}}$
`Inv`(↓)	1.66×10^{-3}	5	$\mathbf{3.8 \times 10^{-4}}$	6.8×10^{-4}	6.8×10^{-4}
`FID`(↓)	36.4	5	29.3	29.9	**27.9**

have better invalidity.

Comparison to data deletion. In Table 4.3, we compare data redaction to a data deletion baseline where we re-train the model after deleting correlated samples to Ω. Correlated samples are defined as those having a cosine similarity ≤ 0.25 with some sample in Ω. We find data redaction has better invalidity and generation quality than the data deletion baseline. For several labels, data deletion does not successfully prohibit samples with these labels to be generated.

Redacting multiple sets. We then investigate how well the proposed algorithms can generalize to multiple redaction sets with methods in Section 4.3.4. We focus on the CelebA dataset [Liu et al., 2015], which has 40 labeled attributes. We use proposed algorithms to redact a combination of these attributes: $\Omega_1 = \{\texttt{Black\_hair and Blurry}\}$, $\Omega_2 = \{\texttt{Brown\_hair and Wear\_eyeglasses}\}$, and $\Omega = \Omega_1 \cup \Omega_2$. These attributes are randomly selected from those easy to capture. See detailed setup in Appendix D.5.3. Results after 1 or 5 epochs are reported in Table 4.4. Consistent with results on redaction just one label, all algorithms can reduce invalidity and retain generation quality and are comparable, while the classifier-based algorithm achieves the best invalidity after one epoch.

Table 4.5. Invalidity after de-biasing boundary artifacts of generated MNIST samples. We run the validity-based redaction algorithm (Alg. 2) for 4 epochs. The invalidity drops significantly, and a small or moderate T leads to slightly lower (better) invalidity.

	Pre-trained	$T = 5\text{K}$	$T = 10\text{K}$	$T = 20\text{K}$	$T = 40\text{K}$	$T = 80\text{K}$
Margin = 1	3.1×10^{-3}	$\mathbf{6.0 \times 10^{-5}}$	8.0×10^{-5}	2.0×10^{-4}	2.0×10^{-4}	7.0×10^{-4}
Margin = 2	1.1×10^{-3}	1.6×10^{-4}	$\mathbf{4.0 \times 10^{-5}}$	6.0×10^{-5}	3.2×10^{-4}	2.8×10^{-4}

4.4.2 Model De-biasing

There can be different artifacts in GAN generated samples, and these could harm the overall generation quality. These artifacts may not exist in training samples, but are caused by inductive biases of the model, and become obvious after training. We can post-edit a pre-trained model to remove these artifacts, which we call *model de-biasing*. In this section, we investigate how well Alg. 2 and Alg. 3 apply to this task. We assume training samples are not biased so Alg. 1 does not apply to de-biasing.

To use these algorithms for de-biasing, we assume the target artifact or bias can be automatically detected by a classifier **f** or a validity function **v**. Specifically, we survey two kinds of biases: boundary artifacts and label biases.

Boundary artifacts. A GAN trained on MNIST might generate samples that have numerous white pixels on the boundary (see Appendix D.6.1). We call this phenomenon the *boundary artifact*. We use the validity-based algorithm (Alg. 2) to de-bias boundary artifacts. The validity function is defined as $\mathbf{v}(x) = 1\{\sum_{(i,j)\in\text{boundary pixels}} x_{ij} < \tau_b\}$, where boundary pixels are those within a certain margin to the boundary, and threshold τ_b satisfies no training image is invalid.

Results are reported in Table 4.5. It is clear that the invalidity reduces in order after data redaction, indicating boundary artifacts are largely removed. Consistent with Table 4.2, a small or moderate T leads to better results. We visualize samples before and after de-biasing in Appendix D.6.1.

Label biases. Neural networks may generate visually smooth but semantically ambigu-

Table 4.6. Invalidity and IS after de-biasing label biases on MNIST. We run Alg. 3 for 8 epochs with $\lambda = 0.8$ The arrow means improvement after data redaction. There is a clear improvement of generation quality, indicating the proposed algorithm can help GANs generate better samples.

τ	Redaction (8 epochs)		Data deletion baseline (200 epochs)	
	`Inv`(↓)	`IS`(↑)	`Inv`(↓)	`IS`(↑)
0.3	$8.19 \times 10^{-4} \to 2.60 \times 10^{-4}$	**8.10**	$8.19 \times 10^{-4} \to 1.14 \times 10^{-3}$	7.75
0.5	$2.07 \times 10^{-2} \to 1.70 \times 10^{-2}$	7.92	$2.07 \times 10^{-2} \to 2.17 \times 10^{-2}$	7.79
0.7	$1.35 \times 10^{-1} \to 1.22 \times 10^{-1}$	7.95	$1.35 \times 10^{-1} \to 1.32 \times 10^{-1}$	7.82

Table 4.7. Invalidity and FID scores after de-biasing label biases on CIFAR-10. We run Alg. 3 for 30 epochs with $\lambda = 0.9$. The arrow means improvement after data redaction. There is a clear improvement of generation quality, indicating the proposed algorithm can help GANs generate better samples.

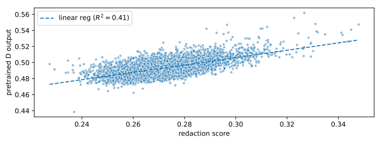
τ	`Inv`(↓)	`FID`(↓)
0.5	$2.28 \times 10^{-2} \to 1.67 \times 10^{-2}$	$36.2 \to$ **26.6**
0.7	$1.72 \times 10^{-1} \to 1.49 \times 10^{-1}$	$36.2 \to 26.8$
0.3	$5.79 \times 10^{-4} \to 2.20 \times 10^{-4}$	$36.2 \to 27.1$

ous samples [Kirichenko et al., 2020], e.g. samples that look like multiple objects (see Appendix D.6.2). We call this phenomenon the *label bias*. We use the classifier-based algorithm (Alg. 3) to de-bias label biases. The classifier is defined as $\mathbf{f}(x) = 1 - \text{Entropy}(\texttt{logit}(x))/\log(\texttt{\#classes})$, where the `logit` function is the same as in Section 4.4.1. We also compare to a data deletion baseline, where we delete invalid samples and fully re-train the model. Results are reported in Table 4.6 and 4.7. After de-biasing, we can improve the generation quality by a significant gap (~ 0.3 in `IS` and ~ 10 in `FID`). There is also a clear drop in terms of invalidity. In contrast, we find that data deletion does not help removing label biases.

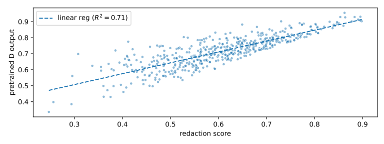
4.4.3 Understanding Training Data through the Lens of Data Redaction

Large datasets can be hard to analyze. In this section, we investigate how data redaction can help us understand these data. Specifically, we ask: which samples are easy or hard to redact?

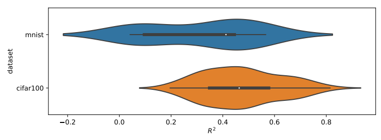
In order to quantify the difficulty to redact a sample, we define the redaction score $\mathscr{RS}$ to be the difference of discriminator outputs before and after data redaction. Formally, let $x \in \Omega$



(a) MNIST (0)



(b) CIFAR100 `aquarium_fish`



(c) Distribution of R^2

Figure 4.4. (a) and (b) Redaction scores of invalid training samples $\mathscr{RS}(x)$ versus $D_0(x)$. There is positive correlation between these two scores, indicating on-manifold samples are easier to redact. (c) Distributions of R^2 scores of linear regression between $\mathscr{RS}(x)$ and $D_0(x)$ for all labels.

be a sample to redact, $\mathscr{M} = (G_0, D_0)$ be the pre-trained model, and $\mathscr{M}' = (G', D')$ be a model after data redaction. Then, the redaction score is $\mathscr{RS}(x) = D_0(x) - D'(x)$. A larger $\mathscr{RS}$ means it is easier to redact x.

To investigate **sample-level** redaction difficulty, we redact a particular label at one time using Alg. 1. We then demonstrate scatter plots of redaction scores $\mathscr{RS}(x)$ versus pre-trained discriminator outputs $D_0(x)$ for all samples x with this label. We also fit linear regression and report R^2 values (larger means stronger linear relationship). Scatter plots for some labels in MNIST and CIFAR-100 and distribution of R^2 for all labels are shown in Fig. 4.4. We also visualize the most and least difficult-to-redact samples in Appendix D.7. We find there is positive correlation between $\mathscr{RS}(x)$ and $D_0(x)$, indicating on-manifold (large $D_0(x)$) samples are easier to be redacted, while off-manifold (small $D_0(x)$) ones are harder to be redacted. This analysis further provides a way to investigate **label-level** redaction difficulty. By averaging redaction scores for samples associated with each label, we can survey which labels are easy or hard to redact in general. The results are in Appendix D.7. We find some labels are harder to redact than others.

4.5 Related Work

Although deep generative models have been highly successful at many domains, it has long been known that they often emit undesirable samples and samples with different types of artifacts that make them untrustworthy. Examples include blurred image samples [Kaneko and Harada, 2021], fairness issues [Tan et al., 2020, Karakas et al., 2022], and checkerboard artifacts [Odena et al., 2016, Zhang et al., 2019b, Wang et al., 2020, Schwarz et al., 2021] in image generation, offensive text in language models [Abid et al., 2021, Perez et al., 2022], and unnatural sound in speech models [Donahue et al., 2018, Thiem et al., 2020].

Some prior works have used post-editing to remove artifacts and improve GANs. Examples include improving fairness [Tan et al., 2020, Karakas et al., 2022], rule rewriting [Bau et al., 2020a], discovering interpretability [Härkönen et al., 2020], and fine-tuning [Mo et al., 2020, Li et al., 2020, Zhao et al., 2020]. The purpose, use cases, and editing methods of these works are different from our work, where we focus on data redaction.

While our problem definition and formalization is novel, the technical solutions that we propose are related to three prior works that use these techniques in different contexts. These are NDA [Sinha et al., 2020], Rumi-GAN [Asokan and Seelamantula, 2020], and Hanneke et al. [2018]. The first two works look at how to avoid generating negative samples while training a generative model from scratch. This is done by defining new fake distributions to penalize the generation of these samples. However, their purposes are different from us: NDA is used to characterize the boundary of the support of the generative distribution more precisely, and Rumi-GAN is used to handle unbalanced data. We extend their idea and theory to data redaction in Section 4.3. [1] Hanneke et al. [2018] propose an active learning approach to avoid generating invalid samples, also while training a generative model from scratch. Their work however is entirely theoretical and apply to discrete distributions. In our work, the validity-based redaction algorithm (Alg. 2) is based on a simplified version of their algorithm. We also use their definition of *invalidity* as an evaluation method.

Our work is also related to data deletion or machine unlearning [Cao and Yang, 2015, Guo et al., 2019, Schelter, 2020, Neel et al., 2021, Sekhari et al., 2021, Izzo et al., 2021, Ullah et al., 2021, Bourtoule et al., 2021]. However, there are two important differences between data deletion and data redaction. First, data deletion aims to approximate the re-trained model when

[1]The loss functions in NDA and Rumi-GAN are similar.

some training samples are removed – mostly due to privacy reasons – while in data redaction we penalize the model from knowing samples that should be redacted. Another difference is that in data redaction, the redaction set Ω may have a zero intersection with training data. These two differences are discussed in Section 4.2.3 in detail. In addition, most data deletion techniques are for supervised learning or clustering, and is much less studied for generative models.

There is also a related line of work on catastrophic forgetting in supervised learning [Kirkpatrick et al., 2017] and generative models [Thanh-Tung and Tran, 2020]. This concept is different from data redaction in that we would like the generative model to redact certain data after training, while catastrophic forgetting means knowledge learned in previous tasks is destroyed during continual learning.

4.6 Conclusion

In this chapter, we propose a systematic framework for redacting data from pre-trained generative models. We provide three different algorithms for GANs that differ on how the samples to be redacted are described. We provide theoretical results that data redaction can be achieved. We then empirically investigate data redaction on real-world image datasets, and show that our algorithms are capable of redacting data while retaining high generation quality at a fraction of the cost of full re-training. One limitation or our work is that the proposed framework only applies to unconditional generative models. It is an important future direction to define data redaction and propose algorithms for conditional generative models, which are more widely used in downstream deep learning applications.

4.7 Acknowledgements

This work was supported by NSF under CNS 1804829 and ARO MURI W911NF2110317.

This chapter is a reformatted version of the paper ”Data Redaction from Pre-trained GANs” published in *IEEE Conference on Secure and Trustworthy Machine Learning, 2023,* by

Zhifeng Kong and Kamalika Chaudhuri [Kong and Chaudhuri, 2023b]. The dissertation author was the primary researcher and author of this paper.

Chapter 5

Data Redaction from Conditional Generative Models

5.1 Introduction

Deep generative models are unsupervised deep learning models that learn a data distribution from samples and then generate new samples from it. These models have shown tremendous success in many domains such as image generation [Rombach et al., 2021, Ramesh et al., 2021, 2022, Sauer et al., 2023], audio synthesis [Kong et al., 2021, Lee et al., 2023], and text generation [OpenAI, 2023, Touvron et al., 2023]. Most modern deep generative models are conditional: the user inputs some context known as the conditionals, and the model generates samples conditioned on the context.

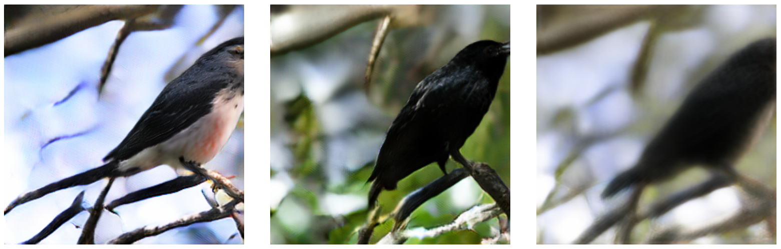
However, as these models have grown more powerful, there has been increasing concern about their trustworthiness: in certain situations, these models produce undesirable outputs. For example, with text-to-image models, one may craft a prompt that contains offensive, biased, malignant, or fabricated content, and generate a high-resolution image that visualizes the prompt [Nichol et al., 2021, Birhane et al., 2021, Schuhmann et al., 2022, Ramesh et al., 2022, Rando et al., 2022, Bedapudi, 2022, Laborde, 2022]. With speech synthesis models, one may easily turn text into celebrity voices [Betker, 2022, Wang et al., 2023, Zhang et al., 2023]. Text generation models can emit offensive, biased, or toxic content [Pitsilis et al., 2018, Wallace et al., 2019, McGuffie and Newhouse, 2020, Gehman et al., 2020, Abid et al., 2021, Perez et al., 2022, Schramowski et al., 2022b].

One plausible solution to mitigate this problem is to remove all undesirable samples from the training set and re-train the model. This is too computationally heavy for modern, large models. Another solution is to apply a classifier that filters out undesirable conditionals or outputs [Rando et al., 2022, Bedapudi, 2022, Laborde, 2022], or to edit the outputs and remove the undesirable content after generation [Schramowski et al., 2022a]. However, in cases where the model owners share the model weights with third parties, they do not have control over whether the filters or editing methods will be used. In order to prevent undesirable outputs more efficiently and reliably, we propose to *post-edit* the weights of a pre-trained model, which we call *data redaction*.

The first challenge is how to frame data redaction for conditional generative models. Prior work in data redaction for *unconditional* generative models considered this problem in the space of outputs, and framed the problem as learning the data distribution restricted to a valid subset of outputs [Kong and Chaudhuri, 2023b]. However, a conditional generative model learns a collection of (usually an infinite number of) distributions (one for each conditional) all of which are induced by networks that share weights; therefore, we cannot apply this method one by one for every conditional we would like to redact. In this paper, we frame data redaction for *conditional* generative models as redacting a set of conditionals that will very likely lead to undesirable content. In particular, we do redaction in the conditional space, instead of separately redacting samples generated from each conditional in the output space.

This framework inspires us to design a *universal*, *efficient*, and *effective* method for data redaction. We only re-train (or *distill*) the conditional part of the network by projecting redacted conditionals onto different, non-redacted reference conditionals. It is computationally light because all but the conditioning network is fixed, and we only need to load a small fraction of the dataset for training.

We show there exists an explicit data redaction formula for simple class-conditional models. For more complicated generative models in real-world applications, we introduce a



(a) Pre-trained **(b)** Our Redaction **(c)** Rewriting

Figure 5.1. Comparison between our redaction method and the Rewriting baseline [Bau et al., 2020a] when we redact `''white belly''` from text-to-image models [Zhu et al., 2019]. Samples generated from the prompt `''this bird has feathers that are black and has a white belly''`.

series of techniques to effectively redact certain conditionals but retain high generation quality. These include model-specific distillation losses and training schemes, methods to increase the capacity of the student conditioning network, ways to improve efficiency, and a few others.

We test our data redaction method on two real-world applications: GAN-based text-to-image [Zhu et al., 2019] and Diffusion-based text-to-speech [Kong et al., 2021]. For text-to-image, we redact prompts that include certain words or phrases. Our method has significantly better redaction quality and robustness than baseline methods while retains similar generation quality as the pre-trained model. For text-to-speech, we redact certain voices outside the training set. Our method achieves both high redaction and speech quality. Audio samples can be found on our demo website (https://dataredact2023.github.io/). Our methods for both applications are extremely computationally efficient: redacting text-to-image models takes approximately 0.5 hour, and redacting text-to-speech models takes less than 4 hours, both on one single NVIDIA 3080 GPU. In contrast, training the text-to-image model takes more than a day on one GPU, and training the text-to-speech model takes 2-3 days on 8 GPUs. This demonstrates that data redaction can be done significantly more efficiently than re-training full models from scratch.

5.1.1 Related Work

Machine Unlearning. Machine unlearning computes or approximates a re-trained machine learning model after removing certain training samples [Cao and Yang, 2015]. Many methods have been proposed for supervised learning [Guo et al., 2019, Schelter, 2020, Neel et al., 2021, Sekhari et al., 2021, Izzo et al., 2021, Ullah et al., 2021, Bourtoule et al., 2021] as well as generative models [Kong and Alfeld, 2022]. The goal of data redaction is very different from machine unlearning, which unlearns training samples and is usually in the privacy context, while data redaction prevents undesirable samples from generation regardless whether they are in the training set. A detailed explanation can be found in Section II-C in [Kong and Chaudhuri, 2023b].

Data Filtering and Semantic Editing. A direct way to prevent certain samples to be generated is to apply a data filter (e.g., a malicious content classifier). The filter can be applied to training data before training [Nichol et al., 2021, Schuhmann et al., 2022, Ramesh et al., 2022], or applied post-hoc to model outputs [Rando et al., 2022, Bedapudi, 2022, Laborde, 2022]. Another line of research has looked at semantically modifying the outputs of generative models. For GANs [Goodfellow et al., 2014], Bau et al. [2020b] computes an editing vector in the latent space to alter a semantic concept. For diffusion models [Ho et al., 2020] especially text-to-image models like Stable Diffusion [Rombach et al., 2021], there are also a number of image editing techniques [Bar-Tal et al., 2022, Hertz et al., 2022, Kawar et al., 2022, Valevski et al., 2022, Brack et al., 2022]. Schramowski et al. [2022a] applied image editing to prevent diffusion models from generating malicious images through a safety guidance term that alters the sampling algorithm for inappropriate prompts.

While these filtering and editing methods can be used to prevent malicious images, the model parameters are not modified. Consequently, in cases where the models owners share the model weights with third parties, they do not have control over whether the third parties will use the filters or editing methods. In contrast, our proposed method modifies the model weights to address this issue.

Data Redaction in Unconditional Models. Several works have studied methods to prevent generative models from producing undesirable samples, either by re-training or post-editing. For GANs, Asokan and Seelamantula [2020] and Sinha et al. [2021] investigated re-training methods via modified loss functions that penalize generation of undesirable samples, and Bau et al. [2020a] and Cherepkov et al. [2021] introduced post-hoc parameter rewriting techniques for semantic editing, which can be used to remove undesirable artifacts. Kong and Chaudhuri [2023b], Malnick et al. [2022], and Moon et al. [2023] designed post-editing data redaction methods for various types of pre-trained generative models.

All these methods are restricted to the unconditional setting as they modify the mapping from latent vectors to samples. In contrast, the goal of this paper is to redact data from pre-trained, *conditional* generative models. In these models, the conditional information heavily controls the content and style of generated samples (e.g. text-to-X), whereas the latent controls variation. It is therefore necessary to also modify the mapping from conditional to samples.

Redaction in Stable Diffusion via Fine-tuning. Gandikota et al. [2023] fine-tunes Stable Diffusion to incorporate negative guidance on undesirable visual styles (e.g., those under copyright protection). As a result, undesirable samples will not be generated with the standard sampling algorithm. However, one might break the safety protection by applying a strong positive guidance during sampling. In addition, this method only applies to diffusion models trained with classifier-free guidance [Ho and Salimans, 2022]. In contrast, our proposed method is universal and applies to a broader range of generative models.

5.2 Preliminaries

Conditional Generative Models. Let $\mathscr{C}$ be the space of conditionals. It could be a finite set of discrete labels, or an infinite set of continuous representations. [1] For any $c \in \mathscr{C}$ there is an underlying data distribution $p_{\text{data}}(\cdot|c)$ (on $\mathbb{R}^d$) conditioned on c. In the discrete label case, this simply corresponds to a finite number of data distributions for all labels. In the more complicated continuous case, there is usually an underlying assumption that $p_{\text{data}}(\cdot|c)$ is Lipschitz with respect to c: that is, $p_{\text{data}}(\cdot|c)$ will not change much if c does not change much.

[1] In cases where there are infinitely many discrete labels such as text or 16-bit floats, these conditionals are usually considered as continuous or transformed to continuous representations.

Let $X = \{(x_i, c_i)\}$ be the set of training data. where each x_i is the sample and c_i is the conditional (for example, x_i is an image and c_i is its caption). Let G be a conditional generative model trained on X. G has two inputs – a sample latent z drawn from a Gaussian distribution and a conditional c – and outputs sample $x = G(z|c)$. For each $c \in \mathscr{C}$, G draws from a generative distribution $p_G(\cdot|c)$, which is trained to learn $p_{\text{data}}(\cdot|c)$. In the discrete label case, this is equivalent to modeling a finite number of distributions. In the continuous case, G also needs to generalize to unseen conditionals, because not all conditionals exist in the training set. We assume that $p_G(\cdot|c)$ learns $p_{\text{data}}(\cdot|c)$ very well, as how to train these models is outside the scope of this paper.

Problem setup. Our goal is to redact a set of conditionals $\mathscr{C}_\Omega \subset \mathscr{C}$, referred to as the redaction conditionals, which with high probability lead to undesirable content. For example, for text-to-image models, we may be looking to redact text prompts related to violence or offensive content. [2]

We assume that the redaction conditionals are given to us either as a set or described by a classifier. We assume that we are working with an already trained generative model G and we are only allowed to post-edit it. Re-training generative models from scratch can be highly compute-intensive, and so our goal is to consider computationally efficient solutions. Additionally, we also want to avoid solutions that involve external filters, since a third-party can choose not to use them. A final requirement of our solution is that it should retain high generation quality for the conditionals that are not to be redacted.

We assume that we have access to the parameters of the network G and (part or whole of) its training dataset X. The goal of this paper is to edit the parameters of model G to form a new model G' so that harmful conditionals lead to the generation of benign outputs.

Our proposed solution addresses this problem in the context where the conditioning networks are separate from the main generative network – which holds for most current network

[2]This does not necessarily redact every possible offensive output; for example, an innocent prompt such as ”a day in the park” might with very low probability result in a violent image which our solution will not address.

architectures – and achieves this by distilling only the conditioning networks.

5.3 Method

In this section, we consider a special solution to our redaction task: for redacted conditionals $c \in \mathscr{C}$, we let G' learn the distribution conditioned on a different, non-redacted conditional $\hat{c} \in \mathscr{C} \setminus \mathscr{C}_\Omega$, which we denote as the reference conditional for c. Formally,

$$p_{G'}(\cdot|c) = p_G(\cdot|\hat{c}) \text{ if } c \in \mathscr{C}_\Omega, \text{ otherwise } p_G(\cdot|c). \tag{5.1}$$

Next, we introduce an efficient way to achieve (5.1). Let H be the (separate) conditioning network in the generator network G. H takes the conditional c as input and computes conditional representation $H(c)$, which is then fused into the main generative network (potentially at different layers). Our solution is to project the conditional representation $H'(c)$ of the new conditioning network to $H(\hat{c})$:

$$H'(c) = H(\hat{c}) \text{ if } c \in \mathscr{C}_\Omega, \text{ otherwise } H(c). \tag{5.2}$$

In Section 5.3.1, we show (5.2) can be done explicitly if the model is conditioned on a few discrete labels and the conditioning network is affine. In Section 5.3.2, we introduce distillation-based methods for two practical applications, where the models are conditioned on continuous representations and the network architectures are more complicated.

5.3.1 Redacting Models Conditioned on Discrete Labels

In this section, we show for simple class-conditional models, there is an explicit formula to redact a label. Suppose there are k labels: $\mathscr{C} = \{c_1, \cdots, c_k\}$, where label j is to be redacted. We consider a common type of conditioning method: each label c_i is represented by a k-dimensional embedding vector $v_i \in \mathbb{R}^k$, and H is an affine transformation whose output dimension $r \geq k$. We assume the embedding vectors are linearly independent: $\mathbf{span}\{v_1, \cdots, v_k\} = \mathbb{R}^k$. A special case of this formulation is the conditioning method proposed by Mirza and Osindero [2014], where each $v_i = \mathbf{e}_i$ is the one-hot vector with the i-th element $= 1$, and is concatenated to the latent code.

Let $H(v) = Mv$, where $M \in \mathbb{R}^{r\times k}$. The redaction problem is equivalent finding an $M' \in \mathbb{R}^{r\times k}$ such that $M'v_i = Mv_i$ for $i \neq j$ and $M'v_j = MV_{-j}\eta_{-j}$ for an one-hot vector $\eta_{-j} \in \mathbb{R}^{k-1}$, where $V_{-j} = [v_1, \cdots, v_{j-1}, v_{j+1}, \cdots, v_k] \in \mathbb{R}^{k\times(k-1)}$. The first condition $M'v_i = Mv_i$ for $i \neq j$ indicates every row of $M' - M$ is in the null space of $\{v_i\}_{i\neq j}$. The null space is a one-dimensional subspace with basis vector u. Then, $M' - M$ can be decomposed as $\omega u^\top$ for some $\omega \in \mathbb{R}^r$. Then, by to the second condition $M'v_j = MV_{-j}\eta_{-j}$, we have $\omega = \frac{1}{u^\top v_j} M(V_{-j}\eta_{-j} - v_j)$. This means by replacing M with $M' = M(I + \frac{1}{u^\top v_j}(V_{-j}\eta_{-j} - v_j)u^\top)$, we are able to redact label j. When conditioned on j, the edited model will generate another digit based on the non-zero element in η_{-j}.

Generalization. We discuss several generalizations in Appendix E.1. First, we can redact multiple labels by re-writing the formula in matrix form. Second, the editing method applies to when the dimension of v_i is larger than k. We also provide simplified formulas for one-hot embedding vectors.

5.3.2 Redacting Models Conditioned on Continuous Representations

Generally there is no explicit formula to achieve (5.2) due to non-linearity and limited expressive power of the conditioning network. We propose to distill the conditioning network by minimizing

$$\min_{H'} L(H'; \lambda) = \mathbb{E}_{c\in\mathscr{C}\setminus\mathscr{C}_\Omega} \|H'(c) - H(c)\| + \lambda \cdot \mathbb{E}_{c\in\mathscr{C}_\Omega} \|H'(c) - H(\hat{c})\| \tag{5.3}$$

for some metric $\|\cdot\|$ and balancing coefficient $\lambda > 0$. In the rest of this section, we study two types of common conditional generative models: image models conditioned on text prompts, and speech models conditioned on spectrogram representations. We will demonstrate specific losses and distillation techniques for each model that align with the slightly different goals in each task.

Redacting GAN-based Text-to-Image Models

In this section, we study how to redact text prompts in text-to-image models. Modern text-to-image models can produce high-resolution images conditioned on text prompts that may be offensive, biased, malignant, or fabricated [Nichol et al., 2021, Birhane et al., 2021, Schuhmann et al., 2022, Ramesh et al., 2022, Rando et al., 2022, Bedapudi, 2022, Laborde, 2022]. These models are usually expensive to re-train, so it is important to redact these prompts without re-training.

Especially, we look at DM-GAN [Zhu et al., 2019], a GAN-based text-to-image model. It is trained on pairs of text and images from the CUB dataset [Welinder et al., 2010, Reed et al., 2016], a dataset for various species of birds. DM-GAN is composed of three cascaded generative networks $\{G_1, G_2, G_3\}$. The first G_1 generates 64×64 images, the second G_2 up-samples to 128×128, and the third G_3 up-samples to 256×256. Each G_i has its own conditioning network H_i. For a given prompt c, the model computes a sentence embedding $v_s(c)$ and word embeddings $v_w(c)$ from a pre-trained text encoder [Xu et al., 2018]. The first conditioning network H_1 performs conditioning augmentation on the sentence embedding and concatenate the output to the latent variable. H_2 and H_3 apply memory writing modules to the word embeddings and fuse the outputs with the previously generated low-resolution images via several gates.

Defining $\hat{c}$. We assume $\mathscr{C}_\Omega$ contains prompts that have undesirable words or phrases. For these prompts, the reference prompts are defined by replacing these words with non-redacted ones.

Sequential distillation. We propose to distill the conditioning networks $\{H_1, H_2, H_3\}$ sequentially based on (5.3). This is because both G_2 and G_3 are generative super-sampling networks, which take G_1 and G_2 outputs as inputs, respectively. After G_1 is edited to G_1' for redaction, G_2 will take G_1' outputs as inputs, and similar for G_3. Formally,

$$\begin{aligned} H_1' &= \arg\min_{H_1'} \mathbb{E}_{c\in\mathscr{C}\setminus\mathscr{C}_\Omega}\|H_1'(v_s(c)) - H_1(v_s(c))\| + \lambda\cdot\mathbb{E}_{c\in\mathscr{C}_\Omega}\|H_1'(v_s(c)) - H_1(v_s(\hat{c}))\|; \\ H_i' &= \arg\min_{H_i'} \{\mathbb{E}_{c\in\mathscr{C}\setminus\mathscr{C}_\Omega,z}\|H_i'(v_w(c), G_{i-1}'(z|c)) - H_i(v_w(c), G_{i-1}'(z|c))\| \\ &\qquad + \lambda\cdot\mathbb{E}_{c\in\mathscr{C}_\Omega,z}\|H_i'(v_w(c), G_{i-1}'(z|c)) - H_i(v_w(\hat{c}), G_{i-1}'(z|\hat{c}))\|\},\ i = 2,3. \end{aligned} \tag{5.4}$$

Improved capacity. As H_1' needs to approximate a piecewise function that is defined differently for two sets of sentence embeddings, we need to increase the capacity of H_1' for better distillation. We append a few LSTM layers to the beginning of H_1', which directly take the sentence embeddings as inputs. The LSTM layers are followed by a convolution layer that reduces hidden dimensions to 1. We initialize this layer with zero weights for training stability. We expect these layers can project sentence embeddings of c to those of $\hat{c}$. The rest of H_1' has the same architecture as H_1 but all weights are initialized for training. We do not increase the capacity of H_2' and H_3' for two reasons. First, H_1' has more direct impact on the generated images because it directly controls the initial low-resolution image. Second, the memory writing modules of H_2 and H_3 are already very expressive.

Fixing the variance prediction part in H_1. We aim to reduce the computational overhead by fixing certain variables. The conditioning augmentation module in H_1 first computes a mean and a variance vector, and then samples from the Gaussian defined by them. We fix the variance prediction part and only distill the mean prediction part. In our experiments the number of parameters to be trained in H_1' (with improved capacity) is reduced by $\sim 32\%$ and therefore matches that of H_1.

λ annealing. In order to make sure the distilled conditioning networks also approximate the pre-trained ones well for non-redacted prompts, we anneal the balancing coefficient λ during distillation: we initialize $\lambda = \lambda_{\min}$ and linearly increases to $\lambda_{\max}$ in the end.

Redacting Diffusion-based Text-to-Speech Models

Modern text-to-speech models can turn text into high-quality speech in unseen voices such as celebrity voices [Kong et al., 2021, Betker, 2022, Wang et al., 2023, Zhang et al., 2023]. This may have unpredictable public impact if these models are used to fake celebrities. In this section, we study redacting certain voices from a pre-trained text-to-speech model.

Especially, we look at DiffWave [Kong et al., 2021], a diffusion probabilistic model that is conditioned on spectrogram and outputs waveform. It is trained on speech of a single female reading a book, which we call the pre-trained voice. There are $n = 30$ layers or residual blocks in DiffWave, each containing one independent conditioning network H_i. The architecture of each H_i includes two up-sampling layers followed by one convolution layer.

Defining $\hat{c}$ with voice cloning. We assume $\mathscr{C}_\Omega$ contains a few clips of speech in a specific voice. We train a voice cloning model (CycleGAN-VC2 [Kaneko et al., 2019]) between the specific and pre-trained voices, and then transform all clips in $\mathscr{C}_\Omega$ to the pre-trained voice. By doing this we obtain time-aligned pairs between $c \in \mathscr{C}_\Omega$ and the corresponding $\hat{c}$: when we select a small duration $[t, t + \Delta t]$, the content of $c_{t:t+\Delta t}$ is the same as $\hat{c}_{t:t+\Delta t}$, yet only the voices are different.

Improved voice cloning. We find the voice cloning quality of CycleGAN-VC2 can be improved by making the two unpaired training sets more similar. We first use a pre-trained Whisper model [Radford et al., 2022] to extract text from redacted speech. Then, we use Tortoise-TTS [Betker, 2022] to turn these text into speech in the pre-trained voice. Note that this cannot be used to define $\hat{c}$ directly because the generated samples are not time-aligned with the speech to be redacted. However, these generated samples are more similar to the redacted samples because they have the same text, and therefore it is easier for CycleGAN-VC2 to learn transformations between these two voices.

Parallel distillation. We propose to distill all conditional layers H_i's in parallel as they are independent. We minimize the following loss:

$$\min \frac{1}{n} \sum_{i=1}^{n} L(H_i'; \lambda). \tag{5.5}$$

Fixing up-sampling layers in H_i. To reduce computation overhead we fix the two up-sampling layers in each H_i. We only distill the last convolution layer in each H_i.

Improved capacity. To improve redaction quality, we increase the capacity of each H_i' by replacing its last convolution layer h_{conv} with a spectrogram-rewriting module. It has two components: a gate h_{gate} consisting of a convolution with zero initialization followed by sigmoid, and a transformation block h_{trans} consisting of two convolution layers. The forward computation of the spectrogram-rewriting module is defined as:

$$y = h_{\text{conv}}(v) \odot h_{\text{gate}}(v) + h_{\text{conv}}(h_{\text{trans}}(v)) \odot (1 - h_{\text{gate}}(v)), \tag{5.6}$$

where v is the up-sampled mel-spectrogram and y is the output representation at each layer. We expect this module can retain the pre-trained voice and also project redacted voices to the pre-trained voice.

Non-uniform distillation losses. We conjecture the all conditioning layers are not the same important because of their order and different hyper-parameters specifically the dilation $2^{i \bmod n'}$ in the corresponding residual layer. This motivates us to use different weights and λ values for each H_i:

$$\min \sum_{i=1}^{n} w_i L(H_i'; \lambda_i). \tag{5.7}$$

We test different schedules described in Table 5.1.

5.4 Experiments

In this section, we aim to answer the following questions. (1) Is the redaction method in Section 5.3.1 able to fully redact labels? And (2) do the redaction algorithms in Section 5.3.2

Table 5.1. Schedules for the non-uniform distillation losses.

name	schedule
w_i-order	$w_i = \frac{1}{n} + \alpha(i - (n+1)/2)$
λ_i-order	$\lambda_i = \lambda + \beta(i - (n+1)/2)$
w_i-dilation	$w_i = \frac{1}{n} + \alpha(i \bmod n' - (n'+1)/6)$
λ_i-dilation	$\lambda_i = \lambda + \beta(i \bmod n' - (n'+1)/6)$

redact certain conditionals well and retain high generation quality on real-world applications?

5.4.1 Redacting Models Conditioned on Discrete Labels

We train a class-conditional GAN called cGAN [Mirza and Osindero, 2014] on MNIST [LeCun et al., 2010]. Each conditional has a 10-dimensional embedding vector, and is concatenated to the latent vector as the input. The affine transformation matrix M in Section 5.3.1 is the last 10 rows of the weight matrix of the first fully connected layer. We redact labels `0,1,2,3` according to (E.1), where we let $\hat{c} = 9 - c$ for them. Generated samples of pre-trained and redacted models are shown in Fig. E.1.

5.4.2 Redacting GAN-based Text-to-Image Models

Setup. We use the pre-trained DM-GAN [Zhu et al., 2019] model trained on the CUB dataset [Welinder et al., 2010], which contains 8855 training images and 2933 testing images of 200 subcategories belonging to birds. Each image has 10 captions [Reed et al., 2016]. Our distillation algorithm is trained with the caption data only. We redact prompts that contain certain words or phrases. We redact the word `blue`$\in c$ by defining $\hat{c}$ as the prompt that replaces all `blue` with another word `red`. [3] Similarly, we redact `blue wings` and `red wings` by replacing these phrases to `white wings`. We redact `long beak` and `white belly` by replacing the first to `short beak` and the second to `black belly`. Finally, we redact `yellow` and `red` by replacing them to `black`, which is more challenging as many samples are redacted. More details are in Appendix E.2.

[3]Any word other than `blue` can be used.

Configurations. We first use the sequential distillation (5.4) with $\lambda = 1$ to perform redaction, which we denote as the base configuration. We then improve the capacity by using a 3-layer bidirectional LSTM with hidden size = 32 and dropout rate = 0.1. Next, we fix the variance prediction in H_1 to reduce the number of parameters to optimize, which matches the base configuration. Finally, we apply λ annealing by setting $\lambda_{\text{min}} = 1$ and $\lambda_{\text{max}} = 3$.

Baseline. We compare to the Rewriting algorithm [Bau et al., 2020a], a semantic editing method originally designed for unconditional generative models. We adapt their method to DM-GAN by rewriting G_1, G_2, and G_3 sequentially. For both G_2 and G_3 we rewrite the up-sampling layer before the feature output. For G_1 we have choices of rewriting the up-sampling layer at different resolutions ranging from 8×8 to 64×64. We test all these choices in the experiment.

Evaluation metrics. We use Inception Scores (IS) [Salimans et al., 2016] to measure the generation quality of G'. We compute IS for images conditioned on redacted and valid prompts, respectively.

Let $c \sim \mathscr{C}_\Omega$ and $z \sim \mathscr{N}$. We compute the following three metrics to evaluate redaction quality.

1. $\mathscr{R}_{G(\cdot|c/\hat{c})}$ measures faithfulness of G' on the redaction prompts. It is defined as the fraction of samples $\{G'(z|c)\}$ such that $\text{dist}(G'(z|c), G(z|\hat{c})) < \text{dist}(G'(z|c), G(z|c))$, where dist is ℓ_2 distance in the Inception-V3 feature space [Szegedy et al., 2016].

2. The R-precision score $\mathscr{R}_r$ measures how well $G'(z|c)$ matches the caption $\hat{c}$ according to a correlation metric $\text{corr}(x, c)$ between sample x and caption c [Xu et al., 2018]. Formally, $\mathscr{R}_r$ is defined as the fraction of samples $G'(z|c)$ such that $\text{corr}(G'(z|c), \hat{c})$ is larger than the correlation between $G'(z|c)$ and 100 random, mismatch captions.

3. We further introduce $\mathscr{R}_{c/\hat{c}}$, which measures how much better $G'(z|c)$ matches $\hat{c}$ than c. It is defined as the fraction of samples $G'(z|c)$ such that $\text{corr}(G'(z|c), \hat{c}) > \text{corr}(G'(z|c), c)$.

Results. The results for redacting `yellow` and `red` shown in Table 5.2. The base configuration already achieves good redaction and generation quality. After improving capacity,

Table 5.2. Generation and redaction quality after redacting `yellow` and `red`. Our method achieves significantly better redaction quality than Rewriting and retains good generation quality. The effects of each component within our method are displayed.

Method		Inception Score (↑)		Redacting quality (↑)		
		redacted	valid	$\mathscr{R}_{G(\cdot\|c/\hat{c})}$	$\mathscr{R}_{c/\hat{c}}$	$\mathscr{R}_r$
Pre-trained		4.62	5.22	0%	6.0%	13.5%
Rewriting	8×8	5.57	5.52	*33.0%*	*39.7%*	*5.0%*
	16×16	5.63	5.53	30.4%	37.2%	4.8%
	32×32	5.72	5.71	28.8%	35.9%	4.7%
	64×64	**5.77**	**5.73**	27.5%	35.2%	4.6%
Ours (base)		4.79	5.23	65.1%	77.0%	46.9%
+ improved capacity		4.74	5.25	67.8%	79.7%	**49.2%**
+ fix variance		4.79	5.35	66.5%	79.0%	48.4%
+ λ annealing		*4.84*	*5.36*	**72.2%**	**84.2%**	**49.2%**

we find all redaction quality metrics increase by $2.3 \sim 2.7\%$, and generation quality is retained. After we fix the variance prediction in H_1, the redaction decrease by $\sim 1\%$, but the generation quality on valid prompts increases by 0.1. Finally, by performing λ annealing, all metrics improve. Notably, $\mathscr{R}_{G(\cdot|c/\hat{c})}$ and $\mathscr{R}_{c/\hat{c}}$ increase by over 5%, indicating generated samples are more similar to $\hat{c}$ rather than c.

We find the Rewriting baselines achieve better IS. However, generated samples are blurred and lack sharp edges as shown in the visualization. The redaction quality of Rewriting has a significant gap with ours: all redaction metrics are less than half of ours. Especially, $\mathscr{R}_r$ is worse than the pre-trained model, indicating generated samples conditioned on redacted prompts are not very correlated to $\hat{c}$. We hypothesize the main problem for Rewriting is that it is crafted for 2D convolutions and edits the main generative network, which makes it hard to handle and distinguish the information from different prompts. In terms of different choices of resolutions, we find rewriting the layer at resolution 8×8 yields the best redaction quality.

Table 5.3 includes results for redacting the other prompts. The Rewriting baseline is applied to 8×8 resolution in H_1 because it yields the best redaction quality. We find the base configuration of our method is already very effective. Our method greatly outperforms Rewriting in all redaction quality metrics and keeps good generation quality. See visualization in Appendix

Table 5.3. Generation and redaction quality after redacting various words or phrases. Our method achieves significantly better redaction quality than Rewriting and retains good generation quality.

Redaction prompts	Method	Inception Score (↑)		Redacting quality (↑)		
		redacted	valid	$\mathscr{R}_{G(\cdot\|c/\hat{c})}$	$\mathscr{R}_{c/\hat{c}}$	$\mathscr{R}_r$
`long beak, white belly`	Pre-trained	4.14	5.61	0%	5.2%	13.1%
	Rewriting	**5.36**	**5.85**	32.6%	51.4%	5.6%
	Ours (base)	4.91	5.81	**70.5%**	**83.6%**	**50.1%**
`blue / red wings`	Pre-trained	3.97	5.48	0%	4.1%	13.1%
	Rewriting	**5.21**	**5.85**	27.8%	15.1%	6.9%
	Ours (base)	5.04	5.28	**68.6%**	**71.7%**	**58.4%**
`blue`	Pre-trained	3.65	5.18	0%	3.2%	7.2%
	Rewriting	**5.00**	**5.45**	61.8%	60.2%	17.7%
	Ours (base)	3.85	5.21	**81.3%**	**89.7%**	**66.2%**

E.2.3.

Computation. Data redaction takes about 30 minutes to train on a single NVIDIA 3080 GPU.

Robustness to adversarial prompting. In order to understand whether adversarial prompts may cause the redacted model to generate content we would like to redact, we perform an adversarial prompting attack to redacted or rewritten models in this section. Specifically, we adopt the Square Attack [Andriushchenko et al., 2020, Maus et al., 2023] directly to the discrete text space. For $c \in \mathscr{C}_\Omega$, the goal is to find an adversarial conditional c_{adv} such that $\text{corr}(G'(z|c_{\text{adv}}), c) > \text{corr}(G'(z|c_{\text{adv}}), \hat{c})$. The algorithm is shown in Algorithm 4 and some examples of successful attacks are shown in Appendix E.2.4. Success rates of the proposed attack are shown in Table 5.4. The success rates for our redaction method is consistently lower than the Rewriting baseline (by $31\% \sim 45\%$), indicating our method is considerably more robust to adversarial prompting attacks than Rewriting.

5.4.3 Redacting Diffusion-based Text-to-Speech Models

Setup. We use the pre-trained DiffWave model [Kong et al., 2021] trained on the LJSpeech dataset [Ito, 2017], which contains 13100 utterances from a female speaker reading

Algorithm 4. Adversarial Prompting via Square Attack [Andriushchenko et al., 2020, Maus et al., 2023]

```
1: Initialize $c_{\text{adv}} = c$.
2: for iteration = $1, \cdots, 16$ do
3:     Uniformly sample a position $s$ of the caption $c_{\text{adv}}$ to update.
4:     Uniformly sample 32 candidate words from the token dictionary. Construct 32 candidate
       adversarial captions by replacing the $s$-th token of $c_{\text{adv}}$ with these words, respectively.
5:     Update the adversarial caption $c_{\text{adv}}$ with the one with the largest $\text{sim}(G'(z|c_{\text{adv}}), c)$.
6: end for
7: return $c_{\text{adv}}$
```

Table 5.4. Success rates of the adversarial prompting attack (Algorithm 4) to our redaction method and the Rewriting baseline. Our redaction method is more robust to such attacks than Rewriting.

Redaction prompts	Method	Attack Success Rate ($\downarrow$)
`long beak, white belly`	Rewriting	92.8%
	Ours (base)	**50.3%**
`blue / red wings`	Rewriting	97.4%
	Ours (base)	**65.7%**
`blue`	Rewriting	81.1%
	Ours (base)	**35.5%**
`yellow, red`	Rewriting	95.5%
	Ours (base)	**59.9%**

books in home environment. The model is conditioned on Mel-spectrogram. We redact unseen voices from the disjoint LibriTTS dataset [Zen et al., 2019]. We randomly choose five voices to redact: speakers `125, 1578, 1737, 1926` (female's voice) and `1040` (men's voice). The training set for each voice has total lengths between 4 and 6 minutes. Our distillation algorithm is trained with the spectrogram data only. The CycleGAN-VC2 is trained with $\sim 1\%$ of training utterances from LJSpeech ($\sim$ 11 minutes) and all training samples from each LibriTTS voice. More details are in Appendix E.3.

Configurations. We first use the uniform parallel distillation loss in (5.5) with $\lambda = 1.5$. We fix all up-sampling layers and denote it as the base configuration. We then use the spectrogram-rewriting module in (5.6) to improve capacity. Next, we improve voice cloning with Whisper and Tortoise-TTS when training CycleGAN-VC2. Finally, we investigate non-uniform distillation losses shown in (5.7) and Table 5.1, where we set $\alpha = 0.001$ and $\beta = 0.01$ so that all w_i's or λ_i's have the same order or magnitude.

Evaluation metrics. To evaluate generation quality on the training voice $\mathscr{C} \setminus \mathscr{C}_\Omega$, we compute the following two speech quality metrics on the test set of LJSpeech: Perceptual Evaluation of Speech Quality (PESQ) [Recommendation, 2001] and Short-Time Objective Intelligibility (STOI) [Taal et al., 2011]. To evaluate redaction quality, we train a speaker classifier between redacted and training voices in each experiment. We extract Mel-frequency cepstral coefficients [Xu et al., 2005], spectral contrast [Jiang et al., 2002], and chroma features [Ellis, 2007] as sample features and train a support vector classifier. We then compute the recall rate of redacted voices after we perform redaction. In contrast to the standard classification, a lower recall rate means a higher fraction of redacted voices are projected to the training voice by the edited model, which indicates better redaction quality. See Appendix E.3.3 for details of these metrics.

Results. The results for redacting speaker `1040` are shown in Table 5.5. With the base configuration we can redact a fraction of conditionals but the generation quality is much worse than the pre-trained model. By improving capacity both generation and redaction quality are improved. Improved voice cloning does not increase the quantitative metrics, but we find the generation quality is perceptually slightly better. The non-uniform distillation losses have a huge impact on the results. The λ_i-order and λ_i-dilation schedules can boost generation quality by a large gap without compensating redaction quality too much. The w_i-order and w_i-dilation schedules can improve redaction quality while keeping the generation quality. As high generation quality is very important for speech synthesis (on non-redacted voices), the λ_i-order schedule leads to the best overall performance.

The results for redacting other speakers are shown in Table 5.6. In most settings the improved capacity configuration leads to much better generation quality than the base configuration with very little compensation for redaction quality, except for speaker `1926` where results are similar.

Table 5.5. Results of generation and redaction quality for redacting the man speaker `1040` in LibriTTS. The λ_i-order schedule in the non-uniform distillation losses leads to the best overall performance. The effects of each component within our method are displayed.

Method		Speech quality (LJSpeech) PESQ (↑)	STOI (↑)	Recall ($\mathscr{C}_\Omega$) (↓)
Pre-trained		3.33	97.8%	-
base		2.85	95.7%	52%
+ improved capacity		3.03	96.6%	35%
+ improved voice cloning		3.02	96.6%	35%
+ non-uniform	λ_i-order	**3.23**	**97.4%**	40%
	λ_i-dilation	3.21	**97.4%**	50%
	w_i-order	3.02	96.6%	**29%**
	w_i-dilation	3.02	96.6%	30%

Table 5.6. Results of generation and redaction quality for redacting several female speakers in LibriTTS. The improved capacity configuration leads to the best overall performance in most settings, with an exception for speaker `1926` where both configurations lead to similar performance.

Redaction voices	Method	Speech quality (LJSpeech) PESQ (↑)	STOI (↑)	Recall ($\mathscr{C}_\Omega$) (↓)
	Pre-trained	3.33	97.8%	-
`speaker 125`	base	3.14	97.0%	**0%**
	+ improved capacity	**3.27**	**97.4%**	3%
`speaker 1578`	base	2.14	94.4%	**1%**
	+ improved capacity	**3.24**	**97.4%**	3%
`speaker 1737`	base	2.49	94.9%	**4%**
	+ improved capacity	**3.24**	**97.2%**	9%
`speaker 1926`	base	**3.06**	96.3%	**16%**
	+ improved capacity	3.04	**96.6%**	**16%**

Computation. On a single NVIDIA 3080 GPU, it takes less than 60 minutes to distill with the base configuration, and around 100 minutes with the other configurations. It takes around 2 hours to train the CycleGAN-VC2 model. As a comparison, DiffWave takes days to train on 8 GPUs.

Demo. We include audio samples in our demo website: https://dataredact2023.github.io/.

5.5 Conclusion

In this paper, we introduce a formal framework for redacting data from conditional generative models, and present a computationally efficient method that only involves the conditioning networks. We introduce explicit formula for simple models, and propose distillation-based methods for practical conditional models. Empirically, our method performs well for practical text-to-image/speech models. It is computationally efficient, and can effectively redact certain conditionals while retaining high generation quality. For redacting prompts in text-to-image models, our method redacts better and is considerably more robust than the baseline methods. For redacting voices in text-to-speech models, our method can redact both similar and different voices while retaining high speech quality and intelligibility. One important future direction is to further improve robustness against adversarial attacks. Another line of future work is to apply the proposed method to Transformer-based architectures, where the conditioning networks are based on cross-attention blocks.

5.6 Acknowledgements

This work was supported by NSF under CNS 1804829 and ARO MURI W911NF2110317.

This chapter is a reformatted version of the paper ”Data Redaction from Conditional Generative Models” by Zhifeng Kong and Kamalika Chaudhuri [Kong and Chaudhuri, 2023a]. The dissertation author was the primary researcher and author of this paper.

Chapter 6

Approximate Data Deletion in Generative Models

6.1 Introduction

Machine learning has proved to be an increasingly powerful tool. With this power comes responsibility and there are growing concerns in academia, government, and the private sector about user privacy and responsible data management. Recent regulations (e.g., GDPR and CCPA) have introduced a *right to erasure* whereby a user may request that their data is deleted from a database. While deleting user data from database is straightforward, a savvy attacker might still be able to reverse-engineer the data by examining a machine learning model trained on it [Balle et al., 2021]. Re-training a model (after deleting the requested data) is computationally expensive, especially for modern deep learning methods [Karras et al., 2020]. This has motivated *machine unlearning* [Cao and Yang, 2015] where learned models are altered (in a computationally cheap way) to emulate the re-training process. In this chapter we focus on machine unlearning for generative models, a class of unsupervised learning methods which learn the probability distribution from data.

Prior work in supervised learning proposed *approximate data deletion* to approximate the re-trained model without actually performing the re-training [Guo et al., 2019, Neel et al., 2021, Sekhari et al., 2021, Izzo et al., 2021]. While these methods have achieved great success, approximate data deletion for *un*supervised learning largely remains an open question. In this chapter we present a density-ratio-based framework for approximate deletion in generative models. We present two novel contributions:

1. We propose a fast method for approximate data deletion for generative models.
2. We provide statistical tests to estimate whether training data have been deleted from a generative model given only sampling access to it.

For both contributions, we provide theoretical guarantees under a variety of learner assumptions. We also perform empirical investigations on real and synthetic datasets. In particular, our fast deletion algorithm is $> 10\times$ more efficient than re-training on real datasets.

The supervised and unsupervised settings have two major differences in the context of data deletion. The first is the definition – what does it mean to effectively delete training data? In the supervised setting, it is the classification function approximates the re-trained one, while in generative models it is to approximate the re-trained generative distribution. The other difference is the user's capability when evaluating data deletion. In the supervised setting, one can construct an input sample and query its predicted target. In contrast, a user can only draw samples from a generative model and then investigate the empirical distribution to evaluate the effectiveness of approximate data deletion.

In Section 6.2 we present our density-ratio-based framework and provide theoretical guarantee under various learner assumptions. We introduce practical algorithms for approximate deletion (our first primary contribution) in Section 6.3. We then study statistical tests with sampling access (our second primary contribution) in Section 6.4. We perform empirical investigations (Section 6.5) on real and synthetic data for both our fast deletion method and statistical test. We discuss related work in Section 6.6 and conclude with a discussion of future work in Section 6.7.

$$\begin{tikzcd}
p_* \arrow[r, "\text{i.i.d.}"] \arrow[dd, "\times\rho_*"'] & X \arrow[r, "\mathscr{A}(\cdot)"] \arrow[dd, "-X'"'] \arrow[dr, "\text{DRE}"'] & \hat{p} \arrow[r, "\times\hat{\rho}_{\mathscr{E}}"] & \mathscr{D}(\hat{p},X,X') \arrow[dd] \\
 & & \hat{\rho}_{\mathscr{E}} \arrow[ur] & \text{goal: } \approx \\
p'_* \arrow[r, "\text{i.i.d.}"] & X\setminus X' \arrow[ur] \arrow[rr, "\mathscr{A}(\cdot)"] & & \hat{p}'
\end{tikzcd}$$

Figure 6.1. Our density-ratio-based framework for approximated data deletion for a generative learning algorithm $\mathscr{A}(\cdot)$. We train a DRE $\hat{\rho}_{\mathscr{E}}$ between X and $X \setminus X'$. We then multiply $\hat{\rho}_{\mathscr{E}}$ to $\hat{p}$ to obtain $\mathscr{D}(\hat{p},X,X')$. The goal is to let $\mathscr{D}(\hat{p},X,X')$ approximate $\hat{p}'$.

6.2 A Density Ratio Based Framework

Let p_* be a distribution over $\mathbb{R}^d$ and X be N i.i.d. samples from p_*. We consider a generative learning algorithm $\mathscr{A}$ which aims to model p_*. We denote the distribution $\mathscr{A}$ learns from X as $p_{\mathscr{A}(X)}$, and we refer to $\hat{p} = p_{\mathscr{A}(X)}$ as the pre-trained model. Let $X' \subset X$ be N' samples we would like to delete from $\hat{p}$, and $\hat{p}' = p_{\mathscr{A}(X \setminus X')}$ be the ground-truth re-trained model. A notation table is provided in Appendix F.1. In this chapter, we present solutions to two problems:

1. Fast deletion: given $\hat{p}$, approximate $\hat{p}'$ more efficiently than full re-training.

2. Deletion test: assuming $q \in \{\hat{p}, \hat{p}'\}$, test whether $q = \hat{p}$ or $q = \hat{p}'$ by drawing samples.

6.2.1 Framework

We propose a density-ratio-based framework to perform fast (approximate) deletion and our deletion test. The density ratio between two distributions μ_1 and μ_2 on $\mathbb{R}^d$ is defined as $\rho(\mu_1,\mu_2) : \mathbb{R}^d \to \mathbb{R}^+, x \mapsto \mu_2(x)/\mu_1(x)$[1]. Let $\hat{\rho} = \rho(\hat{p},\hat{p}')$ be the density ratio between pre-trained and re-trained models. In our proposed framework, we learn a density ratio estimator (DRE) $\hat{\rho}_{\mathscr{E}} = \text{DRE}(X, X \setminus X')$ between X and $X \setminus X'$ to approximate $\hat{\rho}$. Then, to perform fast deletion we define the approximated model $\mathscr{D}(\hat{p},X,X') : \mathbb{R}^d \to \mathbb{R}^+, x \mapsto \hat{\rho}_{\mathscr{E}}(x) \cdot \hat{p}(x)$, which we abbreviate as $\hat{\rho}_{\mathscr{E}} \cdot \hat{p}$ for conciseness.

[1]We choose this order for cleaner theory.

Algorithm 5. Sampling from the approximated model

```
1: Inputs: $\hat{p}$, $\hat{\rho}_{\mathscr{E}}$.
2: while True do
3:     Sample $y \sim \hat{p}$ and $u \sim \texttt{Uniform}([0,1])$.
4:     if $\hat{\rho}_{\mathscr{E}}(y) > B \cdot u$ then
5:         return $y$
6:     end if
7: end while
```

Core to both our method of fast deletion and our deletion test is our DRE based framework (summarized in Fig. 6.1). We model $X \setminus X'$ to be a set of i.i.d. samples from some distribution we denote as p'_*, and define $\rho_* = \rho(p_*, p'_*)$. We assume $\|\rho_*\|_\infty \leq \infty$. Intuitively, deleting some samples from p_* will only increase likelihood of regions far from these samples by at most a constant factor, and reduce likelihood of regions around these samples. We also assume $N' \ll N$ – only a small fraction of training samples are to be deleted. Intuitively, the pre-trained and re-trained models are likely similar. This allows us to provide approximation bounds for consistent learning algorithms $\mathscr{A}$. In Section 6.2.2 we derive such bounds for various forms of consistency.

In the supervised setting, approximate deletion can be done by altering the pre-trained model to be closer to the (never computed) re-trained model. In contrast, we alter the process of sampling from the unsupervised pre-trained model to simulate sampling from the re-trained model. Drawing samples from the approximated model is done in two steps: first draw samples from $\hat{p}$, and then perform rejection sampling according to $\hat{\rho}_{\mathscr{E}}$. Note that this procedure requires there exists a known constant $B \geq \|\hat{\rho}_{\mathscr{E}}\|_\infty$, which we discuss further in Section 6.2.3. We present this procedure in Alg. 5.

6.2.2 Approximation under Consistency

A learning algorithm $\mathscr{A}$ is said to be *consistent* if $p_{\mathscr{A}(X)}$ converges to p_* as $N \to \infty$ [Wied and Weißbach, 2012], where each specific type of convergence leads to a specific definition of consistency. If $\mathscr{A}$ is consistent, then we have $\hat{p} \approx p_*$ and $\hat{p}' \approx p'_*$ for large N. In this section, we derive DREs for two kinds of consistency to achieve approximated deletion: $\hat{\rho}_{\mathscr{E}}$ such that the approximated model $\mathscr{D}(\hat{p},X,X') := \hat{\rho}_{\mathscr{E}} \cdot \hat{p} \approx \hat{p}'$.

In **Def.** 6.2.1, we introduce ratio consistency, which bounds the density ratio between true and learned distributions. We show in **Thm.** 6.2.2 that approximation in L_1 distance can be achieved in this case.

Definition 6.2.1 (Ratio Consistent (RC))**.** *$\mathscr{A}$ is (c_N,δ_N)-RC if for any density μ, with probability at least $1-\delta_N$, it holds that $\|\log\rho(p_{\mathscr{A}(Z)},\mu)\|_\infty \leq \log c_N$, where Z contains N i.i.d. samples from μ, and $c_N \to 1$, $\delta_N \to 0$ as $N \to \infty$.*

Theorem 6.2.2 (Approximation under RC)**.** *If $\mathscr{A}$ is (c_N,δ_N)-RC, then there exists a DRE $\hat{\rho}_{\mathscr{E}}$ such that with probability at least $1-2(\delta_N+\delta_{N-N'})$, it holds that $\|\hat{\rho}_{\mathscr{E}} \cdot \hat{p} - \hat{p}'\|_1 \leq 4(c_N + c_{N-N'} - 2)$.*

We then look at a more practical total variation consistency in **Def.** 6.2.3, which bounds the total variation distance (half of L_1 distance) between true and learned distributions. We show in **Thm.** 6.2.4 that approximation in expectation is achieved in this case.

Definition 6.2.3 (Total Variation Consistent (TVC))**.** *$\mathscr{A}$ is (ε_N,δ_N)-TVC if for any density μ, with probability at least $1-\delta_N$, it holds that $\|p_{\mathscr{A}(Z)}-\mu\|_1 \leq \varepsilon_N$, where Z contains N i.i.d. samples from μ, and $\varepsilon_N \to 0$, $\delta_N \to 0$ as $N \to \infty$.*

Theorem 6.2.4 (Approximation under TVC)**.** *Define $\|h\|_{1,\mu} = \int_x \mu(x)|h(x)|dx$. If $\mathscr{A}$ is (ε_N,δ_N)-TVC, then there exists a DRE $\hat{\rho}_{\mathscr{E}}$ such that with probability at least $1-2(\delta_N+\delta_{N-N'})$, it holds that $\|\hat{\rho}_{\mathscr{E}} \cdot \hat{p} - \hat{p}'\|_{1,\hat{p}} \leq 2(\varepsilon_{N-N'} + \|\rho_*\|_\infty \varepsilon_N)$.*

We prove these theorems by construction. Full proofs are provided in Appendix F.2.1. For each, the high level idea is to choose a fixed consistent algorithm $\mathscr{A}_0$, and define $\hat{\rho}_{\mathscr{E}}(Z_1,Z_2) = \rho(p_{\mathscr{A}_0(Z_1)}, p_{\mathscr{A}_0(Z_2)})$. This yields $\hat{\rho}_{\mathscr{E}}(X, X\setminus X') \approx \rho_* \approx \hat{\rho}$ and therefore $\mathscr{D}(\hat{p},X,X') = \hat{\rho}_{\mathscr{E}} \cdot \hat{p} \approx \hat{p}'$. We briefly summarize the results in Table 6.1.

6.2.3 Practicability under Stability

In practice, we need $\|\hat{\rho}_{\mathscr{E}}\|_\infty$ to be finite in order to perform rejection sampling in Alg. 5 (see Line 3). Under this constraint, to satisfy $\hat{\rho}_{\mathscr{E}} \approx \hat{\rho}$, we need $\|\hat{\rho}\|_\infty$ to be finite as a prerequisite. In this section, we study several stability conditions of the learning algorithm $\mathscr{A}$ that guarantee $\|\hat{\rho}\|_\infty$ to be finite.

We organize these stability conditions in the order from *strong* to *weak*. We first discuss several strong, classic stability conditions that guarantee $\|\hat{\rho}\|_\infty$ to be small (**Def.** 6.2.5 – 6.2.7, **Thm.** 6.2.8). To state our definitions, let Z ($\hat{Z}$) be any training (test) set and z ($\hat{z}$) be any sample in Z ($\hat{Z}$).

Definition 6.2.5 (Differentially Private (DP) [Dwork et al., 2006])**.** *$\mathscr{A}$ is ε-DP if* $|\log p_{\mathscr{A}(Z\setminus\{z\})}(\hat{Z}) - \log p_{\mathscr{A}(Z)}(\hat{Z})| \leq \varepsilon$.

Definition 6.2.6 (Uniformly Stable (US) [Bousquet and Elisseeff, 2002])**.** *$\mathscr{A}$ is ε-US if* $|\log p_{\mathscr{A}(Z\setminus\{z\})}(\hat{z}) - \log p_{\mathscr{A}(Z)}(\hat{z})| \leq \varepsilon$.

Definition 6.2.7 (Lower Bounded in Likelihood Influence (LBLI) [Kong and Chaudhuri, 2021a])**.** *$\mathscr{A}$ is ε-LBLI if $p_{\mathscr{A}(Z\setminus\{z\})}(\hat{z}) \leq e^{\varepsilon} p_{\mathscr{A}(Z)}(\hat{z})$.*

We discuss relationship among DP, US and LBLI algorithms below. Note that ε-DP implies ε-US and ε-US implies ε-LBLI. If $\mathscr{A}$ is ε-DP or ε-US, the re-trained model satisfies $\hat{p}' \approx \hat{p}$, and there is no need to perform deletion. If $\mathscr{A}$ is ε-LBLI but not ε-US, then there exists a sample $\hat{z}$ such that $\hat{p}'(\hat{z}) \ll \hat{p}(\hat{z})$. Intuitively, in non-parametric methods, $\hat{z}$ can be samples near X'. ε-LBLI can be achieved under some regulatory assumptions on the loss function and the Hessian matrix with respect to parameters [Giordano et al., 2019a,b, Basu et al., 2020]. Then, we have the following result.

Theorem 6.2.8. *If $\mathscr{A}$ is ε-DP, ε-US, or ε-LBLI, then* $\log\|\hat{\rho}\|_\infty \leq N'\varepsilon$.

Next, we move on to weaker stability assumptions to the learner. We introduce ratio stability, a concept crafted for our framework (**Def.** 6.2.9), which bounds the difference between log

Table 6.1. High-level summary of approximation results between the approximated model $\mathscr{D}(\hat{p}, X, X')$ and the re-trained model $\hat{p}'$ under different consistency assumptions to the learner $\mathscr{A}$.

Assumption	Approximation Result
RC (**Def.** 6.2.1)	in $\|\cdot\|_1$ with high probability
TVC (**Def.** 6.2.3)	in $\|\cdot\|_{1,\hat{p}}$ with high probability

density ratios of true and learned distributions. We discuss its connection with ratio consistency (**Thm.** 6.2.10), and bound the difference between $\|\hat{\rho}\|_\infty$ and $\|\rho_*\|_\infty$ (**Thm.** 6.2.11). Finally, we discuss a special type of error stability [Bousquet and Elisseeff, 2002] (**Def.** 6.2.12) and show a concentration bound on $\hat{\rho}$ (**Thm.** 6.2.13).

Definition 6.2.9 (Ratio Stable (RS))**.** *$\mathscr{A}$ is (ε, δ)-RS if for any densities μ_1, μ_2 such that $\sup_x \mu_2(x)/\mu_1(x) < \infty$, with probability at least $1-\delta$, when i.i.d. samples $Z_i \sim \mu_i$ satisfy $|Z_1| = |Z_2| + 1$, it holds that $\|\log \rho(\mu_1, p_{\mathscr{A}(Z_1)}) - \log \rho(\mu_2, p_{\mathscr{A}(Z_2)})\|_\infty \leq \varepsilon$.*

Theorem 6.2.10. *If $\mathscr{A}$ is (c_N, δ_N)-RC, then $\mathscr{A}$ is $(2\log c_N, 2\delta_N)$-RS.*

Theorem 6.2.11. *If $\mathscr{A}$ is (ε, δ)-RS, then with probability at least $1 - N'\delta$, it holds that $\log \|\hat{\rho}\|_\infty \leq N'\varepsilon + \log \|\rho_*\|_\infty$.*

Definition 6.2.12 (Error Stable (ES) [Bousquet and Elisseeff, 2002])**.** *$\mathscr{A}$ is (ε, k)-ES if $|\mathbb{E}_{\hat{z} \sim p_{\mathscr{A}(Z)}} \left[\log \left(p_{\mathscr{A}(Z \setminus \{z\})}(\hat{z}) / p_{\mathscr{A}(Z)}(\hat{z})\right)\right]^k| < \varepsilon$.*

Theorem 6.2.13. *Let $N' = 1$. If $\mathscr{A}$ is $(\varepsilon, 2)$-ES, then with probability at least $1-\delta$, it holds that $\log \hat{\rho}(x) \leq \sqrt{\varepsilon(1-\delta)/\delta}$ for $x \sim \hat{p}$.*

We prove these theorems by induction and central inequalities. See Appendix F.2.2 for proofs. We briefly summarize the results in Table 6.2.

6.3 Density Ratio Estimators for Fast Data Deletion

A key step in the proposed framework is to train a density ratio estimator (DRE) $\hat{\rho}_{\mathscr{E}}$ between X and $X \setminus X'$. There is a rich literature of DRE techniques [Sugiyama et al., 2012,

Table 6.2. High-level summary of bounds on $\log \|\hat{\rho}\|_\infty$ under different stability assumptions to the learner $\mathscr{A}$.

Assumption	Bound on $\log \|\hat{\rho}\|_\infty$
DP (**Def.** 6.2.5)	$\mathscr{O}(\varepsilon)$
US (**Def.** 6.2.6)	$\mathscr{O}(\varepsilon)$
LBLI (**Def.** 6.2.7)	$\mathscr{O}(\varepsilon)$
RS (**Def.** 6.2.9)	$\text{const} + \mathscr{O}(\varepsilon)$ with high probability.
ES (**Def.** 6.2.12)	$\mathscr{O}\left(\sqrt{\varepsilon/\delta}\right)$ with probability $1-\delta$.

Nowozin et al., 2016, Moustakides and Basioti, 2019, Khan et al., 2019, Rhodes et al., 2020, Kato and Teshima, 2021, Choi et al., 2021, 2022]. All of these methods are designed for settings with little or no prior information about the data. We leverage the strong prior information that one set ($X \setminus X'$) is a strict subset of the other (X) to design more focused DRE methods for our data deletion setting. In Section 6.3.1, we derive a simple DRE based on probabilistic classification, and compare it with standard methods [Sugiyama et al., 2012]. In Section 6.3.2, we use variational divergence minimization [Nowozin et al., 2016] to train a DRE that can handle high dimensional real-world datasets.

6.3.1 Probabilistic Classification

We derive a simple DRE based on probabilistic classification [Sugiyama et al., 2012]. Let f be a classifier on $\{X \setminus X', X'\}$, where $f(x) = \text{Prob}(x \in X')$. Let `nu` be the event that $X \setminus X'$ is used to train the model, and `de` be the event that X is used to train the model. We apply Bayes rule as follows:

$$\begin{aligned}
\hat{\rho}_{\mathscr{E}}(x) &= \frac{\text{Prob}(x|\texttt{nu})}{\text{Prob}(x|\texttt{de})} = \frac{\text{Prob}(\texttt{nu}|x)/\text{Prob}(\texttt{nu})}{\text{Prob}(\texttt{de}|x)/\text{Prob}(\texttt{de})} \\
&= \frac{N}{N-N'} \cdot \frac{\frac{1}{2}\text{Prob}(x \in X \setminus X')}{\text{Prob}(x \in X') + \frac{1}{2}\text{Prob}(x \in X \setminus X')} \\
&= \frac{N}{N-N'} \cdot \frac{\frac{1}{2}(1-f(x))}{f(x) + \frac{1}{2}(1-f(x))} \\
&= \frac{N}{N-N'} \cdot \frac{1-f(x)}{1+f(x)}.
\end{aligned}$$

As an example, consider Kernel Density Estimation (KDE) [Rosenblatt, 1956, Parzen, 1962], a class of consistent algorithms which learn an explicit probability density.

Example 6.3.1 (KDE)**.** *Let $\mathscr{A}$ be KDE with Gaussian kernel function $K_\sigma(x) = \mathscr{N}(x; 0, \sigma^2 I)$. Then, The following classifier f exactly recovers $\hat{\rho}_{\mathscr{E}} = \hat{\rho}$:*

$$f(x) = \frac{\sum_{i=1}^{N'} K_\sigma(x - x_i)}{\left(\sum_{i=1}^{N'} + 2\sum_{i=N'+1}^{N}\right) K_\sigma(x - x_i)}. \tag{6.1}$$

Example 6.3.1 indicates that we need to up-weight samples in $X \setminus X'$ in the classifier, in addition to standard methods [Sugiyama et al., 2012]. This observation is universal as $1 - f(x)$ is shared by both cases ($x \in X$ and $x \in X \setminus X'$) when we compute DRE.

6.3.2 Variational Divergence Minimization

Note that KDE and classification-based DRE are especially amenable to our methods but may not be able to deal with complicated, high-dimensional datasets [Choi et al., 2022]. Now, we consider the learner to be a Generative Adversarial Network (GAN) [Goodfellow et al., 2014], a class of powerful implicit deep generative models. For these models, we derive a DRE based on variational divergence minimization (VDM) [Nowozin et al., 2016]. Because neural networks can have large capacity and VDM is designed to distinguish distributions, VDM-based DRE is more applicable with complicated data such as images compared to classification-based DRE. We begin with the definition of ϕ-divergence below.

Definition 6.3.2 ([Liese and Vajda, 2006])**.** *Let $\phi : [0, \infty) \to \mathbb{R}$ be a strictly convex function such that $\phi(x)$ is finite for $x > 0$, $\phi(1) = 0$ and $\phi(0) = \lim_{x \to 0^+} \phi(x)$. The ϕ-divergence between distributions μ and ν is defined as $D_\phi(\mu \| \nu) = \int_x \nu(x)\phi\left[\mu(x)/\nu(x)\right] dx$.*

$D_\phi(p'_*||p_*)$ satisfies the following variational bound [Nguyen et al., 2010]:

$$D_\phi(p'_*||p_*) \geq \sup_T \left(\mathbb{E}_{x \sim p'_*} T(x) - \mathbb{E}_{x \sim p_*} \phi^*(T(x))\right), \tag{6.2}$$

where ϕ^* is the conjugate function of ϕ defined as $\phi^*(t) := \sup_u(ut - \phi(u))$. The optimal T is $T(x) = \frac{d}{dt}\phi(p'_*(x)/p_*(x)) = \frac{d}{dt}\phi(\rho_*(x))$, and in this case (6.2) achieves equality. Then, VDM is optimizing the right-hand-side of (6.2), usually via a neural network. Once the optimal T is obtained, we can solve $\hat{\rho}_{\mathscr{E}} = (\frac{d}{dt}\phi)^{-1}(T)$.

To perform the actual training in practice, we optimize the empirical version of the lower bound (6.2) based on the i.i.d. assumptions on X and $X \setminus X'$.

$$T_\phi = \arg\max_T \ \mathbb{E}_{x\sim X\setminus X'} T(x) - \mathbb{E}_{x\sim X}\phi^*(T(x)), \tag{6.3}$$

We provide specific algorithms to train DRE for two ϕ-divergences below. In both examples, T is a neural network.

Example 6.3.3 (Jensen-Shannon)**.** *Let D_ϕ be the Jason-Shannon divergence. With an additional* $\log(\cdot)$ *term, we recover the discriminator loss in GAN [Goodfellow et al., 2014]:*

$$T_\phi = \arg\max_T \ \mathbb{E}_{x\sim X\setminus X'} \log T(x) + \mathbb{E}_{x\sim X} \log(1 - T(x)). \tag{6.4}$$

In this case, the estimated density ratio is $\hat{\rho}_\phi = T_\phi/(1 - T_\phi)$.

Example 6.3.4 (Kullback–Leibler)**.** *Let D_ϕ be the KL divergence. Then, we recover the discriminator loss in KL-GAN [Liu and Chaudhuri, 2018]:*

$$T_\phi = \arg\max_T \ \mathbb{E}_{x\sim X\setminus X'} T(x) - \mathbb{E}_{x\sim X} e^{T(x)}. \tag{6.5}$$

In this case, the estimated density ratio is $\hat{\rho}_\phi = \exp(T_\phi - 1)$.

Note that given enough capacity and data, we have $\hat{\rho}_\phi \approx \rho_*$ rather than $\hat{\rho}$, which may cause some bias. This bias can be alleviated when the learner $\mathscr{A}$ is consistent and expressive enough, such as GAN [Liu et al., 2021]. We find KL divergence in Example 6.3.4 works well in practice.

6.4 Statistical Tests for Data Deletion

Our second main contribution is our statistical deletion tests to distinguish whether a generative model has has particular points deleted. Formally, we assume sample access to a distribution q, which is either the pre-trained model $\hat{p}$ or the re-trained model $\hat{p}'$. We consider the following hypothesis test: $\mathscr{H}_0 : q = \hat{p}$; $\mathscr{H}_1 : q = \hat{p}'$.[2] Several statistics for this test (not in the data deletion setting) have been proposed, including likelihood ratio (LR) [Neyman and Pearson, 1933], ASC statistics [Kanamori et al., 2011], and maximum mean discrepancy (MMD) [Gretton et al., 2012]. In this section, we adapt LR and ASC to the data deletion setting, and discuss MMD in Appendix F.3.3. In practice, we may not know $\hat{p}'$, so we use $\mathscr{H}_1' : q = \mathscr{D}(\hat{p}, X, X')$ to approximate $\mathscr{H}_1$. We present theory on the approximation between $\mathscr{H}_1$ and $\mathscr{H}_1'$ when these statistics are used, thus providing an efficient way to test $\mathscr{H}_0$ vs $\mathscr{H}_1$ without re-training.[3]

6.4.1 Likelihood Ratio

A common goodness-of-fit method is the likelihood ratio test. In terms of having the smallest type-2 error, the likelihood ratio test is the most powerful of statistical tests [Neyman and Pearson, 1933] and is performed as follows. Given m samples $Y \sim q$, the likelihood ratio statistic is defined as

$$\mathrm{LR}(Y, \hat{p}, \hat{p}') = \frac{1}{m} \sum_{y \in Y} \log \frac{\hat{p}'(y)}{\hat{p}(y)} = \frac{1}{m} \sum_{y \in Y} \log \hat{\rho}(y).$$

As it is solely determined by Y and $\hat{\rho}$, we abbreviate it as $\mathrm{LR}(Y, \hat{\rho})$. When we use $\mathscr{H}_1'$ to approximate $\mathscr{H}_1$ in practice, we compute $\mathrm{LR}(Y, \hat{\rho}_{\mathscr{E}})$. In **Thm.** 6.4.1, we prove it approximates $\mathrm{LR}(Y, \hat{\rho})$ with high probability under RC (**Def.** 6.2.1), and in **Thm.** 6.4.2, we show approximation when $\hat{\rho}_{\mathscr{E}}$ is close to $\hat{\rho}$. Statistical properties of likelihood ratio and proofs to the above theorems are in Appendix F.3.1.

[2]This is different from two-sample tests, where $\mathscr{H}_1$ is $q \neq \hat{p}$, and we do not have knowledge of $\hat{p}'$.
[3]It is unclear how to test $\mathscr{H}_0$ vs $\mathscr{H}_1$ even with re-training if $\mathscr{A}$ does not yield explicit likelihood (e.g., GAN).

Theorem 6.4.1. *If $\mathscr{A}$ is (c_N, δ_N)-RC, then there exists a $\hat{\rho}_{\mathscr{E}}$ such that with probability at least $1-2(\delta_N + \delta_{N-N'})$, it holds that $|\mathrm{LR}(Y, \hat{\rho}) - \mathrm{LR}(Y, \hat{\rho}_{\mathscr{E}})| \leq 2(\log c_N + \log c_{N-N'})$.*

Theorem 6.4.2. *(1) If $\| \log \hat{\rho} - \log \hat{\rho}_{\mathscr{E}} \|_\infty \leq \varepsilon$, then $|\mathrm{LR}(Y, \hat{\rho}) - \mathrm{LR}(Y, \hat{\rho}_{\mathscr{E}})| \leq \varepsilon$.*
(2) If $\max(\| \log \hat{\rho} - \log \hat{\rho}_{\mathscr{E}} \|_{1,\hat{p}}, \| \log \hat{\rho} - \log \hat{\rho}_{\mathscr{E}} \|_{1,\hat{p}'}) \leq \varepsilon$, then with probability at least $1-\delta$, it holds that $|\mathrm{LR}(Y, \hat{\rho}) - \mathrm{LR}(Y, \hat{\rho}_{\mathscr{E}})| \leq \varepsilon/\delta$.

6.4.2 ASC Statistics

ASC statistics are used to estimate the ϕ-divergence (**Def.** 6.3.2) [Kanamori et al., 2011]. Because a broad family of ϕ functions can be used, these statistics include a wide range of statistics. Drawing m samples $Y \sim q$ and another m samples $\hat{Y}$ from $\hat{p}$, the ASC statistic is defined as

$$\hat{\mathrm{ASC}}_\phi(\hat{Y}, Y, \hat{\rho}) = \frac{1}{m}\Big[\sum_{y \in \hat{Y}} + \sum_{y \in Y}\Big] \frac{\phi(\hat{\rho}(y))}{1+\hat{\rho}(y)}.$$

When we use $\mathscr{H}_1'$ to approximate $\mathscr{H}_1$ in practice, we compute $\hat{\mathrm{ASC}}_\phi(\hat{Y}, Y, \hat{\rho}_{\mathscr{E}})$. In **Thm.** 6.4.3, we show it approximates $\hat{\mathrm{ASC}}_\phi(\hat{Y}, Y, \hat{\rho})$ when $\hat{\rho}_{\mathscr{E}}$ is close to $\hat{\rho}$.

Theorem 6.4.3. *If $\max(\| \psi(\hat{\rho}) - \psi(\hat{\rho}_{\mathscr{E}}) \|_{1,\hat{p}}, \| \psi(\hat{\rho}) - \psi(\hat{\rho}_{\mathscr{E}}) \|_{1,\hat{p}'}) \leq \varepsilon$ where $\psi(t) = \phi(t)/(1+t)$, then with probability at least $1-\delta$, it holds that $|\hat{\mathrm{ASC}}_\phi(\hat{Y}, Y, \hat{\rho}) - \hat{\mathrm{ASC}}_\phi(\hat{Y}, Y, \hat{\rho}_{\mathscr{E}})| \leq 2\varepsilon/\delta$.*

Statistical properties of ASC statistics and our proof of the above theorem are in Appendix F.3.2.

6.5 Experiments

Empirically, we address the following questions. **1) DRE Approximations**: do methods in Section 6.3 produce ratios $\hat{\rho}_{\mathscr{E}}$ that approximate the target ratio $\hat{\rho}$? **2) Fast Deletion**: is $\mathscr{D}(\hat{p}, X, X') = \hat{\rho}_{\mathscr{E}} \cdot \hat{p}$ indistinguishable from the re-trained model $\hat{p}'$? And **3) Hypothesis Test**: do tests in Section 6.4 distinguish samples from pre-trained and re-trained models?

We first survey these questions in experiments on two-dimensional synthetic datasets. We then look at GANs trained on MNIST [LeCun et al., 2010] and Fashion-MNIST [Xiao et al., 2017].

6.5.1 Classification-based DRE for KDE on Synthetic Datasets

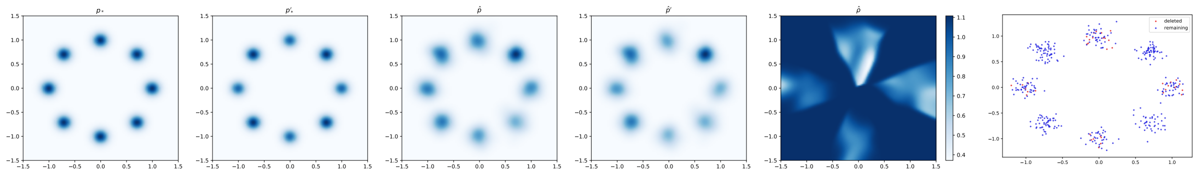
Experiment setup. We generate a synthetic distribution (p_*) over $\mathbb{R}^2$: a mixture of 8 Gaussian distributions (MoG-8) (Fig. 6.2a). We define p'_* to be a weighted mixture version of p_*: 4 re-weighted clusters have weight $= \lambda \in (0,1)$, and the other 4 have weight $= 1$ (see Fig. 6.2b). We draw $N = 400$ samples from p_* to form X, and randomly reject $1 - \lambda$ fraction of samples in re-weighted clusters to form the deletion set X' (see Fig. 6.2f). We run KDE using a Gaussian kernel and $\sigma_{\mathscr{A}} = 0.1$ to obtain pre-trained models in Fig. 6.2c, re-trained models in Fig. 6.2d, and their ratio $\hat{\rho}$ in Fig. 6.2e. We use KDE because its density is explicit and thus we are able to compute the exact likelihood ratio to examine the effectiveness of our DRE-based framework.

Method and results. We use the classification-based DRE described in Section 6.3.1. We up-weigh $X \setminus X'$ when training the classifiers according to Example 6.3.1. We consider two types of non-parametric classifiers: kernel-based classifiers (KBC) defined in (6.1) with potentially different $\sigma = \sigma_{\mathscr{C}} \neq \sigma_{\mathscr{A}}$, and k-nearest-neighbour classifiers (kNN) defined as the fraction of positive votes in k nearest neighbours.[4] For each classifier, we draw 4 sets of i.i.d. samples (each of size m):

1. $\hat{Y} \sim \hat{p}$ (pre-trained model);
2. $Y_{\mathscr{D}} \sim \hat{p} \cdot \hat{\rho}_{\mathscr{E}}$ (approximated model) marked in blue;
3. $Y_{\mathscr{H}_0} \sim (q \text{ under } \mathscr{H}_0) = \hat{p}$ marked in orange;
4. $Y_{\mathscr{H}_1} \sim (q \text{ under } \mathscr{H}_1) = \hat{p}'$ marked in green.[5]

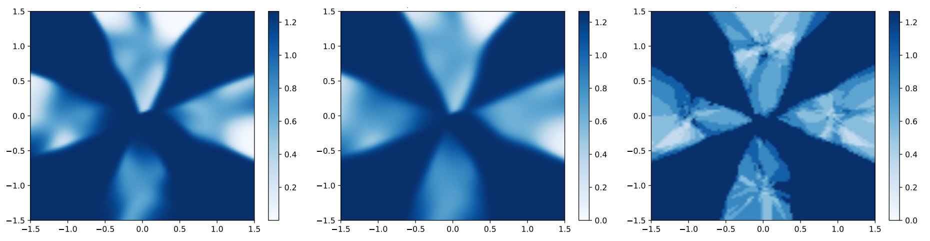
[4]We use non-parametric classifiers because the learning algorithm is non-parametric. In preliminary experiments we found parametric classifiers such as logistic regression are less effective. We conjecture this is due to imbalanced labels, but leave a further investigation as future work.

[5]The colors are used in distribution comparisons and label statistics in Fig. 6.5, Fig. 6.6, and the Appendix.



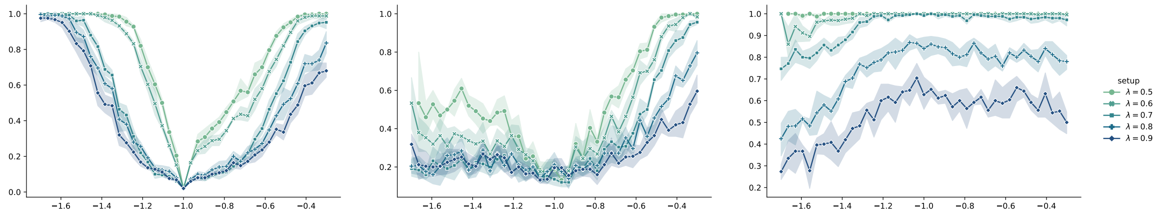
(a) p_* **(b)** p'_* **(c)** $\hat{p}$ **(d)** $\hat{p}'$ **(e)** $\hat{\rho}$ **(f)** X' and $X \setminus X'$

Figure 6.2. Visualization of the experimental setup of MoG-8. (a) Data distribution p_*. (b) Distribution p'_* with $\lambda = 0.8$. (c) Pre-trained KDE $\hat{p}$ on X with $\sigma_{\mathscr{A}} = 0.1$. (d) Re-trained KDE $\hat{p}'$ on $X \setminus X'$ with $\sigma_{\mathscr{A}} = 0.1$. (e) Density ratio $\hat{\rho} = \hat{p}'/\hat{p}$. (f) Deletion set X' and the remaining set $X \setminus X'$.



(a) $\hat{\rho}$ **(b)** KBC ($\sigma_{\mathscr{C}}$=0.125) **(c)** k-NN (k=10)

Figure 6.3. Answer to question 1: visualization of ratios in the setting of MoG-8 with $\lambda = 0.6$ and $\sigma_{\mathscr{A}} = 0.1$. (a) $\hat{\rho}$. (b) $\hat{\rho}_{\mathscr{E}}$ for KBC-based DRE. (c) $\hat{\rho}_{\mathscr{E}}$ for kNN-based DRE. These DREs are visually close to $\hat{\rho}$.



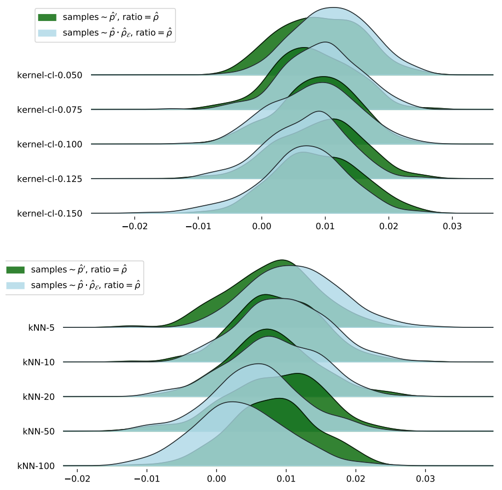
(a) $\mathrm{LR}(Y_{\mathscr{H}_0}, \hat{\rho})$ vs $\mathrm{LR}(Y_{\mathscr{H}_0}, \hat{\rho}_{\mathscr{E}})$ **(b)** $\mathrm{LR}(Y_{\mathscr{H}_1}, \hat{\rho})$ vs $\mathrm{LR}(Y_{\mathscr{D}}, \hat{\rho})$ **(c)** $\mathrm{LR}(Y_{\mathscr{H}_0}, \hat{\rho}_{\mathscr{E}})$ vs $\mathrm{LR}(Y_{\mathscr{H}_1}, \hat{\rho}_{\mathscr{E}})$

Figure 6.4. KS test results (y-axis) between distributions of LR statistics for KBC with different $\log_{10} \sigma_{\mathscr{C}}$ (x-axis). Smaller KS values (y-axis) indicate the two compared distributions are closer. Results for ASC statistics are in Appendix F.4.

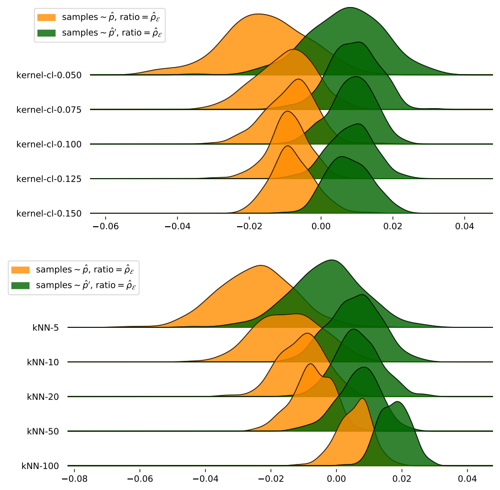
We compute LR and A$\hat{\mathrm{S}}$C statistics for each set and for both density ratios $\{\hat{\rho}, \hat{\rho}_{\mathscr{E}}\}$. The above procedure is repeated for $R = 250$ times and we report empirical distributions of these statistics.

We demonstrate results for MoG-8 with KBC-based DRE and LR statistics. More extensive experiments with k-NN classifiers, ASC statistics, and other hyper-parameters are provided in Appendix F.4.

We investigate **question 1** (DRE Approximations) in two ways. First, we compare $\hat{\rho}_{\mathscr{E}}$



(a) Answer to question 2: these distributions largely overlap with each other, indicating the approximated model cannot be distinguished from the re-trained model.



(b) Answer to question 3: these distributions are separated from each other, indicating the DRE can distinguish between samples from pre-trained and re-trained models.

Figure 6.5. Distributions of (a) $\mathrm{LR}(Y_{\mathscr{H}_1}, \hat{\rho})$ vs $\mathrm{LR}(Y_{\mathscr{D}}, \hat{\rho})$ and (b) $\mathrm{LR}(Y_{\mathscr{H}_0}, \hat{\rho}_{\mathscr{E}})$ vs $\mathrm{LR}(Y_{\mathscr{H}_1}, \hat{\rho}_{\mathscr{E}})$. The approximated models and $\hat{\rho}_{\mathscr{E}}$ are derived from KBC-based DREs with five $\sigma_{\mathscr{C}}$ values (first row) and kNN-based DREs with five k values (second row). x-axis is LR statistic and y-axis is frequency.

and $\hat{\rho}$ in Fig. 6.3. We find that both KBC and k-NN lead to good approximation. We then conduct Kolmogorov–Smirnov (KS) tests between the distributions of $\mathrm{LR}(Y_{\mathscr{H}_0}, \hat{\rho})$ vs $\mathrm{LR}(Y_{\mathscr{H}_0}, \hat{\rho}_{\mathscr{E}})$. If $\hat{\rho} \approx \hat{\rho}_{\mathscr{E}}$ on supp $\hat{p}$ then the KS statistics will be close to 0, meaning the two compared distributions are indistinguishable. In Fig. 6.4a, we plot KS statistics for KBC with different $\sigma_{\mathscr{C}}$. The KS statistics decrease as $\sigma_{\mathscr{C}}$ gets close to $\sigma_{\mathscr{A}}$. We also find larger λ (where fewer samples are deleted) leads to better estimation, as expected.

We investigate **question 2** (Fast Deletion) by asking whether the approximated model $\hat{\rho}_{\mathscr{E}} \cdot \hat{p}$ and the re-trained model $\hat{p}'$ can be distinguished by the ground truth ratio $\hat{\rho}$. We do this by comparing the distributions of $\mathrm{LR}(Y_{\mathscr{H}_1}, \hat{\rho})$ vs $\mathrm{LR}(Y_{\mathscr{D}}, \hat{\rho})$; see qualitative comparisons in Fig. 6.5a and quantitative results in Fig. 6.4b. We find for a wide range of classifiers, it is hard to distinguish between approximated and re-trained models, especially when λ is larger.

Finally, we answer **question 3** (Hypothesis Test) by comparing the distributions of $\mathrm{LR}(Y_{\mathscr{H}_0}, \hat{\rho}_{\mathscr{E}})$ vs $\mathrm{LR}(Y_{\mathscr{H}_1}, \hat{\rho}_{\mathscr{E}})$: see qualitative comparisons in Fig. 6.5b and quantitative results in Fig. 6.4c. We find $\hat{\rho}_{\mathscr{E}}$ can distinguish between samples from pre-trained and re-trained models for a wide range of classifiers. In terms of the size of the deletion set, larger λ makes the two models less distinguishable.

6.5.2 VDM-based DRE for GAN

Experimental setup. The pre-trained model is a DCGAN [Radford et al., 2015] on the full MNIST and Fashion-MNIST. We construct the deletion set X' by randomly selecting samples with certain labels (see details in Appendix F.5).

Method and results. We train VDM-based DRE based on (6.5) introduced in Section 6.3.2. We set T to be the same architecture as the discriminator.

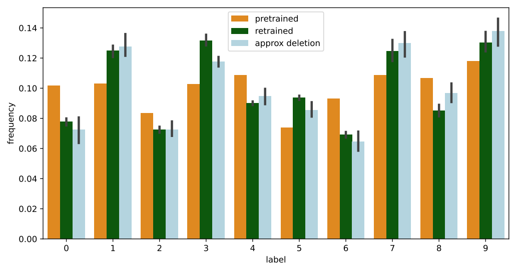
We investigate **question 2** (Fast Deletion) by comparing label distribution of $m = 50\text{K}$ generated samples from the re-trained and approximated models. Results for randomly removing 30% samples with even labels in MNIST and Fashion-MNIST are shown in Fig. 6.6a-6.6b. Results for other deletion sets are provided in Appendix F.5. We find approximated models generate fewer samples with even labels, and the label distributions are close to re-trained models.

We investigate **question 3** (Hypothesis Test) similarly to Section 6.5.1. We draw i.i.d. samples $\hat{Y}, Y_{\mathscr{H}_0} \sim \hat{p}$, and $Y_{\mathscr{H}_1} \sim \hat{p}'$, each of size $m = 1\text{K}$. We then compute LR and $\hat{\text{ASC}}$ statistics for each set with density ratio $\hat{\rho}_{\mathscr{E}}$. This procedure is repeated for $R = 100$ times. We compare distributions of $\text{LR}(Y_{\mathscr{H}_0}, \hat{\rho}_{\mathscr{E}})$ vs $\text{LR}(Y_{\mathscr{H}_1}, \hat{\rho}_{\mathscr{E}})$ and $\hat{\text{ASC}}_\phi(\hat{Y}, Y_{\mathscr{H}_0}, \hat{\rho}_{\mathscr{E}})$ vs $\hat{\text{ASC}}_\phi(\hat{Y}, Y_{\mathscr{H}_1}, \hat{\rho}_{\mathscr{E}})$ in Appendix F.5. In most cases, $\hat{\rho}_{\mathscr{E}}$ can clearly distinguish samples between pre-trained and re-trained models.

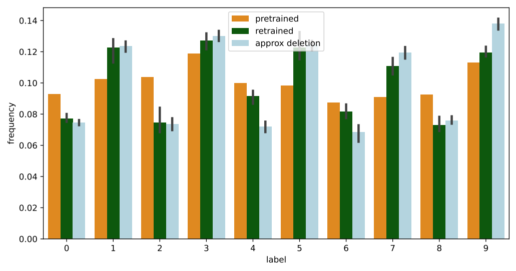
Regarding **time complexity**, our approximated deletion gives a $10.9\times$ speedup over re-training on MNIST, and a $12.0\times$ speedup on Fashion-MNIST.

6.6 Related Work

Exact data deletion from learned models (where the altered model is identical to the re-trained model) was introduced as *machine unlearning* [Cao and Yang, 2015, Bourtoule et al.,



(a) MNIST



(b) Fashion-MNIST

Figure 6.6. The Fast Deletion question: label distributions of 50K generated samples from pre-trained, re-trained, and approximated models. Mean and standard errors of five random runs are reported. The label distributions of the approximated model is close to the re-trained model.

2021]. Such deletion can be performed efficiently for relatively simpler learners such as linear regression [Chambers, 1971] and k-nearest neighbors [Schelter, 2020]. Machine unlearning for convex risk minimization was shown theoretically possible under total variation stability [Ullah et al., 2021]. Others have introduced further definitions of approximate data deletion [Guo et al., 2019, Neel et al., 2021, Sekhari et al., 2021, Izzo et al., 2021] and developed efficient methods for approximate deletion in supervised learning.

The unsupervised setting has received substantially less attention with respect to data deletion. A notable exception is clustering [Ginart et al., 2019, Borassi et al., 2020]. Our work instead focuses on generative models where the goal is to learn distribution from data rather than doing clustering. Potential avenues for future work include forging a deeper connection between approximate data deletion for generative models and differential privacy [Dwork et al., 2006] and using recent advances in *certified removal* [Guo et al., 2019] for generative models.

Outside of the context of data deletion, density ration estimation seeks to estimate the ratio between two densities from samples. For example, the ratio can be estimated via probabilistic classification [Sugiyama et al., 2012] or variational divergence minimization [Nowozin et al., 2016]. There are also many other techniques in the literature [Yamada et al., 2011, Sugiyama et al., 2012, Nowozin et al., 2016, Moustakides and Basioti, 2019, Khan et al., 2019, Rhodes et al., 2020, Kato and Teshima, 2021, Choi et al., 2021, 2022], all designed for settings with little prior information about the data. In contrast, we consider a setting where we have strong prior information (the only two possibilities are that X' was or was not deleted, rather than in prior work where the two samples can be arbitrarily separated). We adapt probabilistic classification [Sugiyama et al., 2012] and variational divergence minimization [Nowozin et al., 2016] for our setting as they lend themselves naturally to incorporating the knowledge that training data is being deleted. An avenue of future work is incorporating such knowledge into other density ratio estimation methods, any of which can be used within our general framework in Fig. 6.1.

6.6.1 Further Discussion of Our Contributions in Relationship to Related Areas

We discuss our contributions in relationship to three related areas: differential privacy, membership inference, and influence functions.

Relationship to differential privacy. First, we note that if the learner is differentially private [Dwork et al., 2006], then the re-trained model is close to the pre-trained model. This means that there is no need to perform data deletion, and it is by definition impossible to test whether training data have been deleted.

Relationship to membership inference. Second, membership inference attackers query whether a particular sample is used for training [Shokri et al., 2017]. This is akin to when the deletion set $X' = \{x'\}$ contains only one sample and membership inference is performed to test whether the training set contains x' or not. In contrast, our deletion test is based on additional prior knowledge and tests whether the training set is X or $X \setminus \{x'\}$. Therefore, membership inference is stronger but harder than the deletion test.

Relationship to influence functions. Finally, we highlight that influence functions [Koh and Liang, 2017, Koh et al., 2019, Basu et al., 2020, Kong and Chaudhuri, 2021a] designed for likelihood in generative models can potentially be used to estimate density ratio in our framework. The influence function of a sample is a measure of the impact of removing that sample from the training set on the loss function of a particular test sample [Koh and Liang, 2017]. When the deletion set only contains one sample, we could use the approximate influence score [Kong and Chaudhuri, 2021a] to derive DRE for deep generative models. We could then generalize to deleting multiple samples by summing individual influences [Koh et al., 2019, Basu et al., 2020], which is another important direction of future work.

6.7 Conclusions and Future Work

In this chapter, we propose a density-ratio-based framework for data deletion in generative modeling. Using this framework, we introduce our two main contributions: a fast method for approximate data deletion and a statistical test for estimating whether or not training points have been deleted. We provide formal guarantees for both contributions under various learner assumptions. In addition, we investigate our approximate deletion method and statistical test on real and synthetic datasets for various generative models. Our experiments confirm that (1) our methods accurately approximate the target density ratio, (2) our deletion method efficiently yields a model indistinguishable from the re-trained model, and (3) our hypothesis tests accurately distinguish samples from pre-trained and re-trained models. We highlight a limitation and important future direction: Our density-ratio-based framework results in stability limitations when applied to more complex datasets, as density ratio estimation becomes challenging when data have higher dimensions and more complex patterns.

6.8 Acknowledgements

We thank Kamalika Chaudhuri for discussion and helpful feedback. This work was supported by NSF under CNS 1804829 and ARO MURI W911NF2110317.

This chapter is a reformatted version of the paper ”Approximate Data Deletion in Generative Models” (in submission) by Zhifeng Kong and Scott Alfeld [Kong and Alfeld, 2022]. The dissertation author was the primary researcher and author of this paper.

Chapter 7
Conclusion

In this dissertation, we present in-depth study on understanding deep generative models from two aspects: expressivity and trustworthiness. In Chapter 1 and Chapter 2, we provide theoretical results on when certain models are universal approximators and when they are not. In Chapter 3, we propose an efficient and theoretically sound algorithm to investigate instance-based interpretability of VAEs. In Chapter 4 and Chapter 5, we provide a number of algorithms to redact undesirable outputs from various kinds of deep generative models. In Chapter 6, we introduce a framework that could help us identify and mitigate privacy issues of deep generative models.

Appendix A

The Expressive Power of a Class of Normalizing Flow Models

A.1 Geometric Intuition of Planar and Radial FLows

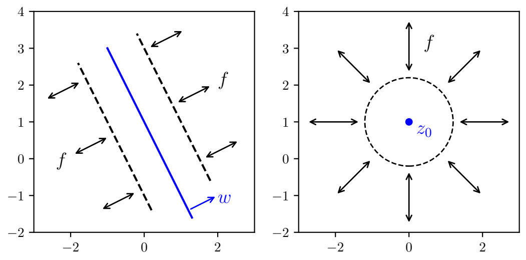


Figure A.1. Geometric intuition of planar (left) versus radial (right) flows. In $\mathbb{R}^d$, a planar flow scales non-linearly w.r.t. a $d-1$ dimensional subspace in the Cartesian coordinate system, while a radial flow scales non-linearly w.r.t. center z_0 in the polar coordinate system.

A.2 Proof of Theorem 1.3.1

Definition A.2.1. *Φ_p is defined as the cumulative function of distribution p:*

$$\Phi_p(z) = \int_{-\infty}^{z_1} dx_1 \int_{-\infty}^{z_2} dx_2 \cdots \int_{-\infty}^{z_d} dx_d \; p(x)dx.$$

Lemma A.2.2 (Possible Transformations (single flow))**.** *If p and q are densities on $\mathbb{R}$ supported* *on n non-intersecting intervals:*

$$\textbf{supp}\, q = \bigcup_{i=1}^{n} \left(l_i^{(q)}, r_i^{(q)}\right),\ \textbf{supp}\, p = \bigcup_{i=1}^{n} \left(l_i^{(p)}, r_i^{(p)}\right)$$

and if $\Phi_q\left(r_i^{(q)}\right) = \Phi_p\left(r_i^{(p)}\right)$ $\forall 1 \le i \le n$*, then there exists a planar flow* f *such that* $f\#q = p$*, a.e..*

Proof. As a special case of **Lemma** A.2.2, if two densities $\tilde{p}, q$ are supported on $\mathbb{R}$, we can transform q into $\tilde{p}$ with a planar flow. Notice that for any density supported on a finite union of intervals, it is possible to approximate it using densities supported on a finite union of intervals excluding infinity. Therefore, we only need to prove for the following case:

$$\textbf{supp}\, p = \bigcup_{i=1}^{n} (l_i, r_i).$$

To achieve this, it is sufficient to prove that there exists a distribution $\tilde{p}$ with support equal to $\mathbb{R}$ that can approximate p to within ε for any $\varepsilon > 0$. We construct $\tilde{p}$ in the following way. We first define the threshold

$$\Delta = \frac{2}{\sum_{i=1}^{n} (r_i - l_i)}.$$

Then, the measure of the set of points x with density $p(x) \ge \Delta$ is at most $1/\Delta$, and thus the measure of the set of points x with density $p(x) \in (0, \Delta)$ is at least $1/\Delta$. Define

$$\gamma = \int_{x: 0 < p(x) < \Delta} p(x) dx \le 1.$$

Now, we define $\tilde{p}(x)$ to be:

- If $p(x) \ge \Delta$, then $\tilde{p}(x) = p(x)$.
- If $0 < p(x) < \Delta$, then $\tilde{p}(x) = (1 - \varepsilon/2) p(x)$.

- If $x \in [r_i, l_{i+1}]$ for some i, then

$$\tilde{p}(x) = \frac{\varepsilon\gamma}{2n\,(l_{i+1} - r_i)}.$$

- If $x \leq l_1$ or $x \geq r_n$, then we assign $\tilde{p}(x)$ to be a tail of Gaussian distribution such that on this halfspace $\tilde{p}(x) \leq \varepsilon/2$ and the integration of it is $\frac{\varepsilon\gamma}{4n}$.

It can be examined that

$$\begin{aligned}\|\tilde{p}\|_1 &= \int_{p(x)\geq\Delta} |\tilde{p}(x)|dx + \int_{0<p(x)<\Delta} |\tilde{p}(x)|dx + \int_{p(x)=0} |\tilde{p}(x)|dx \\ &= 1 - \gamma + (1-\varepsilon/2)\gamma + \sum_{i=1}^{n-1} \frac{\varepsilon\gamma(l_{i+1}-r_i)}{2n(l_{i+1}-r_i)} + \frac{\varepsilon\gamma}{2n} \\ &= 1.\end{aligned}$$

$$\begin{aligned}\|p - \tilde{p}\|_1 &= \int_{0<p(x)<\Delta} |p(x) - \tilde{p}(x)|dx + \int_{p(x)=0} |p(x) - \tilde{p}(x)|dx \\ &= \frac{\varepsilon\gamma}{2} + \sum_{i=1}^{n-1} \frac{\varepsilon\gamma(l_{i+1}-r_i)}{2n(l_{i+1}-r_i)} + \frac{\varepsilon\gamma}{2n} \\ &= \varepsilon\gamma \leq \varepsilon.\end{aligned}$$

Thus we finish the proof. □

Proof of Lemma A.2.2

Proof. We construct such an $f(z)$ for z in different regions, and then show that this f can be written as a planar flow with a continuous non-linearity. To satisfy $f\#q = p$, *a.e.*, it is equivalent to show that $\Phi_p(f(z)) = \Phi_q(z)$ for any $z \in \mathbb{R}$.

- If $z \in \left(l_i^{(q)}, r_i^{(q)}\right)$ for some i, then $f(z) = \Phi_p^{-1} \circ \Phi_q(z)$. Since $q(z) > 0$ in this interval,

$$\Phi_p(l_i^{(p)}) = \Phi_q(l_i^{(q)}) < \Phi_q(z) < \Phi_q(r_i^{(q)}) = \Phi_p(r_i^{(p)}).$$

Therefore, $\Phi_p^{-1} \circ \Phi_q(z)$ exists. Since p, q are densities, Φ_p and Φ_q are continuous. Notice that Φ_p is increasing in a compact neighbourhood of $\Phi_q(z)$. Therefore, Φ_p^{-1} is continuous, so f is continuous.

- If $z \in \left[r_i^{(q)}, l_{i+1}^{(q)}\right]$ for some i, we let

$$f(z) = \frac{l_{i+1}^{(p)} - r_i^{(p)}}{l_{i+1}^{(q)} - r_i^{(q)}} \left(z - r_i^{(q)}\right) + r_i^{(p)}.$$

Intuitively, f linearly maps $\left[r_i^{(q)}, l_{i+1}^{(q)}\right]$ to $\left[r_i^{(p)}, l_{i+1}^{(p)}\right]$. Then, we have if $z \in \left[r_i^{(q)}, l_{i+1}^{(q)}\right]$

$$\Phi_p(f(z)) = \Phi_p\left(r_i^{(p)}\right) = \Phi_q\left(r_i^{(q)}\right) = \Phi_q(z).$$

To keep the continuity of f, we show that the boundary conditions are also satisfied:

$$f\left(r_i^{(q)}\right) = r_i^{(p)}, f\left(l_{i+1}^{(q)}\right) = l_{i+1}^{(p)}.$$

- If $z \geq r_n^{(q)}$, then $f(z) = z - r_n^{(q)} + r_n^{(p)}$ satisfies $\Phi_p(f(z)) = \Phi_q(z) = 1$ and f is continuous. If $z \leq l_1^{(q)}$, then $f(z) = z - l_1^{(q)} + l_1^{(p)}$ satisfies $\Phi_p(f(z)) = \Phi_q(z) = 0$ and f is continuous.

Now, we obtain an f that is continuous on $\mathbb{R}$ and satisfies $f\#q = p$. Finally, if we set

$$h(z) = \frac{1}{u} f\left(\frac{z-b}{w}\right) - \frac{z-b}{uw}$$

for any $u (\neq 0), w (\neq 0)$ and b, then we can see that f can be written as a planar flow: $f(z) = z + uh(wz + b)$. $\square$

A.3 Proof of Theorem 1.3.2

Definition A.3.1 (Piecewise Distributions in $\mathscr{C}$)**.** *Let $\mathscr{C}_0$ be the set of distributions with continuous densities. Suppose $\mathscr{C} \subset \mathscr{C}_0$, then we define $\mathscr{PW}(n, \mathscr{C})$ to be the set of all distributions*

p on $\mathbb{R}$ satisfying: there exists real numbers $-\infty = t_0 < t_1 < \cdots < t_{n-1} < \infty$ such that for any $i = 0, \cdots, n-1$, on the i-th interval ($(-\infty, t_1)$ if $i = 0$, $[t_{n-1}, \infty)$ if $i = n-1$, $[t_i, t_{i+1})$ otherwise) p is equal to some distribution $p_i \in \mathscr{C}$. For conciseness, we say p is described by $\{p_i, t_i\}_{i=0}^{n-1}$. We define $\mathscr{PW}(n) = \mathscr{PW}(n, \mathscr{C}_0)$. If $n' > n$, then $\mathscr{PW}(n) \subset \mathscr{PW}(n')$.

Definition A.3.2 (Piecewise Gaussian Distributions)**.** *Let $\mathscr{G}$ be the set of Gaussian distributions $\{\mathscr{N}(\mu, \sigma^2) : \mu \in \mathbb{R},\ \sigma > 0\}$. We define the set of piecewise Gaussian distributions to be $\mathscr{PW}(n, \mathscr{G})$.*

Definition A.3.3 (Tail-consistency)**.** *Suppose $p \in \mathscr{PW}(n)$ is described by $\{p_i, t_i\}_{i=0}^{n-1}$. We say p is tail-consistent w.r.t. t_k if*

$$\sum_{i=1}^{k} \int_{t_{i-1}}^{t_i} p_i(z)dz + \int_{t_k}^{\infty} p_{k+1}(z)dz = 1.$$

If p is tail-consistent w.r.t. t_k for any $k = 1, \cdots, n-1$, we say p is tail-consistent.

Lemma A.3.4 (Possible Transformations (single flow))**.** *Let two distributions $p, q \in \mathscr{PW}(n, \mathscr{G})$ satisfying: p can be described by $\{p_i, t_i\}_{i=0}^{n-1}$ and q can be described by $\{q_i, t_i\}_{i=0}^{n-1}$, where $p_i = q_i$ for $i < n-1$ (that is, the only difference is $p_{n-1} \neq q_{n-1}$). Then there exists a ReLU planar flow f such that $f\#q = p$.*

Lemma A.3.5 (Possible Transformations (flows))**.** *$\forall p \in \mathscr{PW}(n, \mathscr{G})$, if p is tail-consistent, then there exists $n-1$ ReLU planar flows $\{f_t\}_{t=1}^{n-1}$ and a Gaussian distribution $q_{\mathscr{N}}$ such that $(f_{n-1} \circ \cdots \circ f_1)\#q_{\mathscr{N}} = p$.*

Lemma A.3.6. *Given any piecewise constant distribution q_{pwc} supported on a finite union of compact intervals, $\forall \varepsilon > 0$, there exists a tail-consistent piecewise Gaussian distribution q_{pwg} such that $\|q_{pwg} - q_{pwc}\|_1 \leq \varepsilon$.*

Proof. According to Lin and Jegelka [2018], piecewise constant functions supported on a finite union of compact intervals can approximate any Lebesgue-integrable function, so do densities supported on a finite union of intervals. Therefore, there exists such piecewise constant distribution q_{pwc} such that $\|q_{pwc} - p\|_1 \leq \varepsilon/2$. According to **Lemma** A.3.6, there exists a tail-consistent piecewise Gaussian distribution q_{pwg} such that $\|q_{pwg} - q_{pwc}\|_1 \leq \varepsilon/2$. According to **Lemma** A.3.5, there exists a flow f composed of finitely many ReLU planar flows and a Gaussian distribution $q_{\mathcal{N}}$ such that $q_{pwg} = f\#q_{\mathcal{N}}$. As a result, we have $\|f\#q_{\mathcal{N}} - p\|_1 \leq \varepsilon$. □

Proof of Lemma A.3.4

Proof. By assumption, $p(y) = q(y)$ if $y < t_{n-1}$. Now, we assume on $[t_{n-1}, \infty)$, $q \sim \mathcal{N}(\mu_n, \sigma_n^2)$, and $p \sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$. Let f be a ReLU planar flow with parameters u, w and b, where $u = sgn(\hat{\sigma} - \sigma_n), w = |1 - \hat{\sigma}/\sigma_n|, b = -wt_{n-1}$. Then, for any $y \in \mathbb{R}$,

$$f^{-1}(y) = \begin{cases} y & wy + b < 0 \\ \frac{y-ub}{1+uw} & wy + b \geq 0 \end{cases} = \begin{cases} y & y < t_{n-1} \\ \frac{y-ub}{1+uw} & y \geq t_{n-1} \end{cases}.$$

According to (1.1) and (1.9), if $y < t_{n-1}$, $(f\#q)(y) = q(y)$. If $y \geq t_{n-1}$,

$$(f\#q)(y) = \frac{q\left(\frac{y-ub}{1+uw}\right)}{1+uw} = \frac{\mathcal{N}\left(\frac{y-ub}{1+uw}; \mu_n, \sigma_n^2\right)}{1+uw}.$$

Thus, on $[t_{n-1}, \infty)$,

$$f\#q \sim \mathcal{N}(ub + (1+uw)\mu_n, (1+uw)^2\sigma_n^2) = \mathcal{N}((1+uw)\mu_n - uwt_{n-1}, (1+uw)^2\sigma_n^2).$$

Since $uw = \frac{\hat{\sigma}}{\sigma_n} - 1$, $f\#q \sim \mathcal{N}(\tilde{\mu}, \hat{\sigma}^2)$ for some $\tilde{\mu}$ on $[t_{n-1}, \infty)$. Notice that

$$\int_{t_{n-1}}^{\infty} \mathcal{N}(y; \tilde{\mu}, \hat{\sigma}^2)dy = 1 - \sum_{i=0}^{n-2} \int_{t_i}^{t_{i+1}} q_i(y)dy = 1 - \sum_{i=0}^{n-2} \int_{t_i}^{t_{i+1}} p_i(y)dy = \int_{t_{n-1}}^{\infty} \mathcal{N}(y; \hat{\mu}, \hat{\sigma}^2)dy.$$

We know that $\tilde{\mu} = \hat{\mu}$. Thus, the ReLU flow with the above u, w and b transforms the right-most piece of the input distribution q to the desired target p without changing the other pieces. □

Proof of Lemma A.3.5

Proof. We prove by induction. For $n = 1$, the result is obvious since any Gaussian distribution can be chosen as input. Suppose we are able to generate any tail-consistent distribution in $\mathscr{PW}(n-1,\mathscr{G})$. Given the target distribution $p \in \mathscr{PW}(n,\mathscr{G})$ described by $\{p_i,t_i\}_{i=0}^{n-1}$, where

$$p_i(z) = \mathscr{N}(z;\mu_i,\sigma_i^2),\ i = 0,\cdots,n-1,$$

we first generate an intermediate distribution $q_{int} \in \mathscr{PW}(n-1,\mathscr{G})$ described by $\{q_i,t_i\}_{i=0}^{n-2}$, where

$$q_i = p_i,\ i = 0,\cdots,n-2.$$

Since p is tail-consistent, q_{int} integrates to 1 on $\mathbb{R}$, so it is a probability distribution. Notice that q_{int} can be viewed as an element in $\mathscr{PW}(n,\mathscr{G})$ described by $\{q_i,t_i\}_{i=0}^{n-1}$, where $q_{n-1} = q_{n-2}$. Then, according to **Lemma** A.3.4, we can apply one more layer of ReLU flow to transform q_{int} into the desired distribution p. $\square$

Proof of Lemma A.3.6

Proof. Suppose the target distribution q_{pwc} has a compact support $\subset [t_-,t_+]$, where $q_{pwc}(t_-)$ and $q_{pwc}(t_+)$ are strictly positive. We construct q_{pwg} as follows. First , we let

$$\int_{-\infty}^{t_-} q_{pwg}(x)dx = \int_{t_+}^{\infty} q_{pwg}(x)dx = \frac{\varepsilon}{3}.$$

This can be done by setting $q_{pwg} = \mathscr{N}(\mu_-,\sigma_-^2)$ on $(-\infty,t_-)$ where $(t_- - \mu_-)/\sigma_- = \Phi^{-1}_{\mathscr{N}(0,1)}(\frac{\varepsilon}{3})$, and $q_{pwg} = \mathscr{N}(\mu_+,\sigma_+^2)$ on (t_+,∞) where $(t_+ - \mu_+)/\sigma_+ = \Phi^{-1}_{\mathscr{N}(0,1)}(1-\varepsilon/3)$.

On $[t_-,t_+]$, suppose q_{pwc} is a piecewise constant function on n intervals of δ_i width, where $\delta/2 < \delta_i < \delta$ for $1 \le i \le n$, and δ is an arbitrarily small positive value. Then, the number of intervals n is $\Theta(1/\delta)$.

Now, we look at the i-th interval, where $1 \leq i \leq n$. Suppose $q_{pwc}(x) = \alpha$ for $x \in [t, t+\delta_i)$. Then, a valid tail-consistent piecewise Gaussian piece on this interval has the form $\mathscr{N}(\mu, \sigma^2)$ with

$$\int_t^\infty \mathscr{N}(x; \mu, \sigma^2) dx = \left(1 - \frac{2}{3}\varepsilon\right) \int_t^\infty q_{pwc}(x) dx.$$

This guarantees that q_{pwg} is tail-consistent and integrates to 1 on $\mathbb{R}$. The solution of μ and σ is given by $(t-\mu)/\sigma = c$ for some constant c such that $|c| \leq \Phi^{-1}_{\mathscr{N}(0,1)}(\varepsilon/3)$. Now, we show that $\mathscr{N}(x; \mu, \sigma^2)$ approximates α in ℓ_1 norm on $[t, t+\delta_i)$. If $\alpha = 0$, by letting $\sigma \to \infty$ we are able to approximate 0 to within any precision. Thus, we only discuss cases where $\alpha > 0$. We assign $\mathscr{N}(t; \mu, \sigma^2) = \alpha$. The solution is given by

$$\sigma = \frac{\exp(-\frac{c^2}{2})}{\sqrt{2\pi}\alpha}, \ \mu = t - c\sigma.$$

One can check that the Lipschitz constant of the Gaussian distribution is $\frac{1}{\sqrt{2\pi e}\sigma^2}$. Thus, the ℓ_1 norm of the difference between $\mathscr{N}(\mu, \sigma^2)$ and α on $[t, t+\delta_i)$ is bounded by

$$\int_t^{t+\delta_i} \left|\mathscr{N}(x; \mu, \sigma^2) - \alpha\right| dx \leq \frac{\delta^2}{2\sqrt{2\pi e}\sigma^2} = \sqrt{\frac{\pi}{2}} \alpha^2 \exp\left(c^2 - \frac{1}{2}\right) \delta^2.$$

Since we have finite subdivisions, α can be seen as an $\mathscr{O}(1)$ constant. Combining with the bound on c, we have

$$\int_t^{t+\delta_i} \left|\mathscr{N}(x; \mu, \sigma^2) - q_{pwc}(x)\right| dx \leq \sqrt{\frac{\pi}{2}} \left(\sup_{x \in \mathbb{R}} q_{pwc}(x)^2\right) \exp\left(\Phi^{-1}_{\mathscr{N}(0,1)}(\varepsilon/3)^2 - \frac{1}{2}\right) \delta^2.$$

Since there are $n \leq 2/\delta$ intervals, we know that

$$\int_{t_-}^{t_+} \left|\mathscr{N}(x; \mu, \sigma^2) - q_{pwc}(x)\right| dx \leq \sqrt{2\pi} \left(\sup_{x \in \mathbb{R}} q_{pwc}(x)^2\right) \exp\left(\Phi^{-1}_{\mathscr{N}(0,1)}(\varepsilon/3)^2 - \frac{1}{2}\right) \delta.$$

Since δ is arbitrary, we can assign

$$\delta = \frac{\varepsilon}{3\sqrt{2\pi}\,(\sup_{x\in\mathbb{R}} q_{pwc}(x)^2)\exp\left(\Phi^{-1}_{\mathscr{N}(0,1)}(\varepsilon/3)^2 - \frac{1}{2}\right)}.$$

Then,

$$\int_{-\infty}^{\infty} |q_{pwg}(x) - q_{pwc}(x)|dx = \left(\int_{-\infty}^{t_-} + \int_{t_-}^{t_+} + \int_{t_+}^{\infty}\right) |q_{pwg}(x) - q_{pwc}(x)|dx \leq \frac{\varepsilon}{3} + \frac{\varepsilon}{3} + \frac{\varepsilon}{3} = \varepsilon.$$

As a result, $\forall \varepsilon > 0$, there exists a tail-consistent distribution $q_{pwg} \in \mathscr{PW}(n+2, \mathscr{G})$ satisfying $\|q_{pwg} - q_{pwc}\|_1 \leq \varepsilon$, where $n = \mathscr{O}\left(\exp\left(\Phi^{-1}_{\mathscr{N}(0,1)}(\varepsilon/3)^2\right)/\varepsilon\right)$. □

A.4 Proof of Theorem 1.4.1

Lemma A.4.1. *Let $\{f_i\}_{i=1}^n$ be n ReLU Sylvester flows on $\mathbb{R}^d$ and $f = f_n \circ \cdots \circ f_1$. Then, there exists a zero-measure closed set $\Omega \subset \mathbb{R}^d$ such that $\forall x \in \mathbb{R}^d \setminus \Omega$, there exists an open neighbourhood of x called Γ_x, such that $J_f(z)$ is equal to a constant matrix for $z \in \Gamma_x$.*

Proof. According to **Lemma** A.4.1, there exists a zero-measure closed set $\Omega \subset \mathbb{R}^d$ such that $\forall z \in \mathbb{R}^d \setminus \Omega$, $J_f(z)$ is constant in an open neighbourhood of z. By the change-of-variable formula in (1.2), for small $\alpha \in \mathbb{R}$ and any direction $\delta \in \mathbb{R}^d$,

$$\log p(f(z+\alpha\delta)) - \log q(z+\alpha\delta) = \log p(f(z)) - \log q(z).$$

Next, we expand the Taylor series of $f(z+\alpha\delta)$ for small α:

$$f(z+\alpha\delta) = f(z) + \alpha J_f(z)\delta + \mathscr{O}(\alpha^2).$$

Therefore,

$$\log p(f(z) + \alpha J_f(z)\delta + \mathscr{O}(\alpha^2)) - \log p(f(z)) = \log q(z+\alpha\delta) - \log q(z).$$

By multiplying $1/\alpha$ on both sides and taking $\alpha \to 0$, we finish the proof. $\square$

Remark A.4.2. ***Theorem** 1.4.1 can be extended to any Sylvester flow with $h'' = 0$ almost everywhere.*

Remark A.4.3. ***Theorem** 1.4.1 can be extended to Householder flows [Tomczak and Welling, 2016].*

Remark A.4.4. *An example of directional derivative is illustrated in Figure A.2.*

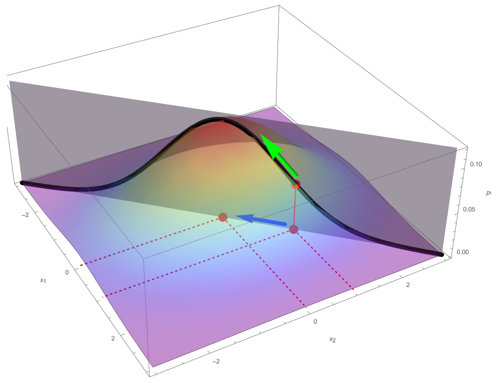


Figure A.2. Directional derivative (green arrow) of a two-dimensional Gaussian distribution at point $z = (1,1)$ and direction $\delta = (-1,-1)$ (blue arrow).

Proof of Lemma A.4.1

Proof. Suppose the ith Sylvester flow f_i has parameters A_i, B_i, b_i for $i = 1, \cdots, n$. Notice that when $B_i^\top z + b_i \neq 0$, there exists an open set $\mathscr{B}_z$ containing z such that $\forall y \in \mathscr{B}_z$, the signs of $B_i^\top y + b_i$ is identical to those of $B_i^\top z + b_i$. Therefore, $J_{f_i}(y)$ is equal to a constant matrix in $\mathscr{B}_z$. Then, the statement straightly follows from the chain rule of Jacobian matrix, where

$$\Omega = \bigcup_{i=1}^{n} \{z : B_i^\top z + b_i = 0\}.$$

$\square$

A.5 Formal Version of Corollary 1.4.2

Corollary A.5.1 (MoG$\not\to$MoG)**.** *Suppose p, q are mixture of Gaussian distributions on $\mathbb{R}^d$ in the following form:*

$$p(z) = \sum_{i=1}^{r_p} w_p^i \mathcal{N}(z; \mu_p^i, \Sigma_p),\ q(z) = \sum_{j=1}^{r_q} w_q^j \mathcal{N}(z; \mu_q^j, \Sigma_q).$$

If a flow f composed of finitely many ReLU Sylvester flows satisfies $p = f\#q$, then for almost every point $x \in \mathbb{R}^d$, it has an open neighbourhood Γ_x such that $\forall z \in \Gamma_x$,

$$\begin{aligned}&\frac{\sum_{i,j=1}^{r_p} w_p^i \mathcal{N}(Az+b; \mu_p^i, \Sigma_p)\mathcal{N}(Az+b; \mu_p^j, \Sigma_p) A^\top \Sigma_p^{-1} \mu_p^i (\mu_p^i - \mu_p^j)^\top \Sigma_p^{-1} A}{\left(\sum_{j=1}^{r_p} w_p^j \mathcal{N}(Az+b; \mu_p^j, \Sigma_p)\right)^2}\\ - &\frac{\sum_{i,j=1}^{r_q} w_q^i \mathcal{N}(z; \mu_q^i, \Sigma_q)\mathcal{N}(z; \mu_q^j, \Sigma_q)\Sigma_q^{-1} \mu_q^i (\mu_q^i - \mu_q^j)^\top \Sigma_q^{-1}}{\left(\sum_{j=1}^{r_q} w_q^j \mathcal{N}(z; \mu_q^j, \Sigma_q)\right)^2}.\end{aligned}$$

is a constant function in z on Γ_x for some $A \in \mathbb{R}^{d\times d}$ and $b \in \mathbb{R}^d$.

Proof. Suppose $f = f_1 \circ \cdots \circ f_n$ is a normalizing flow composed of finite ReLU Sylvester flows. For almost every $x \in \mathbb{R}^d$, we have J_f is equal to a constant matrix A in an open neighbourhood of x called Γ_x. That is, for some $b \in \mathbb{R}^d$,

$$f(z) = Az + b,\ \forall z \in \Gamma_x.$$

Now, we solve the topology matching condition in **Theorem** 1.4.1 on Γ_x.

$$\begin{aligned}\nabla_z \log q(z) &= -\frac{1}{q(z)} \Sigma_q^{-1} \left(\sum_{j=1}^{r_q} w_q^j \mathcal{N}(z; \mu_q^j, \Sigma_q)(z - \mu_q^j) \right)\\ &= -\Sigma_q^{-1} \left(z - \sum_{j=1}^{r_q} \frac{w_q^j \mathcal{N}(z; \mu_q^j, \Sigma_q)}{q(z)} \mu_q^j \right).\end{aligned}$$

Similarly,

$$\begin{aligned} J_f(z)^\top \nabla_z \log p(f(z)) &= -\frac{1}{p(f(z))} J_f(z)^\top \Sigma_p^{-1} \left(\sum_{i=1}^{r_p} w_p^i \mathscr{N}(f(z); \mu_p^i, \Sigma_p)(f(z) - \mu_p^i) \right) \\ &= -J_f(z)^\top \Sigma_p^{-1} \left(f(z) - \sum_{i=1}^{r_p} \frac{w_p^i \mathscr{N}(f(z); \mu_p^i, \Sigma_p)}{p(f(z))} \mu_p^i \right). \end{aligned}$$

Therefore, we obtain

$$\begin{aligned} &\left(A^\top \Sigma_p^{-1} A - \Sigma_q^{-1} \right) z + A^\top \Sigma_p^{-1} b \\ &= A^\top \Sigma_p^{-1} \left(\sum_{i=1}^{r_p} \frac{w_p^i \mathscr{N}(Az+b; \mu_p^i, \Sigma_p)}{p(Az+b)} \mu_p^i \right) - \Sigma_q^{-1} \left(\sum_{j=1}^{r_q} \frac{w_q^j \mathscr{N}(z; \mu_q^j, \Sigma_q)}{q(z)} \mu_q^j \right). \end{aligned}$$

Notice that the left-hand-side is linear in z. Thus, if $p = f\#q$, then the right-hand-side is should be linear in z. By standard arithmetic we can calculate the derivative of the right-hand-side over z as follow:

$$\begin{aligned} &\frac{\sum_{i,j=1}^{r_p} w_p^i \mathscr{N}(Az+b; \mu_p^i, \Sigma_p) \mathscr{N}(Az+b; \mu_p^j, \Sigma_p) A^\top \Sigma_p^{-1} \mu_p^i (\mu_p^i - \mu_p^j)^\top \Sigma_p^{-1} A}{\left(\sum_{j=1}^{r_p} w_p^j \mathscr{N}(Az+b; \mu_p^j, \Sigma_p) \right)^2} \\ - &\frac{\sum_{i,j=1}^{r_q} w_q^i \mathscr{N}(z; \mu_q^i, \Sigma_q) \mathscr{N}(z; \mu_q^j, \Sigma_q) \Sigma_q^{-1} \mu_q^i (\mu_q^i - \mu_q^j)^\top \Sigma_q^{-1}}{\left(\sum_{j=1}^{r_q} w_q^j \mathscr{N}(z; \mu_q^j, \Sigma_q) \right)^2}. \end{aligned}$$

However, this is generally a non-constant function in z except for some special cases. □

Remark A.5.2. *To give a simple case where the condition in* ***Corollary*** *A.5.1 does not hold, we let $r_p = r_q = 2$, $\mu_p^1 = \mu_p^1$, $\mu_q^1 \neq \mu_q^2$ and $w_q^1 = w_q^2 = \frac{1}{2}$. Then, the difference in the condition is given by*

$$-\frac{2\mathscr{N}(z; \mu_q^1, \Sigma_q) \mathscr{N}(z; \mu_q^2, \Sigma_q)}{\left(\mathscr{N}(z; \mu_q^1, \Sigma_q) + \mathscr{N}(z; \mu_q^2, \Sigma_q) \right)^2} \Sigma_q^{-1} (\mu_q^1 - \mu_q^2)(\mu_q^1 - \mu_q^2)^\top \Sigma_q^{-1}.$$

If it is a constant function in z, then both $\mathscr{N}(z; \mu_q^1, \Sigma_q) + \mathscr{N}(z; \mu_q^2, \Sigma_q)$ and $\mathscr{N}(z; \mu_q^1, \Sigma_q) -$

$\mathcal{N}(z;\mu_q^2,\Sigma_q)$ are equal to a constant times $\sqrt{\mathcal{N}(z;\mu_q^1,\Sigma_q)\mathcal{N}(z;\mu_q^2,\Sigma_q)}$. As a result,

$$\mathcal{N}(z;\mu_q^1,\Sigma_q)/\mathcal{N}(z;\mu_q^2,\Sigma_q)$$

is a constant for $z \in \Gamma_x$. By expanding the density expression, we have $z^\top \Sigma_q^{-1}(\mu_q^1-\mu_q^2)$ is a constant for $z \in \Gamma_x$. However, since $\mu_q^1 \neq \mu_q^2$, $\Sigma_q^{-1}(\mu_q^1-\mu_q^2) \neq 0$. Contradiction.

A.6 Formal Version of Corollary 1.4.3

Corollary A.6.1 (Prod$\not\twoheadrightarrow$Prod)**.** *Suppose p,q are product distributions in the following form:*

$$p(z) \propto \prod_{i=1}^{d} g(z_i)^{r_p};\ q(z) \propto \prod_{i=1}^{d} g(z_i)^{r_q},$$

where $r_p, r_q > 0, r_p \neq r_q$, and g is a smooth function. If a flow f composed of finitely many ReLU Sylvester flows satisfies $p = f\#q$, then for almost every point $x \in \mathbb{R}^d$, it has an open neighbourhood Γ_x such that $\forall z \in \Gamma_x$,

$$r_q \tilde{\nabla} \log \boldsymbol{g}(z) = r_p A^\top \tilde{\nabla} \log \boldsymbol{g}(Az+b)$$

holds for some $b \in \mathbb{R}^d$, where $\boldsymbol{g}(z) = (g(z_1),\cdots,g(z_d))^\top$, and $\tilde{\nabla}$ takes the gradient of the i-th function w.r.t the i-th variable for $1 \leq i \leq d$.

Proof. Suppose $f = f_1 \circ \cdots \circ f_n$ is a normalizing flow composed of finite ReLU Sylvester flows. For almost every $x \in \mathbb{R}^d$, we have J_f is equal to a constant matrix A in an open neighbourhood of x called Γ_x. That is, for some $b \in \mathbb{R}^d$,

$$f(z) = Az+b,\ \forall z \in \Gamma_x.$$

Now, we solve the topology matching condition in **Theorem** 1.4.1 on Γ_x. By matching the corresponding elements, we have the following result:

$$r_p \sum_{i=1}^{d} A_{ij} \frac{g'((Az+b)_i)}{g((Az+b)_i)} = r_q \frac{g'(z_j)}{g(z_j)},\ j = 1, \cdots, d.$$

Rewriting this equation into vector form, we finish our proof. □

Remark A.6.2. *To give a simple case where the condition in **Corollary** A.6.1 does not hold, we let $d = 2$ and $g(x) = x$. Then, the necessary condition becomes*

$$\begin{cases} \dfrac{r_q}{r_p z_1} &= \dfrac{A_{11}}{A_{11}z_1 + A_{12}z_2 + b_1} + \dfrac{A_{21}}{A_{21}z_1 + A_{22}z_2 + b_2} \\ \dfrac{r_q}{r_p z_2} &= \dfrac{A_{12}}{A_{11}z_1 + A_{12}z_2 + b_1} + \dfrac{A_{22}}{A_{21}z_1 + A_{22}z_2 + b_2} \end{cases},$$

or equivalently,

$$\begin{aligned} & r_q(A_{11}z_1 + A_{12}z_2 + b_1)(A_{21}z_1 + A_{22}z_2 + b_2) \\ = \quad & (A_{11}(A_{21}z_1 + A_{22}z_2 + b_2) + A_{21}(A_{11}z_1 + A_{12}z_2 + b_1))r_p z_1 \\ = \quad & (A_{12}(A_{21}z_1 + A_{22}z_2 + b_2) + A_{22}(A_{11}z_1 + A_{12}z_2 + b_1))r_p z_2. \end{aligned}$$

By checking the z_1z_2 term, we obtain $A_{11}A_{22} + A_{12}A_{21} = 0$, which indicates that $\det A = 0$. *This contradicts the fact that f is an invertible flow. As a result, there does not exist a flow composed of finitely many ReLU flows that transform $p(z) \propto (z_1z_2)^{r_p}$ to $q(z) \propto (z_1z_2)^{r_q}$.*

A.7 Positive Results for ReLU Planar Flows

Theorem A.7.1 (Linear Transformations)**.** *If $A \in \mathbb{R}^d$ has the LU decomposition, then the linear transformation $g(z) = Az$ can be generated by $4d - 4$ ReLU planar flows.*

Proof. First, we show that certain rank-one-modification transformations ($f(z) = (I+R)z$ where *rank*$(R) = 1$) can be achieved by composing two ReLU planar flows. Suppose $R = uw^\top$ where

$\det(I+R) = 1+u^\top w > 0$. We assign

$$f_1(z) = z + h(w^\top z)u,$$

$$f_2(z) = z - h(-w^\top z)u,$$

then $f = f_2 \circ f_1$: if $w^\top z < 0$, then $f_1(z) = z$, so $f_2 \circ f_1(z) = f_2(z) = z + uw^\top z$; if $w^\top z \geq 0$, then $f_1(z) = z + uw^\top z$, and since $w^\top(I+uw^\top)z = (1+u^\top w)w^\top z \geq 0$, we have $f_2 \circ f_1(z) = f_1(z) = z + uw^\top z$.

Now, assume that A has the LU decomposition:

$$A = LU$$

where $L(U)$ is a lower (upper) triangular matrix. Notice that both L and U can be decomposed to a product of $d-1$ Frobenius matrices. Since the determinant of a Frobenius matrix is $1 > 0$, both L and U can be decomposed to product of $2(d-1)$ ReLU planar flows. Therefore, we need $4d-4$ planar flows to express A. $\square$

Corollary A.7.2. *For any $A \in \mathbb{R}^d$, the linear transformation $g(z) = Az$ can be generated by $4d-4$ ReLU planar flows and d Householder flows.*

Proof. Since any matrix has LUP decomposition, we have $A = LUP$ where $L(U)$ is a lower (upper) triangular matrix, and P is a permutation matrix. Since any permutation matrix is an orthogonal matrix, P can be decomposed to a product of d Householder matrices. Using the analysis in the proof of **Theorem** A.7.1, we finish the proof. $\square$

Corollary A.7.3. *Given any Gaussian distributions $q \sim \mathcal{N}(0, \Sigma_q)$ and $p \sim \mathcal{N}(0, \Sigma_p)$ centered at the origin, we can transform q into p with $4d-4$ ReLU planar flows and d Householder flows.*

Proof. Notice that a PSD matrix Σ can be decomposed to $Q^\top \Lambda Q$, where Q is an orthogonal matrix and Λ is a diagonal matrix. Therefore, we have

$$\Sigma_q = Q_q^\top \Lambda_q Q_q.$$

$$\Sigma_p = Q_p^\top \Lambda_p Q_p.$$

Now, we assign

$$f(z) = Q_p^{-1} \Lambda_p^{-\frac{1}{2}} \Lambda_q^{\frac{1}{2}} Q_q z.$$

One can check that this linear function f transforms q into p. Using the result in **Corollary** A.7.2, we finish the proof. $\square$

A.8 Proof of Theorem 1.4.4

Lemma A.8.1 (Topology Matching for single Sylvester flow)**.** *Suppose distribution q is defined on $\mathbb{R}^d$, and a Sylvester flow f on $\mathbb{R}^d$ has tangent matrix B and smooth non-linearity. Let $p = f\#q$. Then $\forall z \in \mathbb{R}^d$, we have*

$$\nabla_z \log p(f(z)) - \nabla_z \log q(z) \in \textbf{span}\{B\}.$$

Proof. We prove by induction on n. If $n = 1$, then it is equivalent to **Lemma** A.8.1. Suppose the conclusion holds for $n-1$: $\forall z \in \mathbb{R}^d$,

$$\nabla_z \log(g\#q)(g(z)) - \nabla_z \log q(z) \in \textbf{span}\{B_1, \cdots, B_{n-1}\}$$

where $g = f_{n-1} \circ \cdots \circ f_1$. Then, we apply **Lemma** A.8.1 on f_n at $g(z)$. As a result, we obtain that $\forall z \in \mathbb{R}^d$,

$$\nabla_z \log((f_n \circ g)\#q)(f_n \circ g(z)) - \nabla_z \log(g\#q)(g(z)) \in \textbf{span}\{B_n\}.$$

By adding these two equations, we finish the proof. $\square$

Proof of Lemma A.8.1

Proof. For any $\alpha \in \mathbb{R}$, according to the expression of Sylvester flows, we have for any $w^{\perp} \in$ $\mathbf{span}\{B\}^{\perp}$,

$$B^{\top} w^{\perp} = \mathbf{0}.$$

Therefore,

$$f(z+\alpha w^{\perp}) = z+\alpha w^{\perp} + Ah(B^{\top} z + b + \alpha B^{\top} w^{\perp}) = f(z) + \alpha w^{\perp}.$$

Therefore, $\det J_f(z) = \det J_f(z+\alpha w^{\perp})$. According to (1.2), we have

$$\begin{aligned} \log p(f(z)) &= \log(q(z)) - \log\det J_f(z), \\ \log p(f(z+\alpha w^{\perp})) &= \log(q(z+\alpha w^{\perp})) - \log\det J_f(z+\alpha w^{\perp}). \end{aligned}$$

Subtracting these two equations, we have

$$\log p(f(z)+\alpha w^{\perp}) - \log p(f(z)) = \log(q(z+\alpha w^{\perp})) - \log q(z).$$

By multiplying $1/\alpha$ on both sides and taking $\alpha \to 0$, we have $\forall w^{\perp} \in \mathbf{span}\{B\}^{\perp}$,

$$(\nabla_z \log p(f(z)))^{\top} w^{\perp} - (\nabla_z \log q(z))^{\top} w^{\perp} = 0.$$

Therefore, $\nabla_z \log p(f(z)) - \nabla_z \log q(z) \in \mathbf{span}\{B\}$. $\square$

Remark A.8.2. *The property $f(z+\alpha w^{\perp}) = f(z) + \alpha w^{\perp}$ is enjoyed exclusively by Sylvester flows. Let $g(z) = f(z) - z$, then we have $g(z+\alpha w^{\perp}) = g(z)$ $\forall z \in \mathbb{R}^d, \forall \alpha \in \mathbb{R}, \forall w^{\perp} \in \mathbf{span}\{B\}^{\perp}$. Therefore,*

$$g(z) = g(\mathscr{P}_B z)$$

where $\mathscr{P}_B$ is the projection matrix to the subspace spanned by column vectors of B. Then, we have

$$f(z) = z + g(z) = z + g(\mathscr{P}_B z)$$

As a result, f can be expressed as a Sylvester flow.

A.9 Formal Version of Corollary 1.4.5 for Planar and Sylvester Flows

Corollary A.9.1 (Planar flow $\mathscr{N} \nrightarrow \mathscr{N}$)**.** *Let $p \sim \mathscr{N}(0, \Sigma_p), q \sim \mathscr{N}(0, \Sigma_q)$ be two Gaussian distributions on $\mathbb{R}^d$. If there exists a planar flow f on $\mathbb{R}^d$ with smooth non-linearity such that $p = f\#q$, then rank $\left(\Sigma_q - \Sigma_p\right) \leq 1$.*

Proof. If there exists a planar flow $f(z) = z + uh(w^\top z + b)$ transforming q into p, then according to **Lemma** A.8.1, we have $\forall z \in \mathbb{R}^d, \forall w^\perp \in \mathbf{span}\{w\}^\perp$,

$$z^\top \Sigma_q^{-1} w^\perp = f(z)^\top \Sigma_p^{-1} w^\perp$$

or equivalently,

$$z^\top \left(\Sigma_q^{-1} - \Sigma_p^{-1}\right) w^\perp = h(w^\top z + b) u^\top \Sigma_p^{-1} w^\perp.$$

First, by setting $z = 0$, we obtain

$$h(b) u^\top \Sigma_p^{-1} w^\perp = 0, \ \forall w^\perp \in \mathbf{span}\{w\}^\perp.$$

Then, by setting $z = w^\perp$ and using the above equation, we obtain

$$(w^\perp)^\top \left(\Sigma_q^{-1} - \Sigma_p^{-1}\right) w^\perp = h(b) u^\top \Sigma_p^{-1} w^\perp = 0, \ \forall w^\perp \in \mathbf{span}\{w\}^\perp.$$

- If $w^\top \left(\Sigma_q^{-1} - \Sigma_p^{-1}\right) w = 0$, then $\Sigma_p = \Sigma_q$.

- If $w^\top \left(\Sigma_q^{-1} - \Sigma_p^{-1}\right) w > 0$, then $\Sigma_q^{-1} - \Sigma_p^{-1}$ is PSD, and can be factorized as $Q^\top \Lambda Q$, where Q is orthogonal and Λ is diagonal. As a result,

$$\Lambda^{\frac{1}{2}} Q w^\perp = 0, \ \forall w^\perp \in \mathbf{span}\{w\}^\perp.$$

  This indicates that $rank\left(\Lambda^{\frac{1}{2}}\right) = 1$, or $rank\left(\Sigma_q^{-1} - \Sigma_p^{-1}\right) = 1$.

- If $w^\top \left(\Sigma_q^{-1} - \Sigma_p^{-1}\right) w < 0$, we do the same analysis to $\Sigma_p^{-1} - \Sigma_q^{-1}$ and obtain the same result as above.

Therefore, if $rank(\Sigma_q^{-1} - \Sigma_p^{-1}) > 1$, there does not exist such planar flow that transforms q into p. Suppose $rank\left(\Sigma_q^{-1} - \Sigma_p^{-1}\right) = 1$, Since covariance matrices are symmetric, we have $\Sigma_q^{-1} - \Sigma_p^{-1} = \pm \tilde{v}\tilde{v}^\top$. Therefore, $\Sigma_q = (\Sigma_p^{-1} \pm \tilde{v}\tilde{v}^\top)^{-1}$ for some $\tilde{v} \in \mathbb{R}^d$. According to the Sherman$-$Morrison formula [Sherman and Morrison, 1950], we obtain

$$\Sigma_q = \Sigma_p - \frac{\pm \Sigma_p \tilde{v}\tilde{v}^\top \Sigma_p}{1 \pm \tilde{v}^\top \Sigma_p \tilde{v}}.$$

By assigning $v = \frac{\Sigma_p \tilde{v}}{\sqrt{1 \pm \tilde{v}^\top \Sigma_p \tilde{v}}}$, we obtain $\Sigma_p - \Sigma_q = \pm v v^\top$. □

Corollary A.9.2 (Sylvester flow $\mathcal{N} \nrightarrow \mathcal{N}$). *Let $p \sim \mathcal{N}(0, \Sigma_p), q \sim \mathcal{N}(0, \Sigma_q)$ be two Gaussian distributions on $\mathbb{R}^d$, and $A = \Sigma_q^{-1} - \Sigma_p^{-1}$ with eigenvalues $\lambda_1 \leq \cdots \leq \lambda_d$. Suppose a flow f on $\mathbb{R}^d$ composed of n Sylvester flows with flow dimensions $\{m_i\}_{i=1}^n$ and smooth non-linearities satisfies $p = f\#q$. If $m = \sum_{i=1}^n m_i < d$, then we have $\lambda_{m+1} \geq 0$, $\lambda_{d-m} \leq 0$. As a result, $rank(A) \leq 2m$.*

Proof. Since $m < d$, $U^* = \mathbf{span}\{B_1, \cdots, B_n\}^\perp$ is a subspace of $\mathbb{R}^d$ with dimension at least $d - m$. According to the proof of **Corollary** A.9.1, we have

$$(w^\perp)^\top \left(\Sigma_q^{-1} - \Sigma_p^{-1}\right) w^\perp = 0, \ \forall w^\perp \in U^*.$$

Let $A = \Sigma_q^{-1} - \Sigma_p^{-1}$ with eigenvalues $\lambda_1 \leq \cdots \leq \lambda_d$. According to the Courant-Fischer theorem

(Chapter 5.2.2. (4), [Lütkepohl, 1996]),

$$\begin{aligned}
\lambda_{d-m} &\leq \lambda_{\dim U^*} \\
&= \min_{\dim W = \dim U^*} \max_{x \in W, x \neq 0} \frac{x^\top A x}{x^\top x} \\
&\leq \max_{x \in U^*, x \neq 0} \frac{x^\top A x}{x^\top x} \\
&= 0.
\end{aligned}$$

$$\begin{aligned}
\lambda_{m+1} &\geq \lambda_{d+1-\dim U^*} \\
&= \max_{\dim W = \dim U^*} \min_{x \in W, x \neq 0} \frac{x^\top A x}{x^\top x} \\
&\geq \min_{x \in U^*, x \neq 0} \frac{x^\top A x}{x^\top x} \\
&= 0.
\end{aligned}$$

When $m+1 \leq d-m$ (or $m < d/2$), we can infer that $\lambda_i = 0$ for $m+1 \leq i \leq d-m$. Therefore, A has at least $d-2m$ zero eigenvalues. This indicates that $rank(A) \leq 2m$. $\square$

A.10 Comparison with Radial Flows

In this section, we present the connection and difference between Sylvester and radial flows from geometric insights. First, we present the topology matching condition for a single radial flow in the following theorem.

Theorem A.10.1 (Topology Matching for single radial flow)**.** *Suppose distribution q is defined on $\mathbb{R}^d$, and a radial flow f on $\mathbb{R}^d$ has smoothing factor $a \in \mathbb{R}^+$, scaling factor $b \in \mathbb{R}$, and center $z_0 \in \mathbb{R}^d$. Let $p = f\#q$. Then $\forall z \in \mathbb{R}^d \setminus \{z_0\}$, we have*

$$\left(1 + \frac{b}{a + \|z - z_0\|_2}\right) \nabla_z \log p(f(z)) - \nabla_z \log q(z)$$

is parallel to $z - z_0$.

Though similar to the condition presented in **Lemma** A.8.1 in the high level sketch, there are two notable differences in **Theorem** A.10.1: (i) there is the additional term $\left(1+\frac{b}{a+\|z-z_0\|_2}\right)$ in the condition, and (ii) the complementary subspace $\mathscr{V}$ for planar flows is invariant in z, while for radial flows $\mathscr{V}(z) = \mathbf{span}\{z-z_0\}^\perp$ is dependent on z. Next, we show that a radial flow cannot transform between Gaussian distributions with different covariance matrices, an even stronger result than **Corollary** A.9.1.

Corollary A.10.2 ($\mathscr{N} \nrightarrow \mathscr{N}$)**.** *Let $p \sim \mathscr{N}(0,\Sigma_p), q \sim \mathscr{N}(0,\Sigma_q)$ be two Gaussian distributions on $\mathbb{R}^d$. If there exists a radial flow f on $\mathbb{R}^d$ such that $p = f\#q$, then $\Sigma_q = \Sigma_p$.*

Proof of Theorem A.10.1

Proof. By standard algebra, it can be shown that the Jacobian of f is given by

$$J_f(z) = \left(1+\frac{b}{a\|z-z_0\|_2}\right)I - \frac{b(z-z_0)(z-z_0)^\top}{(a+\|z-z_0\|_2)^2\|z-z_0\|_2}.$$

Therefore, its determinant is

$$\det J_f(z) = \left(1+\frac{b}{a+\|z-z_0\|_2}\right)^d - \left(1+\frac{b}{a+\|z-z_0\|_2}\right)^{d-1}\frac{b\|z-z_0\|_2}{(a+\|z-z_0\|_2)^2}.$$

Notice that if $\|z-z_0\|_2$ does not change then $\det J_f(z)$ remains the same. Therefore, for any $z \neq z_0$, any direction $w^\perp \in \mathbf{span}\{z-z_0\}^\perp$ and small positive real number r, we have

$$\det J_f(z+rw^\perp) - \det J_f(z) = \mathscr{O}(r^2).$$

$$f(z+rw^\perp) - f(z) = \left(1+\frac{b}{a+\|z-z_0\|_2}\right)rw^\perp + \mathscr{O}(r^2).$$

By the change-of-variable formula in (1.1)

$$p(f(z)) = \frac{q(z)}{|\det J_f(z)|},\ p(f(z+rw^\perp)) = \frac{q(z+rw^\perp)}{|\det J_f(z+rw^\perp)|}.$$

For r small, $q(z)$ is continuous and positive in $B_r(z)$, so $p(f(z+rw^{\perp}))/q(z) = \mathscr{O}(1)$. Therefore,

$$\frac{q(z+rw^{\perp})}{q(z)} = \frac{p(f(z+rw^{\perp}))}{p(f(z))} + \mathscr{O}(r^2).$$

By taking the logarithm, multiplying $1/r$, and letting $r \to 0$ on both sides, we have that

$$\left(1+\frac{b}{a+\|z-z_0\|_2}\right)(\nabla_z \log p(f(z)))^{\top} w^{\perp} = (\nabla_z \log q(z))^{\top} w^{\perp}.$$

Therefore,

$$\left(1+\frac{b}{a+\|z-z_0\|_2}\right)\nabla_z \log p(f(z)) - \nabla_z \log q(z)$$

is parallel to $z - z_0$. $\square$

Proof of Corollary A.10.2

Proof. Let the radial flow be $f(z) = z + \frac{b}{a+\|z-z_0\|_2}(z-z_0)$ with $b \neq 0$. For conciseness, we write $v_x = 1 + \frac{b}{a+\|x\|_2}$ for any $x \in \mathbb{R}^d$. Now, we assign $x = z - z_0$ and solve the topology matching condition in **Theorem** A.10.1. By standard algebra, we obtain for any $x \in \mathbb{R}^d \setminus \{0\}$, if $x^{\top} w^{\perp} = 0$, then

$$\left(v_x(z_0+v_x x)^{\top}\Sigma_p^{-1} - (x+z_0)^{\top}\Sigma_q^{-1}\right) w^{\perp} = 0.$$

This indicates that

$$(v_x^2\Sigma_p^{-1} - \Sigma_q^{-1})x + (v_x\Sigma_p^{-1} - \Sigma_q^{-1})z_0$$

is parallel to x (or equal to 0). By applying the same analysis to $-x$, we have

$$(v_x^2\Sigma_p^{-1} - \Sigma_q^{-1})(-x) + (v_x\Sigma_p^{-1} - \Sigma_q^{-1})z_0.$$

is parallel to x (or equal to 0). Adding these two vectors, we have $(v_x\Sigma_p^{-1} - \Sigma_q^{-1})z_0$ is parallel to x (or equal to 0) for any $x \in \mathbb{R}^d \setminus \{0\}$. As a result, z_0 is the origin, and $(v_x^2\Sigma_p^{-1} - \Sigma_q^{-1})x$ is parallel to x. The only possibility to this claim is that $v_x^2\Sigma_p^{-1} - \Sigma_q^{-1}$ is a multiple of the identity matrix for any $x \in \mathbb{R}^d \setminus \{0\}$. Since v_x varies as x changes, both Σ_p and Σ_q are multiple of the identity matrix: $\Sigma_p = \kappa_p I$, $\Sigma_q = \kappa_q I$.

Next, we apply the results above to the change-of-variable equation in (1.2) of radial flow. By standard algebra, we have for any $z \in \mathbb{R}^d \setminus \{0\}$,

$$\frac{1}{2}\log\kappa_p + \frac{v_z^2 z^\top z}{2\kappa_p} = \frac{1}{2}\log\kappa_q + \frac{z^\top z}{2\kappa_q} + \log|\det J_f(z)|.$$

Notice that as $\|z\|_2 \to \infty$, the left-hand-side is equal to $\frac{\|z\|_2^2}{2\kappa_p} + \frac{b}{\kappa_p}\|z\|_2 + o(\|z\|_2)$, while the right-hand-side is equal to $\frac{\|z\|_2^2}{2\kappa_q} + \mathscr{O}(1)$. Then, b must be 0, which means that f is the identity map, and p, q are identical. $\square$

A.11 Proof of Lemma 1.5.2

Proof. According to (1.1), for any distribution q' on $\mathbb{R}^d$, we have $(f\#q')(f(z)) = \frac{q'(z)}{|\det J_f(z)|}$. By letting $y = f(z), z = f^{-1}(y)$, we have

$$\begin{aligned}
\int_{\mathbb{R}^d} |p(y) - (f\#q')(y)| dy &= \int_{\mathbb{R}^d} \left| p(y) - \frac{q'(z)}{|\det J_f(z)|} \right| dy \\
&= \int_{\mathbb{R}^d} \left| p(f(z)) - \frac{q'(z)}{|\det J_f(z)|} \right| |\det J_f(z)| dz \\
&= \int_{\mathbb{R}^d} \big| |\det J_f(z)| p(f(z)) - q'(z) \big| dz.
\end{aligned}$$

By the triangular inequality, we have

$$\int_{\mathbb{R}^d} |p(z) - q'(z)| dz - \int_{\mathbb{R}^d} \big| |\det J_f(z)| p(f(z)) - q'(z) \big| dz \le \int_{\mathbb{R}^d} \big| |\det J_f(z)| p(f(z)) - p(z) \big| dz.$$

By taking the supremum over q', we finish the proof. $\square$

A.12 Proof of Theorem 1.5.5

Lemma A.12.1. *Let $f(z) = z + uh(w^\top z + b)$ be a c_h-local planar flow. If $p(z) \propto \exp(-\|z\|_2^\tau)$, then*

$$\int_{\mathbb{R}^d} \frac{\|w\|_2 p(z)}{1 + |w^\top z + b|} dz = \mathscr{O}\left((\log d)^{\frac{1}{\tau}} d^{-\left(\frac{1}{\tau} - \frac{1}{2}\right)}\right).$$

Lemma A.12.2. *Let $f(z) = z + uh(w^\top z + b)$ be a c_h-local planar flow. If*

$$p(z) \propto \begin{cases} \exp(-d) & \|z\|_2 \le d^{\frac{1}{\tau}} \\ \exp(-\|z\|_2^\tau) & \|z\|_2 > d^{\frac{1}{\tau}} \end{cases},$$

then

$$\int_{\mathbb{R}^d} (\Delta_{c_h} p|_z) \, dz = \mathscr{O}\left(d^{-\left(\frac{1}{\tau} - 1\right)}\right).$$

Proof. Let $f(z) = z + uh(w^\top z + b)$ be a c_h-local planar flow. According to **Lemma** 1.5.2 and the fact that $|u^\top w h'(w^\top z + b))| < 1$, we have

$$\begin{aligned} \mathscr{L}(p, f) \le \hat{\mathscr{L}}(p, f) &= \int_{\mathbb{R}^d} \left| |\det J_f(z)| p(f(z)) - p(z) \right| dz \\ &= \int_{\mathbb{R}^d} \left| (1 + u^\top w h'(w^\top z + b)) p(z + uh(w^\top z + b)) - p(z) \right| dz. \end{aligned}$$

Now we define

$$\Delta_s p|_z = \sup_{\|\delta\| \le s} |p(z + \delta) - p(z)|.$$

Then, since $\forall x \in \mathbb{R}, |h(x)| \le c_h, |h'(x)| \le c_h/(1+|x|)$, we have

$$
\begin{aligned}
&\hat{\mathscr{L}}(p,f)\\
&= \int_{\mathbb{R}^d} \left| (1+u^\top w h'(w^\top z+b))(p(z+uh(w^\top z+b)) - p(z)) + u^\top w h'(w^\top z+b) p(z) \right| dz\\
&\le \int_{\mathbb{R}^d} \Big(|u^\top w h'(w^\top z+b)| p(z) + (1+c_h)|p(z+uh(w^\top z+b)) - p(z)| \Big) dz\\
&\le \int_{\mathbb{R}^d} \left(\frac{c_h}{1+|w^\top z+b|} |u^\top w| p(z) + (1+c_h)\Delta_{\|u\|_2 c_h} p|_z \right) dz\\
&\le c_h \int_{\mathbb{R}^d} \frac{\|w\|_2 p(z)}{1+|w^\top z+b|} dz + (1+c_h)\int_{\mathbb{R}^d} (\Delta_{c_h} p|_z) dz.
\end{aligned}
$$

Finally, using **Lemma** A.12.1 and **Lemma** A.12.2 and setting $\varepsilon = \frac{1}{2}\|p-q\|_1 = \Theta(1)$, we finish the proof. □

Proof of Lemma A.12.1

Proof. For conciseness, we denote $p(z)$ by $p(r)$ for any z such that $\|z\|_2 = r \ge 0$. That is, $p(r) \propto \exp(-r^\tau)$. The outline of the proof is: (i) simplify the expression by showing $b = 0$, (ii) rewrite the expression in polar coordination system, and (iii) apply bounds from Gamma functions to obtain the result.

Step 1: *simplification by solving w and b.* First of all, we have $\|w\|_2/(1+|w^\top z+b|) = 1/(1/\|w\|_2 + |\tilde{w}^\top z+b|)$, where $\tilde{w} = w/\|w\|_2$. Thus, to make the integration largest, $\|w\|_2$ should equal to 1. Since $p(z)$ is symmetric, any direction of w yields the same result. Therefore, we set $w = e_d = (0,\cdots,0,1)^\top \in \mathbb{R}^d$.

Next, we show $b = 0$. To maximize the integration, we have

$$
\frac{\partial}{\partial b}\int_{\mathbb{R}^d} \frac{p(z)}{1+|z_d+b|} dz = 0.
$$

The partial derivative is equal to

$$\begin{aligned}
&\int_{\mathbb{R}^d} -\frac{sgn(z_d+b)}{(1+|z_d+b|)^2}p(z)dz \\
&= \int_{\mathbb{R}^{d-1}} dz_{1:d-1}\left(\int_{-\infty}^{-b}\frac{p(z)}{(1+|z_d+b|)^2}dz_d - \int_{-b}^{\infty}\frac{p(z)}{(1+|z_d+b|)^2}dz_d\right) \\
&= \int_{\mathbb{R}^{d-1}} dz_{1:d-1}\left(\int_{-\infty}^{0}\frac{p(z-be_d)}{(1+|z_d|)^2}dz_d - \int_{0}^{\infty}\frac{p(z-be_d)}{(1+|z_d|)^2}dz_d\right) \\
&= \int_{\mathbb{R}^{d-1}} dz_{1:d-1}\int_{0}^{\infty}\frac{p(z+be_d)-p(z-be_d)}{(1+z_d)^2}dz_d.
\end{aligned}$$

If $b>0$, then $\|z+be_d\|_2 > \|z-be_d\|_2$, so $p(z+be_d)-p(z-be_d)<0$. Similarly, if $b<0$, then $p(z+be_d)-p(z-be_d)>0$. Therefore, we conclude $b=0$, and our objective becomes

$$\int_{\mathbb{R}^d}\frac{p(z)}{1+|z_d|}dz.$$

Step 2: *rewriting in polar coordinates.* Let $r=\|z\|_2$, then z can be expressed by polar coordinates in the following form:

$$\begin{cases}
z_1 & = r\sin\theta_1\sin\theta_2\cdots\sin\theta_{d-1} \\
z_2 & = r\cos\theta_1\sin\theta_2\cdots\sin\theta_{d-1} \\
z_3 & = r\cos\theta_2\sin\theta_3\cdots\sin\theta_{d-1} \\
\vdots & \vdots \\
z_{d-1} & = r\cos\theta_{d-1}\sin\theta_{d-1} \\
z_d & = r\cos\theta_{d-1}
\end{cases}, \ \theta_1\in[0,2\pi), \theta_i\in[0,\pi), 2\le i\le d-1.$$

The determinant of the Jacobian matrix of this transformation is given below [Muleshkov and Nguyen, 2017]:

$$\det J_d = (-1)^{d-1}r^{d-1}\prod_{k=2}^{d-1}\sin^{k-1}\theta_k.$$

Therefore, we have

$$\begin{aligned}
&\int_{\mathbb{R}^d} \frac{p(z)}{1+|z_d|} dz \\
&= \int_0^\infty dr \int_0^{2\pi} d\theta_1 \int_0^\pi d\theta_2 \cdots \int_0^\pi d\theta_{d-1} \left(\frac{p(r)}{1+|r\cos\theta_{d-1}|} r^{d-1} \prod_{k=2}^{d-1} \sin^{k-1}\theta_k \right) \\
&= \left(2\pi \prod_{k=2}^{d-2} \int_0^\pi \sin^{k-1}\theta_k d\theta_k \right) \times \int_0^\infty dr \int_0^\pi d\theta_{d-1} \left(\frac{p(r)}{1+|r\cos\theta_{d-1}|} r^{d-1} \sin^{d-2}\theta_{d-1} \right).
\end{aligned}$$

Step 3: *further simplification via normalization.* Since the integration of $p(z)$ over $\mathbb{R}^d$ is 1, we can write

$$\begin{aligned}
1 = \int_{\mathbb{R}^d} p(z) dz &= \int_0^\infty dr \int_0^{2\pi} d\theta_1 \int_0^\pi d\theta_2 \cdots \int_0^\pi d\theta_{d-1} \left(p(r) r^{d-1} \prod_{k=2}^{d-1} \sin^{k-1}\theta_k \right) \\
&= \left(2\pi \prod_{k=2}^{d-1} \int_0^\pi \sin^{k-1}\theta_k d\theta_k \right) \times \int_0^\infty p(r) r^{d-1} dr.
\end{aligned}$$

Furthermore, notice that

$$\int_0^\pi \sin^{d-2}\theta d\theta = \frac{\sqrt{\pi}\,\Gamma\left(\frac{d-1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)}.$$

According to Stirling's formula for Gamma functions, we have

$$\log\Gamma\left(\frac{d-1}{2}\right) - \log\Gamma\left(\frac{d}{2}\right) = \Theta\left(\frac{d-1}{2}\log\frac{d-1}{2} - \frac{d}{2}\log\frac{d}{2}\right) = \Theta\left(-\frac{1}{2}\log d\right).$$

We are then able to simplify the integration as

$$\int_{\mathbb{R}^d} \frac{p(z)}{1+|z_d|} dz = \frac{\int_0^\infty dr \int_0^\pi d\theta_{d-1} \left(\frac{p(r)}{1+|r\cos\theta_{d-1}|} r^{d-1} \sin^{d-2}\theta_{d-1} \right)}{\int_0^\infty p(r) r^{d-1} dr} \cdot \Theta(\sqrt{d}).$$

Step 4: *applying inequalities for two cases.*

- When $r \le d$, we use

$$\int_0^\pi \frac{\sin^{d-2}\theta}{1+|r\cos\theta|} d\theta \le \int_0^\pi \sin^{d-2}\theta d\theta = \Theta(d^{-\frac{1}{2}}).$$

- When $r > d$, we use

$$\int_0^\pi \frac{\sin^{d-2}\theta}{1+|r\cos\theta|}d\theta \le \int_0^\pi \frac{\sin\theta}{1+|r\cos\theta|}d\theta = \frac{2\log(1+r)}{r}.$$

Then, we have

$$\int_{\mathbb{R}^d} \frac{p(z)}{1+|z_d|}dz = \frac{\int_0^d p(r)r^{d-1}dr}{\int_0^\infty p(r)r^{d-1}dr} + \Theta(\sqrt{d}) \cdot \frac{\int_d^\infty p(r)r^{d-2}\log(1+r)dr}{\int_0^\infty p(r)r^{d-1}dr}. \tag{A.1}$$

Step 5: *final computation.* By applying $p(r) \propto \exp(-r^\tau)$, we are able to prove the following bounds.

- For the first term in (A.1), let the incomplete Gamma function be

$$\gamma(a,x) = \int_0^x t^{a-1}e^{-t}dt.$$

The incomplete Gamma function can be upper bounded below [Neuman, 2013]:

$$\gamma(a,x) \le \frac{x^a(1+ae^{-x})}{a^2}.$$

Therefore, we could bound the first term as

$$\begin{aligned}
\frac{\int_0^d p(r)r^{d-1}dr}{\int_0^\infty p(r)r^{d-1}dr} &= \frac{\int_0^d e^{-r^\tau}r^{d-1}dr}{\int_0^\infty e^{-r^\tau}r^{d-1}dr} \\
(\text{let } s = r^\tau) &= \frac{\frac{1}{\tau}\int_0^{d^\tau} e^{-s}s^{\frac{d}{\tau}-1}ds}{\frac{1}{\tau}\int_0^\infty e^{-s}s^{\frac{d}{\tau}-1}ds} \\
&= \gamma\left(\frac{d}{\tau}, d^\tau\right) \Big/ \Gamma\left(\frac{d}{\tau}\right) \\
&= \mathcal{O}\left(\frac{\tau^2 d^{d-2} + \tau d^{d-1}e^{-d^\tau}}{\sqrt{d}\left(\frac{d-\tau}{\tau e}\right)^{\frac{d}{\tau}-1}}\right) \\
&= \mathcal{O}\left((\tau e)^{\frac{d}{\tau}} d^{-\left(\frac{3}{2}+\frac{d}{\tau}-d\right)}\right).
\end{aligned}$$

- For the second term in (A.1), we let β satisfy $\log(1+d) = d^{\beta}$. Then, $\log(1+r) \leq r^{\beta}$ when $r > d$, and $\beta \to 0$ as d goes to infinity. Thus, we obtain

$$\begin{aligned}
\frac{\int_d^\infty p(r) r^{d-2} \log(1+r) dr}{\int_0^\infty p(r) r^{d-1} dr} &\leq \frac{\int_d^\infty p(r) r^{d+\beta-2} dr}{\int_0^\infty p(r) r^{d-1} dr} \\
&\leq \frac{\int_0^\infty p(r) r^{d+\beta-2} dr}{\int_0^\infty p(r) r^{d-1} dr} \\
&= \mathscr{O}\left(\frac{\Gamma\left(\frac{d+\beta-1}{\tau}\right)}{\Gamma\left(\frac{d}{\tau}\right)}\right) \\
&= \mathscr{O}\left(d^{-\frac{1-\beta}{\tau}}\right) \\
&= \mathscr{O}\left((\log d)^{\frac{1}{\tau}} d^{-\frac{1}{\tau}}\right).
\end{aligned}$$

In conclude, the first term in (A.1) is exponential in $1/d$ while the second term is polynomial in $1/d$. Thus, we finish the proof. □

Proof of Lemma A.12.2

Proof. Similar to the notations in the proof of **Lemma** A.12.2, for $r \geq 0$, we denote $p(z)$ by $p(r)$ for any z such that $\|z\|_2 = r$.

Suppose

$$p(r) \propto \tilde{p}(r) = \begin{cases} \exp(-d) & r \leq d^{\frac{1}{\tau}} \\ \exp(-r^{\tau}) & r > d^{\frac{1}{\tau}} \end{cases}.$$

One can check that $p(r)$ is continuous on $\mathbb{R}$. First, we compute the integration in polar coordinates. Following by steps 2-3 in the proof of **Lemma** A.12.1, we obtain

$$\int_{\mathbb{R}^d} (\Delta_{c_h} p|_z)\, dz = \frac{\int_0^\infty (\Delta_{c_h} p|_r)\, r^{d-1} dr}{\int_0^\infty p(r) r^{d-1} dr} = \frac{\int_0^\infty (\Delta_{c_h} \tilde{p}|_r)\, r^{d-1} dr}{\int_0^\infty \tilde{p}(r) r^{d-1} dr}.$$

Next, we show that $(\Delta_{c_h} \tilde{p}|_r) / \tilde{p}(r) = \mathscr{O}\left(d^{-\left(\frac{1}{\tau}-1\right)}\right)$. We split the rest of the proof into 3 cases.

- When $r \leq d^{1/\tau} - c_h$, $\Delta_{c_h}\tilde{p}|_r = 0$.

- When $r \geq d^{1/\tau}$, the second derivative of $\tilde{p}(r)$ is strictly positive, so

$$\Delta_{c_h}\tilde{p}|_r \leq c_h|\tilde{p}'(r)| = c_h \tau r^{\tau-1}\exp(-r^\tau).$$

  Therefore,

$$\frac{\Delta_{c_h}\tilde{p}|_r}{\tilde{p}(r)} \leq c_h \tau r^{\tau-1} \leq c_h \tau d^{-\left(\frac{1}{\tau}-1\right)}.$$

- When $d^{1/\tau} - c_h < r < d^{1/\tau}$, we have

$$\Delta_{c_h}\tilde{p}|_r \leq \Delta_{c_h}\tilde{p}|_{d^{1/\tau}} \leq c_h|\tilde{p}'(d^{1/\tau})| = c_h \tau d^{-\left(\frac{1}{\tau}-1\right)}\exp(-d).$$

  Since $\tilde{p}(r) = \exp(-d)$ in this case, we obtain

$$\frac{\Delta_{c_h}\tilde{p}|_r}{\tilde{p}(r)} \leq c_h \tau d^{-\left(\frac{1}{\tau}-1\right)}.$$

Summing these up, we finish the proof. □

A.13 Proof of Theorem 1.5.6

Proof. Let f be any Householder flow. According to **Lemma** 1.5.2 and the fact that $\det J_f(z) = -1$ for any $z \in \mathbb{R}^d$, we have

$$\mathscr{L}(p,f) \leq \hat{\mathscr{L}}(p,f) = \int_{\mathbb{R}^d} |p(f(z)) - p(z)| dz.$$

Since a Householder matrix does not change the ℓ_2 norm of a vector, we have

$$|p(f(z)) - p(z)| \leq \sup_{\|x\|_2 = \|y\|_2 = \|z\|_2} |p(x) - p(y)|.$$

Now, we rewrite the integration in polar coordinates (see step 2 in the proof of **Lemma** A.12.1), and we obtain that

$$\hat{\mathscr{L}}(p,f) \le \left(2\pi \prod_{k=2}^{d-1} \int_0^{\pi} \sin^{k-1}\theta_k d\theta_k\right) \times \int_0^{\infty} r^{d-1} \sup_{\|x\|_2=\|y\|_2=r} |p(x)-p(y)| dr.$$

First of all,

$$2\pi \prod_{k=2}^{d-1} \int_0^{\pi} \sin^{k-1}\theta_k d\theta_k = \left(2\pi \prod_{k=2}^{d-1} \frac{\sqrt{\pi}\Gamma\left(\frac{k}{2}\right)}{\Gamma\left(\frac{k+1}{2}\right)}\right) = \frac{2\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}\right)}.$$

Next, we bound $\sup_{\|x\|_2=\|y\|_2=r} |p(x)-p(y)|$ for $r>0$. Let $\Sigma = I+S$. According to the Courant-Fischer theorem (Chapter 5.2.2. (4), [Lütkepohl, 1996]),

$$\begin{aligned} \max_{\|z\|_2=r} z^\top \Sigma^{-1} z &= r^2 \lambda_{max}(\Sigma^{-1}) &= \frac{r^2}{\lambda_{min}(\Sigma)}, \\ \min_{\|z\|_2=r} z^\top \Sigma^{-1} z &= r^2 \lambda_{min}(\Sigma^{-1}) &= \frac{r^2}{\lambda_{max}(\Sigma)}. \end{aligned}$$

Therefore,

$$\sup_{\|x\|_2=\|y\|_2=r} |p(x)-p(y)| = \frac{\exp\left(-\frac{r^2}{2\lambda_{max}(\Sigma)}\right) - \exp\left(-\frac{r^2}{2\lambda_{min}(\Sigma)}\right)}{(2\pi)^{\frac{d}{2}}\sqrt{\det\Sigma}}.$$

Then. we obtain

$$\begin{aligned} \int_0^{\infty} r^{d-1} \sup_{\|x\|_2=\|y\|_2=r} |p(x)-p(y)| dr &= \frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{\det\Sigma}} \int_0^{\infty} r^{d-1} \exp\left(-\frac{r^2}{2\lambda_{max}(\Sigma)}\right) dr \\ &\quad - \frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{\det\Sigma}} \int_0^{\infty} r^{d-1} \exp\left(-\frac{r^2}{2\lambda_{min}(\Sigma)}\right) dr \\ &= \frac{2^{\frac{d}{2}-1}\Gamma\left(\frac{d}{2}\right)}{(2\pi)^{\frac{d}{2}}\sqrt{\det\Sigma}} \left(\lambda_{max}(\Sigma)^{\frac{d}{2}} - \lambda_{min}(\Sigma)^{\frac{d}{2}}\right). \end{aligned}$$

Combining these computations, we have

$$\mathscr{L}(p,f) \le \frac{2\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}\right)} \cdot \frac{2^{\frac{d}{2}-1}\Gamma\left(\frac{d}{2}\right)}{(2\pi)^{\frac{d}{2}}\sqrt{\det\Sigma}} \left(\lambda_{max}(\Sigma)^{\frac{d}{2}} - \lambda_{min}(\Sigma)^{\frac{d}{2}}\right) = \frac{\lambda_{max}(\Sigma)^{\frac{d}{2}} - \lambda_{min}(\Sigma)^{\frac{d}{2}}}{\sqrt{\det\Sigma}}.$$

Since $\det\Sigma$ is equal to the product of all eigenvalues of Σ, we have

$$\begin{aligned}\hat{\mathscr{L}}(p,f) &\le \left(\frac{\lambda_{max}(\Sigma)}{\lambda_{min}(\Sigma)}\right)^{\frac{d}{2}} - 1\\ &= \left(\frac{\lambda_{max}(S)+1}{\lambda_{min}(S)+1}\right)^{\frac{d}{2}} - 1\\ &= \left(1+\frac{\lambda_{max}(S)-\lambda_{min}(S)}{\lambda_{min}(S)+1}\right)^{\frac{d}{2}} - 1.\end{aligned}$$

According to the Gershgorin Circle Theorem, the absolute value of each eigenvalue of S does not exceed $\max_{1\le i\le d}\sum_{j=1}^{d}|S_{ij}| \le d^{-(1+\kappa)}$. Therefore,

$$\begin{aligned}\hat{\mathscr{L}}(p,f) &\le \left(1+\frac{2d^{-(1+\kappa)}}{1-d^{-(1+\kappa)}}\right)^{\frac{d}{2}} - 1\\ &= \mathscr{O}\left(\frac{2d^{-(1+\kappa)}}{1-d^{-(1+\kappa)}}\cdot\frac{d}{2}\right)\\ &= \mathscr{O}\left(d^{-\kappa}\right).\end{aligned}$$

Finally, by setting $\varepsilon = \frac{1}{2}\|p-q\|_1 = \Theta(1)$, we finish the proof. □

A.14 Experiments for Theorem 1.4.1

In Figure A.3, we plot two examples that illustrate **Theorem** 1.4.1. In each example, we plot the surface of q and its transformed distribution $p = f\#q$, where f is a ReLU planar flow. The four peaks of q are marked as red points, and their mapped locations on the surface of p are also marked as red. As illustrated, the mapped locations still correspond to the peaks of p, which is consistent with **Theorem** 1.4.1 because both $\nabla_z \log q(z)$ and $\nabla_z \log p(f(z))$ are zero vectors.

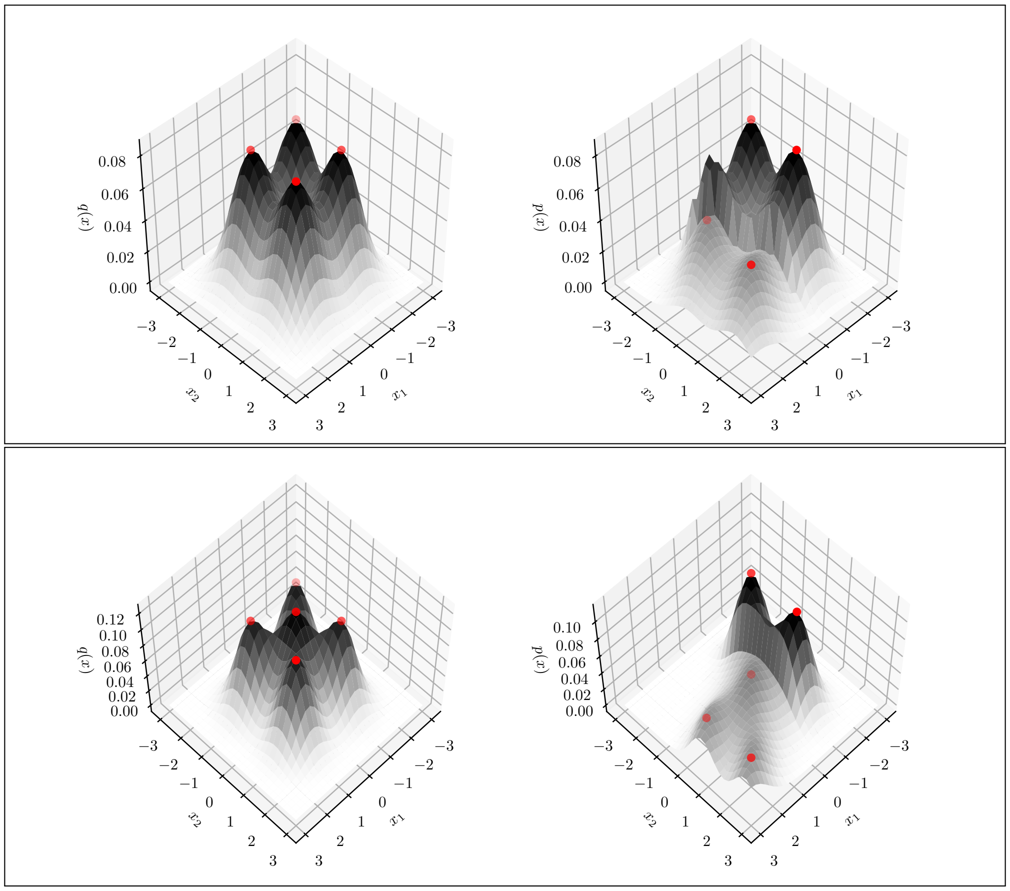


Figure A.3. Two examples that illustrate **Theorem** 1.4.1. Each example includes the surface plot of q (left) – a mixture of Gaussian distribution, and $p = f\#q$ (right) – the transformed distribution of q. The red points correspond to the peaks of q and their mapped points.

A.15 Experiments for Theorem 1.4.4

In Figure A.4, we plot four examples that illustrate **Theorem** 1.4.4. In each example, we plot the surface of q, its transformed distribution $p = f\#q$ where f is a planar flow with non-linearity tanh, and the value $\log p(f(x)) - \log q(x)$. As illustrated, the results are consistent with **Theorem** 1.4.4 because the gradient of $\log p(f(x)) - \log q(x)$ is parallel to some constant vector as indicated by applying **Theorem** 1.4.4 to single planar flow.

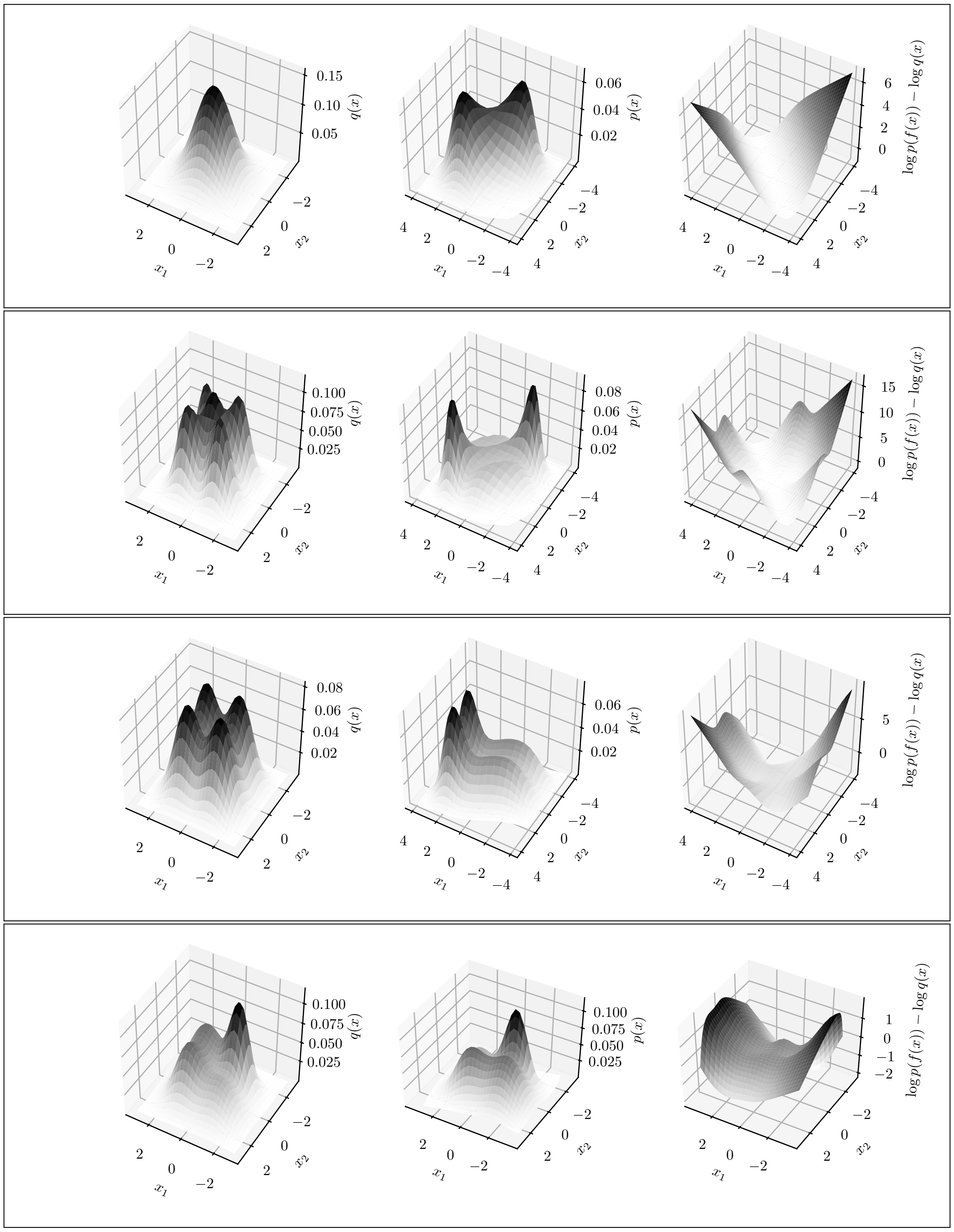


Figure A.4. For examples that illustrate **Theorem** 1.4.4. Each example includes the surface plot of q (left) – a mixture of Gaussian distribution, $p = f\#q$ (middle) – the transformed distribution of q with a planar flow, and $\log p(f(x)) - \log q(x)$ (right).

Appendix B

Universal Approximation of Residual Flows in Maximum Mean Discrepancy

B.1 Proof of Lemma 2.4.3

Proof. According to (2.7) and the chain rule,

$$\begin{aligned}
\Delta_1(q,p;\hat{f}_\varepsilon) &= 2\cdot\mathbb{E}_{z\sim q}\Big(\psi(p,q)^\top J_\phi(z)^\top \hat{f}_\varepsilon(z)\Big)\\
&= 2\varepsilon\cdot\mathbb{E}_{z\sim q}\Big(\psi(p,q)^\top J_\phi(z)^\top \nabla g(z)\Big)\\
&= 2\varepsilon\cdot\mathbb{E}_{z\sim q}\Big(\psi(p,q)^\top J_\phi(z)^\top J_\phi(z)\psi(p,q)\Big)\\
&\geq 2\varepsilon\cdot\min_{z\in\mathbb{R}^d}\Big(\psi(p,q)^\top J_\phi(z)^\top J_\phi(z)\psi(p,q)\Big)\\
\big(\mathrm{MMD}(q,p)^2=\|\psi(p,q)\|^2\big) &\geq 2\varepsilon\cdot\mathrm{MMD}(q,p)^2\min_{z\in\mathbb{R}^d}\lambda_{\min}\Big(J_\phi(z)^\top J_\phi(z)\Big)\\
&\geq 2\varepsilon\cdot\mathrm{MMD}(q,p)^2\min_{z\in\mathbb{R}^d}\sigma^2_{\min}(J_\phi(z))\\
&\geq 2\varepsilon b\cdot\mathrm{MMD}(q,p)^2.
\end{aligned}$$

□

B.2 Proof of Lemma 2.4.4

Proof. For any $x, y \in \mathbb{R}^d$,

$$\begin{aligned}
\frac{\|\hat{f}_\varepsilon(y) - \hat{f}_\varepsilon(x)\|}{\|y-x\|} &= \frac{\varepsilon\sqrt{\sum_{i=1}^{d}\left(\sum_{j=1}^{d_\phi}(J_\phi(y) - J_\phi(x))_{ij}\psi(p,q)_j\right)^2}}{\|y-x\|} \\
&\leq \frac{\varepsilon\sqrt{\sum_{i=1}^{d}\left(\sum_{j=1}^{d_\phi} L_{\text{Jac}}\|y-x\|\psi(p,q)_j\right)^2}}{\|y-x\|} \\
&\leq \varepsilon\sqrt{d}L_{\text{Jac}}\|\psi(p,q)\|_1 \\
&\leq \varepsilon\sqrt{d\cdot d_\phi}L_{\text{Jac}}\|\psi(p,q)\|_2 \\
&= \varepsilon\sqrt{d\cdot d_\phi}L_{\text{Jac}}\text{MMD}(q,p).
\end{aligned}$$

Therefore, by taking the supreme over the left-hand-side, we have the Lipschitz constant of $\hat{f}_\varepsilon$ is upper bounded by the right-hand-side. □

B.3 Proof of Theorem 2.4.5

Proof. Let $r > 0$ and $\varepsilon = r/N$. Define

$$D_n(r) = \text{MMD}((\mathbf{Id} + f_n)\circ\cdots\circ(\mathbf{Id} + f_1)\#q_{\text{source}}, p_{\text{target}})^2$$

where each

$$f_i(z) = \varepsilon J_\phi(z)\psi(p_{\text{target}}, (\mathbf{Id} + f_{i-1})\circ\cdots\circ(\mathbf{Id} + f_1)\#q_{\text{source}}).$$

Note that each f_i is exactly the $\hat{f}_\varepsilon$ in **Definition** 2.4.2 for $q = (\mathbf{Id} + f_{i-1})\circ\cdots\circ(\mathbf{Id} + f_1)\#q_{\text{source}}$ and $p = p_{\text{target}}$.

By **Lemma** 2.4.3,

$$D_n(r) \leq \left(1 - 2b\frac{r}{N} + \mathcal{O}\left(\frac{r^2}{N^2}\right)\right)D_{n-1}(r).$$

Therefore,

$$\begin{aligned} D_N(r) &\leq \prod_{n=1}^{N}\left(1-2b\frac{r}{N}+\mathscr{O}\left(\frac{r^2}{N^2}\right)\right)\mathrm{MMD}(q_{\mathrm{source}},p_{\mathrm{target}})^2 \\ &\leq \left(e^{-2br}+\mathscr{O}\left(\frac{r^2}{N}\right)\right)\mathrm{MMD}(q_{\mathrm{source}},p_{\mathrm{target}})^2. \end{aligned}$$

For small $\delta > 0$, we choose

$$r=\frac{1}{2b}\log\frac{2}{\delta},\ N=\Theta\left(\frac{r^2}{\delta}\right)=\Theta\left(\frac{1}{\delta}\left(\log\frac{1}{\delta}\right)^2\right).$$

Then, we have

$$D_N(r)\leq \delta\cdot\mathrm{MMD}(q_{\mathrm{source}},p_{\mathrm{target}})^2.$$

Note that

$$\varepsilon=\frac{r}{N}=\Theta\left(\frac{\delta}{r}\right)=\Theta\left(\frac{\delta}{\log\frac{1}{\delta}}\right).$$

Therefore, by **Lemma** 2.4.4, when δ is small enough, the Lipschitz constant of each f_n is less than $\frac{1}{2}$. □

B.4 Proof of Lemma 2.5.1

Proof. According to (2.1) and (2.5),

$$\Delta(q,p;\hat{f}_\varepsilon)=\mathbb{E}_{z\sim q,x\sim q}(K(z,x)-K(z+\hat{f}_\varepsilon(z),x+\hat{f}_\varepsilon(x)))+2\mathbb{E}_{z\sim q,x\sim p}(K(z+\hat{f}_\varepsilon(z),x)-K(z,x))$$

There is a closed-form expression for $\Delta_2(q,p;\hat{f}_\varepsilon)$. According to the remainder of multivariate Taylor polynomials, there exist two maps $\xi_1,\xi_2:\mathbb{R}^d\to(0,1)$ such that

$$\begin{aligned} \Delta_2(q,p;\hat{f}_\varepsilon) = &\quad \mathbb{E}_{z\sim q}\mathbb{E}_{x\sim p}\hat{f}_\varepsilon(z)^\top[\nabla^2_{zz}K(z+\xi_1(z)\hat{f}_\varepsilon(z),x+\xi_1(x)\hat{f}_\varepsilon(x))]\hat{f}_\varepsilon(z) \quad (=:\Delta_2^{(1)}) \\ &-\mathbb{E}_{z\sim q}\mathbb{E}_{x\sim q}\hat{f}_\varepsilon(z)^\top[\nabla^2_{zz}K(z+\xi_1(z)\hat{f}_\varepsilon(z),x+\xi_1(x)\hat{f}_\varepsilon(x))]\hat{f}_\varepsilon(z) \quad (=:-\Delta_2^{(2)}) \\ &-\mathbb{E}_{z\sim q}\mathbb{E}_{x\sim q}\hat{f}_\varepsilon(z)^\top[\nabla^2_{zx}K(z+\xi_2(z)\hat{f}_\varepsilon(z),x+\xi_2(x)\hat{f}_\varepsilon(x))]\hat{f}_\varepsilon(x) \quad (=:-\Delta_2^{(3)}) \\ = &\quad \Delta_2^{(1)}-\Delta_2^{(2)}-\Delta_2^{(3)}. \end{aligned}$$

First, we bound $|\Delta_2^{(1)} - \Delta_2^{(2)}|$. Define

$$\psi' := (\mathbb{E}_{x\sim p} - \mathbb{E}_{x\sim q})\phi(x + \xi_1(x)\hat{f}_\varepsilon(x)) = \psi(p,q) + \hat{\psi}_\varepsilon.$$

Since ϕ is L_{feat}-Lipschitz and $0 < \xi_1(x) < 1$, we have

$$\begin{aligned}
\|\hat{\psi}_\varepsilon\| &\leq \sup_{x\in\mathbb{R}^d} L_{\text{feat}}\xi_1(x)\|\hat{f}_\varepsilon(x)\| \\
&\leq \sup_{x\in\mathbb{R}^d} L_{\text{feat}}\varepsilon\|J_\phi(x)\psi(p,q)\| \\
&\leq \sup_{x\in\mathbb{R}^d} L_{\text{feat}}\varepsilon\sigma_{\max}(J_\phi(x))\|\psi(p,q)\| \\
&\leq \varepsilon L_{\text{feat}}\sqrt{B}\|\psi(p,q)\|.
\end{aligned}$$

Therefore,

$$\|\psi'\| \leq (1 + \varepsilon L_{\text{feat}}\sqrt{B})\|\psi(p,q)\|.$$

For any $z', v \in \mathbb{R}^d$,

$$\begin{aligned}
\left|v^\top[\nabla^2_{zz}(\phi(z')^\top\psi')]v\right| &= \left|\sum_{i=1}^{d_\phi}\psi'_i v^\top\nabla^2\phi_i(z')v\right| \\
&\leq \|\psi'\|\|v\|^2\sqrt{\sum_{i=1}^{d_\phi}\max\lambda(\nabla^2\phi_i(z'))^2} \\
&\leq \sqrt{d_\phi}C\|\psi'\|\|v\|^2.
\end{aligned}$$

By letting $z' = z + \xi_1(z)\hat{f}_\varepsilon(z)$ and $v = \hat{f}_\varepsilon(z)$, we have

$$\begin{aligned}
|\Delta_2^{(1)} - \Delta_2^{(2)}| &= \left|\mathbb{E}_{z\sim q}v^\top[\nabla^2_{zz}(\phi(z')^\top\psi')]v\right| \\
&\leq \varepsilon^2\|\psi(p,q)\|^2\sigma^2_{\max}(J_\phi)\sqrt{d_\phi}C\|\psi'\| \\
&\leq \varepsilon^2\|\psi(p,q)\|^3\sqrt{d_\phi}BC(1 + \varepsilon L_{\text{feat}}\sqrt{B}).
\end{aligned}$$

Next, we bound Δ_2^3. Observe that

$$\Delta_2^{(3)} = -\mathbb{E}_{z\sim q}\mathbb{E}_{x\sim q}\hat{f}_\varepsilon(z)^\top J_\phi(z+\xi_2(z)\hat{f}_\varepsilon(z))J_\phi(x+\xi_2(x)\hat{f}_\varepsilon(x))^\top \hat{f}_\varepsilon(x).$$

Therefore,

$$\begin{aligned}|\Delta_2^{(3)}| &\le \max_{z\in\mathbb{R}^d}\|\hat{f}_\varepsilon(z)\|^2 \max_{z\in\mathbb{R}^d}\|J_\phi(z)\|^2 \\ &\le \varepsilon^2\|\psi(p,q)\|^2 \max_{z\in\mathbb{R}^d}\sigma_{\max}^4(J_\phi(z)) \\ &\le \varepsilon^2\|\psi(p,q)\|^2 B^2.\end{aligned}$$

Combining these bounds, we have

$$|\Delta_2(q,p;\hat{f}_\varepsilon)| \le \varepsilon^2\cdot \mathrm{MMD}(q,p)^2\left(\|\psi(p,q)\|\sqrt{d_\phi}BC(1+\varepsilon L_{\text{feat}}\sqrt{B})+B^2\right).$$

□

B.5 Proof of Theorem 2.5.2

Proof. Let

$$q_0 = q_{\text{source}} \text{ and } q_m = (\mathbf{Id}+f_m)\circ\cdots\circ(\mathbf{Id}+f_1)\#q_{\text{source}},$$

where each

$$f_i(z) = \varepsilon J_\phi(z)\psi(p_{\text{target}},q_{i-1}).$$

Define $\psi_0 = \psi(p_{\text{target}},q_0)$ and assume $\|\psi(p_{\text{target}},q_m)\| \le \|\psi_0\|$ (which we will prove by induction). Note that

$$\Delta(q_m,p_{\text{target}};f_m) = \mathrm{MMD}(q_m,p_{\text{target}})^2 - \mathrm{MMD}(q_{m+1},p_{\text{target}})^2.$$

According to **Lemma** 2.4.3 and **Lemma** 2.5.1, we have

$$\begin{aligned}
&\frac{\Delta(q_m, p_{\text{target}}; f_m)}{\text{MMD}(q_m, p_{\text{target}})^2} \\
&\geq 2b\varepsilon - \left(\|\psi(p_{\text{target}}, q_m)\|\sqrt{d_\phi}BC + B^2\right)\varepsilon^2 - \|\psi(p_{\text{target}}, q_m)\|\sqrt{d_\phi}B^{\frac{3}{2}}CL_{\text{feat}}\varepsilon^3 \\
&\geq 2b\varepsilon - \left(\|\psi_0\|\sqrt{d_\phi}BC + B^2\right)\varepsilon^2 - \|\psi_0\|\sqrt{d_\phi}B^{\frac{3}{2}}CL_{\text{feat}}\varepsilon^3.
\end{aligned}$$

When

$$\varepsilon \leq \varepsilon_\Delta = \min\left(\frac{b}{2\left(\|\psi_0\|\sqrt{d_\phi}BC + B^2\right)}, \sqrt{\frac{b}{2\|\psi_0\|\sqrt{d_\phi}B^{\frac{3}{2}}CL_{\text{feat}}}}\right),$$

we have

$$\frac{\Delta(q_m, p_{\text{target}}; f_m)}{\text{MMD}(q_m, p_{\text{target}})^2} \geq b\varepsilon.$$

Next, by **Lemma** 2.4.4, in order to satisfy the Lipschitz condition, we require

$$\varepsilon \leq \frac{1}{2\sqrt{d \cdot d_\phi}L_{\text{Jac}}\|\psi(p_{\text{target}}, q_m)\|}.$$

This is satisfied when we assign

$$\varepsilon \leq \varepsilon_{\text{Lip}} := \frac{1}{2\sqrt{d \cdot d_\phi}L_{\text{Jac}}\|\psi_0\|}.$$

Now, we set $\varepsilon = \hat{\varepsilon} := \min(\varepsilon_\Delta, \varepsilon_{\text{Lip}})$. Then, we have

$$\text{MMD}(q_{m+1}, p_{\text{target}})^2 \leq (1 - b\hat{\varepsilon}) \cdot \text{MMD}(q_m, p_{\text{target}})^2,$$

which also implies $\|\psi(p_{\text{target}}, q_{m+1})\| \leq \sqrt{1 - b\hat{\varepsilon}}\|\psi_0\| \leq \|\psi_0\|$. Finally, in order to satisfy (2.4), we only need to take the number of residual blocks as

$$N = \frac{\log\frac{1}{\delta}}{\log\frac{1}{1-b\hat{\varepsilon}}}.$$

☐

Appendix C

Understanding Instance-based Interpretability of Variational Auto-Encoders

C.1 Omitted Proofs

C.1.1 Omitted Proofs in Section 3.3

Proof of (3.1)

Proof. By definition, we have

$$p_{k\mathrm{NN}}(z;X) = \frac{k}{NV_dR_k(z;X)^d}$$

and

$$p_{k\mathrm{NN}}(z;X_{-i}) = \frac{k}{(N-1)V_dR_k(z;X_{-i})^d}.$$

If x_i belongs to k-NN of z, then $R_k(z;X_{-i})$ is $R_{k+1}(z;X)$; otherwise, $R_k(z;X_{-i})$ is $R_k(z;X)$. The result follows by subtracting the logarithm of these two densities. □

Proof of (3.2)

Proof. By definition, we have

$$p_{\mathrm{KDE}}(z;X) = \frac{1}{N}\sum_{j=1}^{N} K_\sigma(z-x_j)$$

and

$$p_{\text{KDE}}(z; X_{-i}) = \frac{1}{N-1} \sum_{j \neq i}^{N} K_\sigma(z - x_j).$$

The result follows by subtracting the logarithm of these two densities. It is interesting to notice when $\max_i K_\sigma(z - x_i) \ll N \cdot p_{\text{KDE}}(z; X)$, we have

$$\sum_{i=1}^{N} \text{IF}_{X,\text{KDE}}(x_i, z) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\frac{K_\sigma(z - x_i)}{p_{\text{KDE}}(z; X)} - 1 \right) = 0.$$

□

Proof of (3.4)

Proof. By definition, we have

$$p_{\text{WS-GMM}}(z; X) = \frac{N_0}{N} \mathcal{N}(z; \mu_0, \sigma_0^2 I).$$

If $x_i \notin X_0$, then

$$p_{\text{WS-GMM}}(z; X_{-i}) = \frac{N_0}{N-1} \mathcal{N}(z; \mu_0, \sigma_0^2 I).$$

In this case, we have $\text{IF}_{X,\text{WS-GMM}}(x_i, z) = -\log(1 + 1/N)$.

If $x_i \in X_0$, then parameters μ_0 and σ_0 are to be modified to maximize likelihood estimates over $X_0 \setminus \{x_i\}$. Denote the modified parameters as μ_0' and σ_0'. Then, we have

$$p_{\text{WS-GMM}}(z; X_{-i}) = \frac{N_0 - 1}{N-1} \mathcal{N}(z; \mu_0', (\sigma_0')^2 I).$$

Next, we express μ_0' and σ_0' in terms of known variables. For conciseness, we let $v = z - \mu_0$ and $u = x_i - \mu_0$.

$$\begin{aligned} \mu_0' &= \frac{1}{N_0 - 1} \sum_{x \in X_0 \setminus \{x_i\}} x \\ &= \frac{N_0 \mu_0 - x_i}{N_0 - 1} \\ &= \mu_0 - \frac{u}{N_0 - 1}. \end{aligned}$$

$$
\begin{aligned}
(\sigma_0')^2 &= \frac{1}{(N_0-1)d} \sum_{x \in X_0 \setminus \{x_i\}} x^\top x - \frac{1}{d}(\mu_0')^\top \mu_0' \\
&= \frac{1}{(N_0-1)d} \left(\sum_{x \in X_0} x^\top x - x_i^\top x_i - (N_0-1) \left(\mu_0^\top \mu_0 - \frac{2u^\top \mu_0}{N_0-1} + \frac{u^\top u}{(N_0-1)^2} \right) \right) \\
&= \frac{1}{(N_0-1)d} \left(N_0 d \sigma_0^2 - x_i^\top x_i + \mu_0^\top \mu_0 + 2u^\top \mu_0 - \frac{u^\top u}{N_0-1} \right) \\
&= \frac{1}{(N_0-1)d} \left(N_0 d \sigma_0^2 - x_i^\top x_i - \frac{N_0 u^\top u}{N_0-1} \right) \\
&= \frac{N_0}{N_0-1} \sigma_0^2 - \frac{N_0 u^\top u}{(N_0-1)^2 d}.
\end{aligned}
$$

Then, we have

$$
\begin{aligned}
\log p_{\text{WS-GMM}}(z;X) &= \log \frac{N_0}{N} - \frac{d}{2} \log 2\pi - \frac{d}{2} \log \sigma_0^2 - \frac{1}{2\sigma_0^2}(z-\mu_0)^\top (z-\mu_0) \\
&= \log \frac{N_0}{N} - \frac{d}{2} \log 2\pi - \frac{d}{2} \log \sigma_0^2 - \frac{v^\top v}{2\sigma_0^2},
\end{aligned}
$$

and

$$
\begin{aligned}
\log p_{\text{WS-GMM}}(z;X_{-i}) &= \log \frac{N_0-1}{N-1} - \frac{d}{2} \log 2\pi - \frac{d}{2} \log(\sigma_0')^2 - \frac{1}{2{\sigma_0'}^2}(z-\mu_0')^\top (z-\mu_0') \\
&= \log \frac{N_0-1}{N-1} - \frac{d}{2} \log 2\pi - \frac{d}{2} \log \frac{N_0}{N_0-1} - \frac{d}{2} \log \sigma_0^2 \\
&\quad - \frac{d}{2} \log \left(1 - \frac{u^\top u}{(N_0-1) d \sigma_0^2} \right) \\
\left(z - \mu_0' = v + \tfrac{u}{N_0-1} \right) \qquad &\quad - \frac{(N_0-1)^2 v^\top v + 2(N_0-1) u^\top v + u^\top u}{2N_0 \left((N_0-1)\sigma_0^2 - \frac{1}{d} u^\top u \right)} \\
&= \log \frac{N_0-1}{N-1} - \frac{d}{2} \log 2\pi - \frac{d}{2N_0} - \frac{d}{2} \log \sigma_0^2 + \frac{u^\top u}{2N_0 \sigma_0^2} \\
&\quad - \frac{v^\top v}{2\sigma_0^2} - \frac{1}{2N_0 \sigma_0^2} \left(2u^\top v - v^\top v + \frac{v^\top v}{\sigma_0^2} \right) + \mathscr{O}\left(N_0^{-2} \right).
\end{aligned}
$$

Subtracting the above two equations, we have

$$
\begin{aligned}
\text{IF}_{X,\text{WS-GMM}}(x_i, z) &= \frac{1}{N_0} - \frac{1}{N} + \frac{d}{2N_0} + \frac{1}{2N_0 \sigma_0^2} \left(2u^\top v - v^\top v - u^\top u + \frac{v^\top v}{\sigma_0^2} \right) + \mathscr{O}\left(N_0^{-2} \right) \\
&= \frac{d+2}{2N_0} + \frac{1}{2N_0 \sigma_0^2} \left(\frac{\|z-\mu_0\|^2}{\sigma_0^2} - \|z - x_i\|^2 \right) - \frac{1}{N} + \mathscr{O}\left(N_0^{-2} \right).
\end{aligned}
$$

$\square$

C.1.2 Probabilistic Bound on Influence Estimates

Let $\{\xi_j\}_{j=1}^m$ be m i.i.d. samples drawn from $Q_{\psi^*}(\cdot|z)$ and $\{\xi_j'\}_{j=1}^m$ be m i.i.d. samples drawn from $Q_{\psi_{-i}^*}(\cdot|z)$. We can use the empirical influence $\hat{\mathrm{IF}}_{X,\mathrm{VAE}}^{(m)}(x_i,z)$ to estimate the true influence in (3.7), which is defined below:

$$\begin{aligned}\hat{\mathrm{IF}}_{X,\mathrm{VAE}}^{(m)}(x_i,z) &= \beta\left(\mathrm{KL}\left(Q_{\psi_{-i}^*}(\cdot|z)\|P_{\mathrm{latent}}\right) - \mathrm{KL}\left(Q_{\psi^*}(\cdot|z)\|P_{\mathrm{latent}}\right)\right) \\ &\quad -\tfrac{1}{m}\textstyle\sum_{j=1}^m\left(\log P_{\phi_{-i}^*}(z|\xi_j') - \log P_{\phi^*}(z|\xi_j)\right).\end{aligned} \tag{C.1}$$

The questions is, when can we guarantee the empirical influence score $\hat{\mathrm{IF}}_{X,\mathrm{VAE}}^{(m)}(x_i,z)$ is close to the true influence score $\mathrm{IF}_{X,\mathrm{VAE}}(x_i,z)$? We answer this question via an (ε,δ)-probabilistic bound: as long as m is larger than a function of ε and δ, then with probability at least $1-\delta$, the difference between the empirical and true influence scores is no more than ε. To introduce the theory, we first provide the following definition.

Definition C.1.1 (Polynomially-bounded functions)**.** *Let $f:\mathbb{R}^d\to\mathbb{R}^{d'}$. We say f is polynomially bounded by $\{a_c\}_{c=1}^C$ if for any $x\in\mathbb{R}^d$, we have*

$$\|f(x)\| \leq \sum_{c=1}^C a_c\|x\|^c. \tag{C.2}$$

We provide a useful lemma on polynomially-bounded functions below.

Lemma C.1.2. *The composition of polynomially bounded functions is polynomially bounded.*

Next, we show common neural networks are polynomially bounded in the following proposition.

Proposition C.1.3. *Let f be a neural network taking the following form:*

$$f(x) = \sigma_l(W_l \sigma_{l-1}(W_{l-1} \cdots \sigma_1(W_1 x) \cdots)). \tag{C.3}$$

If every activation function σ_j is polynomially bounded, then f is polynomially bounded.

With the above result, we state the (ε, δ)-probabilistic bound on influence estimates below.

Theorem C.1.4 (Error bounds on influence estimates)**.** *Let P and Q be two polynomially bounded networks. For any small $\varepsilon > 0$ and $\delta > 0$, there exists an $m = \Theta\left(\frac{1}{\varepsilon^2 \delta}\right)$ such that*

$$\text{Prob}\left(\left|\text{IF}_{X,\text{VAE}}(x_i, z) - \hat{\text{IF}}^{(m)}_{X,\text{VAE}}(x_i, z)\right| \geq \varepsilon\right) \leq \delta, \tag{C.4}$$

where the randomness is over all ξ_j and ξ_j'.

Proof of Lemma C.1.2

Proof. Let $f : \mathbb{R}^{m_0} \to \mathbb{R}^{m_1}$ be polynomially bounded by $\{a_c\}_{c=1}^{C_f}$ and $g : \mathbb{R}^{m_1} \to \mathbb{R}^{m_2}$ be polynomially bounded by $\{b_c\}_{c=1}^{C_g}$. Then, for any $x \in \mathbb{R}^{m_0}$,

$$\|f(x)\| \leq \sum_{c=1}^{C_f} a_c \|x\|^c,$$

and

$$\|g \circ f(x)\| \leq \sum_{c=1}^{C_g} b_c \|f(x)\|^c.$$

Therefore, we have

$$\|g \circ f(x)\| \leq \sum_{c=1}^{C_g} b_c \left(\sum_{c'=1}^{C_f} a_{c'} \|x\|^{c'}\right)^c.$$

This indicates that $g \circ f$ is polynomially bounded. $\square$

Proof of Proposition C.1.3

Proof. First, an affine transformation Wx is polynomially bounded because $\|Wx\| \leq \|W\| \cdot \|x\|$. Then, we show an element-wise transformation $\sigma(x)$ is polynomially bounded. Let σ be polynomially bounded by $\{a_c\}_{c=1}^{C}$. Then,

$$\|\sigma(x)\| \leq \frac{1}{\sqrt{d}} \left(\sum_{i=1}^{d} |\sigma(x_i)| \right)^2 \leq \frac{1}{\sqrt{d}} \left(\sum_{i=1}^{d} \left| \sum_{c=1}^{C} a_c |x_i|^c \right| \right)^2 \leq \frac{1}{\sqrt{d}} \left(\sum_{i=1}^{d} \left| \sum_{c=1}^{C} a_c \|x\|^c \right| \right)^2 .$$

By **Lemma** C.1.2, since f is a composition of polynomially bounded functions, we have f is polynomially bounded. $\square$

Proof of Theorem C.1.4

Proof. According to (3.7) and (C.1),

$$\begin{aligned} \mathrm{IF}_{X,\mathrm{VAE}}(x_i,z) - \hat{\mathrm{IF}}^{(m)}_{X,\mathrm{VAE}}(x_i,z) &= \frac{1}{m}\sum_{j=1}^{m} \log P_{\phi^*_{-i}}(z|\xi_j') - \mathbb{E}_{\xi \sim Q_{\psi^*_{-i}}(\cdot|z)} \log P_{\phi^*_{-i}}(z|\xi) \\ &\quad - \frac{1}{m}\sum_{j=1}^{m} \log P_{\phi^*}(z|\xi_j) + \mathbb{E}_{\xi \sim Q_{\psi^*}(\cdot|z)} \log P_{\phi^*}(z|\xi). \end{aligned}$$

If we have

$$\left| \mathbb{E}_{\xi \sim Q_{\psi^*}(\cdot|z)} \log P_{\phi^*}(z|\xi) - \frac{1}{m}\sum_{j=1}^{m} \log P_{\phi^*}(z|\xi_j) \right| \leq \frac{\varepsilon}{2}$$

and

$$\left| \mathbb{E}_{\xi \sim Q_{\psi^*_{-i}}(\cdot|z)} \log P_{\phi^*_{-i}}(z|\xi) - \frac{1}{m}\sum_{j=1}^{m} \log P_{\phi^*_{-i}}(z|\xi_j') \right| \leq \frac{\varepsilon}{2},$$

then $\left| \mathrm{IF}_{X,\mathrm{VAE}}(x_i,z) - \hat{\mathrm{IF}}^{(m)}_{X,\mathrm{VAE}}(x_i,z) \right| \leq \varepsilon$. Let ζ and ζ_j be i.i.d. standard Gaussian random variables for $j = 1, 2, \cdots, m$. First, we provide the probabilistic bound for the first inequality. Let $P = P_{\phi^*}$ and $Q = Q_{\psi^*}$. Then, we can reparameterize $\xi = \mu_Q(z) + \sigma_Q(z) \odot \zeta$ and $\xi_j = \mu_Q(z) + \sigma_Q(z) \odot \zeta_j$. Let

$$f(\zeta) = \|z - \mu_P(\mu_Q(z) + \sigma_Q(z) \odot \zeta)\|_2^2.$$

Since $\log P(z|\xi)$ is a constant α times $\|z-\mu_P(\xi)\|^2$ plus another constant, we have

$$\mathbb{E}_{\xi\sim Q(\cdot|z)}\log P(z|\xi) - \frac{1}{m}\sum_{j=1}^{m}\log P(z|\xi_j) = \mathbb{E}_\zeta f(\zeta) - \frac{1}{m}\sum_{j=1}^{m} f(\zeta_j).$$

By Chebyshev's inequality,

$$\text{Prob}\left(\left|\mathbb{E}_\zeta f(\zeta) - \frac{1}{m}\sum_{j=1}^{m} f(\zeta_j)\right| \ge \frac{\varepsilon}{2}\right) \le \frac{4\text{Var}_\zeta f(\zeta)}{m\varepsilon^2} \le \frac{4\mathbb{E}_\zeta f(\zeta)^2}{m\varepsilon^2}.$$

By **Lemma** C.1.2, if P and Q are polynomially bounded, then f is polynomially bounded and so is f^2. Let

$$f(\zeta)^2 \le \sum_{c=1}^{C} a_c \|\zeta\|^c.$$

Then,

$$\begin{aligned}
\mathbb{E}_\zeta f(\zeta)^2 &= \int_{\mathbb{R}^d} \mathcal{N}(\zeta;0,I) f(\zeta)^2 d\zeta \\
&\le \frac{1}{(2\pi)^{\frac{d}{2}}}\sum_{c=1}^{C} a_c \int_{\mathbb{R}^d} \|\zeta\|^c e^{-\frac{\|\zeta\|^2}{2}} d\zeta \\
\text{(use polar coordinate)} &= \frac{\pi^{\frac{d-1}{2}}}{(2\pi)^{\frac{d}{2}}\Gamma\left(\frac{d+1}{2}\right)}\sum_{c=1}^{C} a_c \int_0^{+\infty} r^{c+d-1} e^{-\frac{r^2}{2}} dr \\
&= \frac{\pi^{\frac{d-1}{2}}}{(2\pi)^{\frac{d}{2}}\Gamma\left(\frac{d+1}{2}\right)}\sum_{c=1}^{C} a_c \int_0^{+\infty} (2r)^{\frac{c+d}{2}-1} e^{-r} dr \\
&= \frac{1}{2\sqrt{\pi}\Gamma\left(\frac{d+1}{2}\right)}\sum_{c=1}^{C} 2^{\frac{c}{2}} a_c \Gamma\left(\frac{c+d}{2}\right),
\end{aligned}$$

which is a constant. Therefore, there exists an $M_1 = \Theta\left(\frac{1}{\varepsilon^2\delta}\right)$ such that when $m \ge M_1$,

$$\text{Prob}\left(\left|\mathbb{E}_\zeta f(\zeta) - \frac{1}{m}\sum_{j=1}^{m} f(\zeta_j)\right| \ge \frac{\varepsilon}{2}\right) \le \frac{\delta}{2},$$

or

$$\text{Prob}\left(\left|\mathbb{E}_{\xi\sim Q_{\psi^*}(\cdot|z)}\log P_{\phi^*}(z|\xi) - \frac{1}{m}\sum_{j=1}^{m}\log P_{\phi^*}(z|\xi_j)\right| \ge \frac{\varepsilon}{2}\right) \le \frac{\delta}{2}.$$

Similarly, when $P = P_{\phi^*_{-i}}$ and $Q = Q_{\psi^*_{-i}}$, there exists an $M_2 = \Theta\left(\frac{1}{\varepsilon^2\delta}\right)$ such that when $m \geq M_2$,

$$\text{Prob}\left(\left|\mathbb{E}_{\xi\sim Q_{\psi^*_{-i}}(\cdot|z)}\log P_{\phi^*_{-i}}(z|\xi) - \frac{1}{m}\sum_{j=1}^{m}\log P_{\phi^*_{-i}}(z|\xi'_j)\right| \geq \frac{\varepsilon}{2}\right) \leq \frac{\delta}{2}.$$

Taking $m = \max(M_1, M_2) = \Theta\left(\frac{1}{\varepsilon^2\delta}\right)$, we have

$$\text{Prob}\left(\left|\text{IF}_{X,\text{VAE}}(x_i,z) - \hat{\text{IF}}^{(m)}_{X,\text{VAE}}(x_i,z)\right| \geq \varepsilon\right) \leq \delta.$$

□

C.1.3 Derivation of (3.9)

$$\begin{aligned}
&\nabla_\phi \ell_\beta(x;\theta)\\
&= -\nabla_\phi \mathbb{E}_{\xi\sim Q_\psi(\cdot|x)}\log P_\phi(x|\xi)\\
&= -\mathbb{E}_{\xi\sim Q_\psi(\cdot|x)}\nabla_\phi \log P_\phi(x|\xi);\\
&\nabla_\psi \ell_\beta(x;\theta)\\
&= \nabla_\psi \int_\xi Q_\psi(\xi|x)\left(\beta\log\frac{Q_\psi(\xi|x)}{P_{\text{latent}}(\xi)} - \log P_\phi(x|\xi)\right)d\xi\\
&= \int_\xi \left[\nabla_\psi Q_\psi(\xi|x)\left(\beta\log\frac{Q_\psi(\xi|x)}{P_{\text{latent}}(\xi)} - \log P_\phi(x|\xi)\right) + \beta Q_\psi(\xi|x)\cdot\frac{\nabla_\psi Q_\psi(\xi|x)}{Q_\psi(\xi|x)}\right]d\xi\\
&= \int_\xi \nabla_\psi Q_\psi(\xi|x)\left(\beta + \beta\log\frac{Q_\psi(\xi|x)}{P_{\text{latent}}(\xi)} - \log P_\phi(x|\xi)\right)d\xi\\
&= \int_\xi \nabla_\psi Q_\psi(\xi|x)\left(\beta\log\frac{Q_\psi(\xi|x)}{P_{\text{latent}}(\xi)} - \log P_\phi(x|\xi)\right)d\xi\\
&= \mathbb{E}_{\xi\sim Q_\psi(\cdot|x)}\nabla_\psi \log Q_\psi(\xi|x)\left(\beta\log\frac{Q_\psi(\xi|x)}{P_{\text{latent}}(\xi)} - \log P_\phi(x|\xi)\right).
\end{aligned}$$

C.2 Additional Experiments and Details

C.2.1 Additional Results on Density Estimators in Section 3.3

The synthetic data of six clusters are illustrated in Fig. C.1. The sizes for cluster zero through five are 25, 15, 20, 25, 10, 5, respectively. Each cluster is drawn from a spherical Gaussian distribution. The standard errors are 0.5, 0.5, 0.4, 0.4, 0.5, 1.0, respectively.

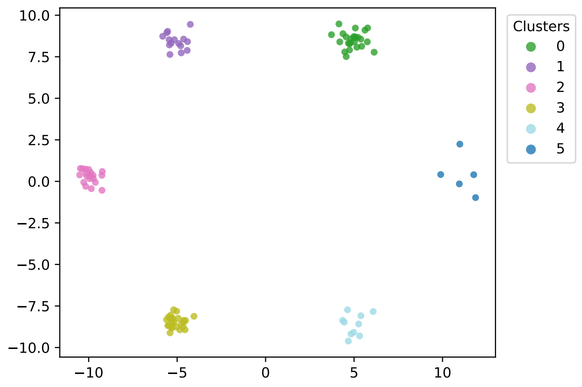
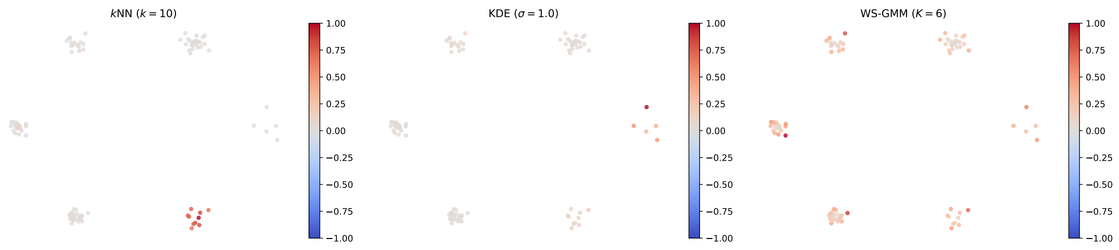
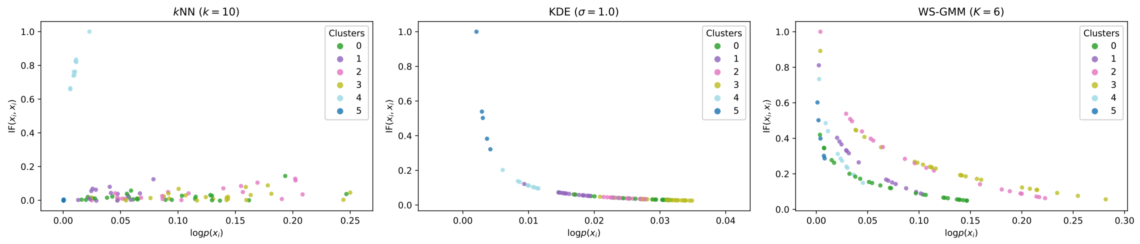


Figure C.1. Synthetic data of six clusters in Section 3.3.1.

We visualize the self influences of all data samples, and compare them to the log likelihood in Fig. C.2. For k-NN samples from cluster 4 have the highest self influences because the size of this cluster is exactly $k = 10$. For KDE samples in cluster 5 (which is the smallest cluster in terms of number of samples) have the highest self influences. The self influences strictly obey the reverse order of likelihood, which can be derived from (3.2). For GMM samples far away to cluster centers have high self influences, which can be derived from (3.5). Samples from clusters 2 and 3 generally have higher self influences because σ_2 and σ_3 are smaller than others.



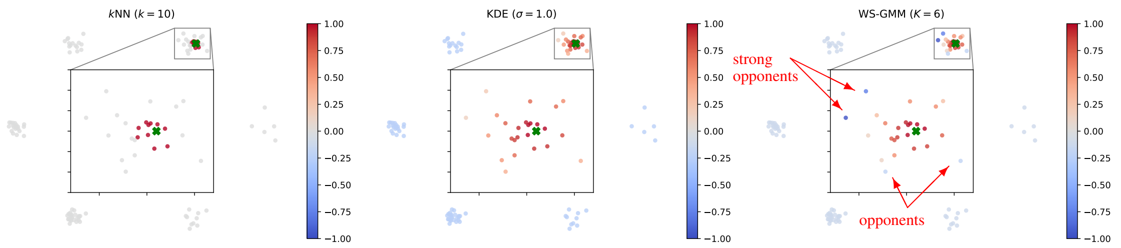
(a) Self influence scores of training samples in different methods. The high self influence samples in k-NN are from a cluster with exactly k samples; those in KDE are from the cluster with the smallest size; and those in GMM are far away to the center of the corresponding cluster.



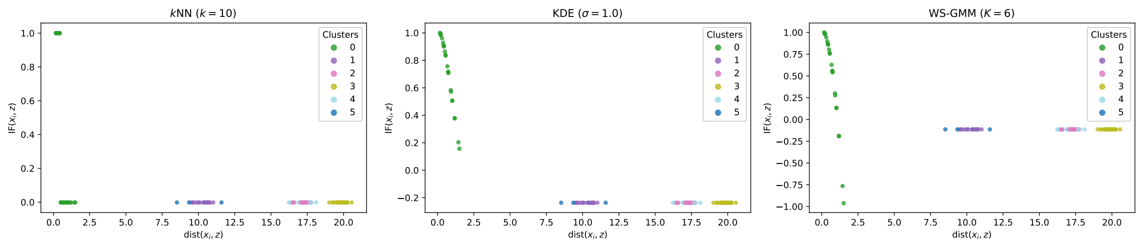
(b) Self influences versus log-likelihood. In k-NN only samples from cluster 4 (which has exatly k points) have large self influences. In KDE and GMM the self influences tend to decrease as the likelihood increases.

Figure C.2. Self influences in different density estimators.

We let z be a data point near the center of cluster 0. We then visualize influences of all data samples over z, and compare these influences to the distances between the samples and z in Fig. C.3. For k-NN the k nearest samples are strong proponents of z, and the rest have little influences over z. For KDE proponents of z are all samples from cluster 0, and the rest have slightly negative influences over z. The influences strictly obey the reverse order of distances to z, which can be derived from (3.2). For GMM, it is surprising that samples in the same cluster as z can be (even strong) opponents of z. This observation can be mathematically derived from (3.4). When $\|z-x_i\|^2 > (d+2)\sigma_0^2 + \|z-\mu_0\|^2/\sigma_0^2$, we have $\mathrm{IF}_{X,\text{WS-GMM}}(x_i,z) \lessapprox 0$. When d is large and z is sampled from the mixture $\mathcal{N}(\mu_0,\sigma_0^2 I)$, then with high probability, $\|z-\mu_0\|^2 \approx d\sigma_0^2$. Therefore, the influence of x_i over z is negative with high probability when $\|z-x_i\|^2 \gtrapprox (1+\sigma_0^2)d + 2\sigma_0^2$.



(a) Influences of training samples over a test sample z (shown as ✖) in different methods. In all cases the strongest proponents are nearest samples. In GMM, surprisingly, samples from the same cluster can be strong opponents.



(b) Influences of training samples over z versus distances to z.

Figure C.3. Influences of training samples over a test sample z in different methods. In all methods the strongest proponents are nearest samples. Surprisingly, in GMM strong opponents are also nearby samples.

C.2.2 Details of Experiments in Section 3.5

Datasets.

We conduct experiments on MNIST and CIFAR-10 (shortened as CIFAR). Because it is challenging to train a successful VAE model on the entire CIFAR dataset, we also train VAE models on each subclass of CIFAR. There are ten subclasses in total, which we name CIFAR_0 though CIFAR_9, and each subclass contains 5k training samples. In the main text, CIFAR-Airplane is CIFAR_0. All CIFAR images are resized to 64×64.

In Section 3.5.1, we examine influences of all training samples over the first 128 training samples in the trainset. In the unsupervised data cleaning application in Section 3.5.2, the extra samples are the first 1k samples from EMNIST and CelebA, respectively. In Section 3.5.3, we randomly select 128 samples from the test set and compute influences of all training samples over these test samples.

Models and hyperparameters.

For MNIST, our VAE models are composed of multilayer perceptrons as described by Meehan et al. [2020]. In these experiments we let $\beta = 4$ and $d_{\text{latent}} = 128$ unless clearly specified.

For CIFAR and CIFAR subclasses, our VAE models are composed of convolution networks as described by Higgins et al. [2016]. We let $\beta = 2, d_{\text{latent}} = 128$ for CIFAR and $\beta = 2, d_{\text{latent}} = 64$ for CIFAR subclass unless clearly specified.

We use stochastic gradient descent to train these VAE models based on a public implementation. [1] In all experiments, we set the batch size to be 64 and train for 1.5M iterations. The learning rates are 1×10^{-4} in MNIST experiments and 3×10^{-4} in CIFAR experiments.

VAE-TracIn settings.

In all experiments, we average the loss for $m = 16$ times when computing VAE-TracIn according to (3.10). We use $C = 30$ (evenly distributed) checkpoints to compute influences in Section 3.5.1 and Section 3.5.3. We use the last checkpoint to compute self influences in Section 3.5.2. For visualization purpose, all self influences are normalized to $[0, 1]$, all influences over test data are normalized to $[-1, 1]$, and all distributions are normalized to densities.

[1]https://github.com/1Konny/Beta-VAE (MIT License)

C.2.3 Sanity Checks for VAE-TracIn

As a sanity check for VAE-TracIn, we examine the frequency that a training sample is the most influential one among all training samples over itself, or formally, the frequency that $i = \arg\max_{1 \le i' \le N} \text{VAE-TracIn}(x_{i'}, x_i)$. Due to computational limits we examine the first 128 training samples. The results for MNIST, CIFAR, and CIFAR subclasses are reported in Table 3.1 and Table C.1. These results indicate that VAE-TracIn can find the most influential training samples in MNIST and CIFAR subclasses.

Table C.1. Sanity check on the frequency of a training sample being more influential than other samples over itself. Results on CIFAR subclasses (CIFAR$_i$, $0 \le i \le 9$) are reported.

i	0	1	2	3	4	5	6	7	8	9
Top-1 scores	1.00	1.00	0.99	1.00	0.98	1.00	1.00	1.00	1.00	1.00

We next conduct an additional sanity check for VAE-TracIn on MNIST. For two training samples $x_{\text{major}} = x_i$ and $x_{\text{minor}} = x_j$, we synthesize a new sample $\hat{x} = \alpha x_{\text{major}} + (1-\alpha)x_{\text{minor}}$, where $\alpha = 0.75$. Then, $\hat{x}$ is very similar to x_{major} but the minor component x_{minor} can also be visually recognized. For each pair of different labels, we obtain x_{major} and x_{minor} by randomly picking one sample within each class. The entire 90 samples are shown in Fig. C.4. We expect a perfect instance-based interpretation should indicate x_i and x_j have very high influences over $\hat{x}$. We report quantiles of the 90 ranks of x_{major} and x_{minor} sorted by influences over $\hat{x}$ in Table C.2. We then compute the frequency that x_{major} is exactly the strongest proponent of $\hat{x}$, namely the top-1 score of the major component. We compare the results to a baseline model that finds nearest neighbours in a perceptual autoencoder latent space (PAE-NN, [Meehan et al., 2020, Zhang et al., 2018]). Although VAE-TracIn does not detect x_{major} as well as PAE-NN, it still has reasonable results, and performs much better in detecting x_{minor}. The results indicate that VAE-TracIn can capture potentially influential components.

We then evaluate the approximation accuracy of VAE-TracIn. We randomly select 128 test samples for each CIFAR-subclass and save 1000 checkpoints at the first 1000 iterations. We

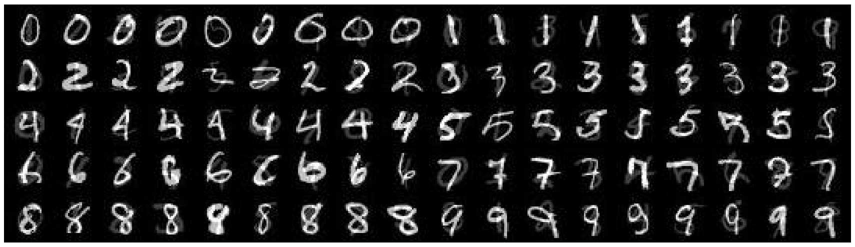


Figure C.4. Synthesized samples $\hat{x} = \alpha x_{\text{major}} + (1 - \alpha)x_{\text{minor}}$, where $\alpha = 0.75$.

Table C.2. Quantiles of ranks of x_{major} and x_{minor} sorted by influences over $\hat{x}$, and top-1 scores of the major components. We let 0 to be the highest rank.

Method	rank(x_{major}) quantiles			Top-1	rank(x_{minor}) quantiles		
	25%	50%	75%	scores	25%	50%	75%
PAE-NN	0	0	1	0.633	6943	13405	29993
VAE-TracIn ($d_{\text{latent}} = 64$)	0	2	146	0.422	1097	5206	10220
VAE-TracIn ($d_{\text{latent}} = 96$)	0	1	44	0.456	1372	4283	15319
VAE-TracIn ($d_{\text{latent}} = 128$)	0	1	18	0.467	1203	6043	13873

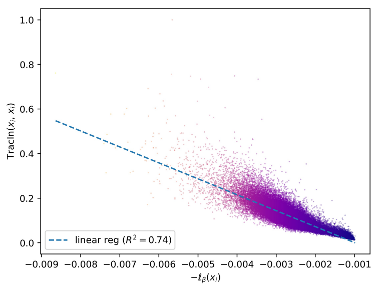
compute the total Pearson correlation coefficients between (1) the VAE-TracIn scores and (2) the loss change of all training samples on the selected test samples between consecutive checkpoints, similar to Appendix G by Pruthi et al. [2020]. The results are reported in Table C.3, which indicate high correlations.

Table C.3. Pearson correlation coefficients ρ between VAE-TracIn scores and loss change on each CIFAR-subclass (CIFAR$_i$, $0 \leq i \leq 9$). A coefficient $= 1$ means perfect correlation.

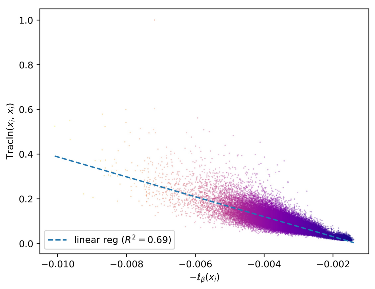
i	0	1	2	3	4	5	6	7	8	9
ρ	0.882	0.910	0.895	0.934	0.828	0.865	0.908	0.852	0.861	0.613

C.2.4 Self Influences (MNIST)

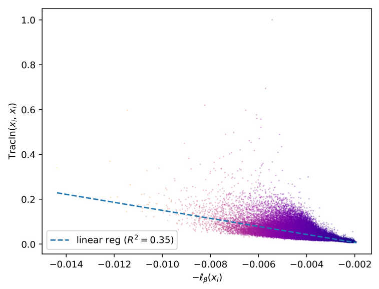
In MNIST experiments, we compare self influences and losses across different hyperparameters. The scatter and density plots are shown in Fig. C.5. We fit linear regression models to these points and report R^2 scores. In all settings high self influence samples have large losses. We find R^2 is larger under high latent dimensions or smaller β.



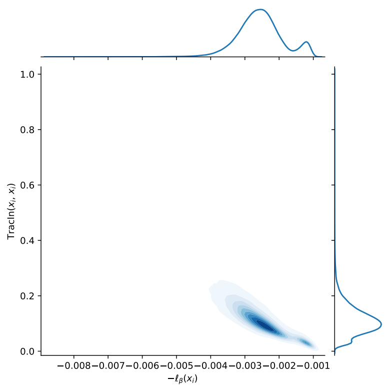
(a) $\beta = 1$, $d_{\text{latent}} = 128$



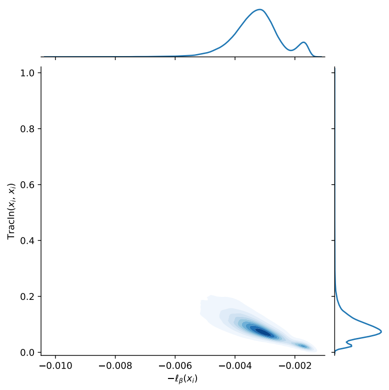
(b) $\beta = 2$, $d_{\text{latent}} = 128$



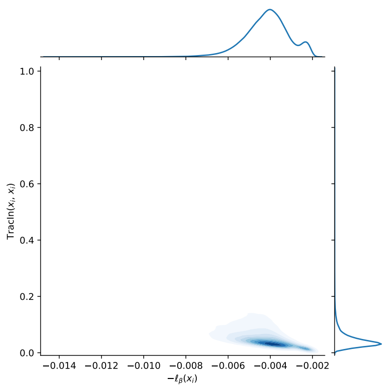
(c) $\beta = 4$, $d_{\text{latent}} = 128$



(d) $\beta = 1$, $d_{\text{latent}} = 128$



(e) $\beta = 2$, $d_{\text{latent}} = 128$

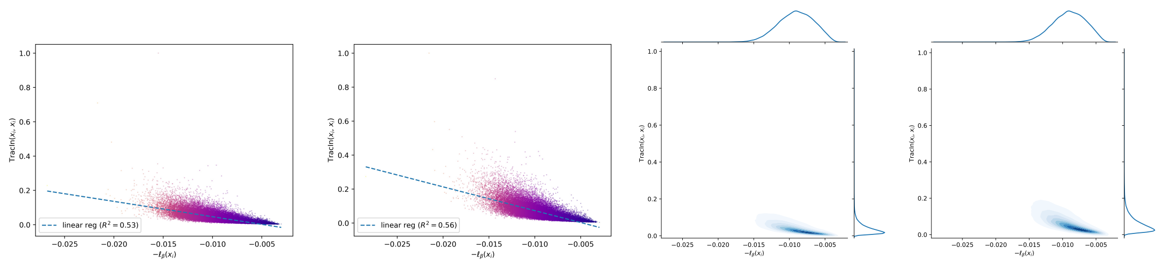


(f) $\beta = 4$, $d_{\text{latent}} = 128$

Figure C.5. Scatter and density plots of self influences versus negative losses of all training samples in MNIST. The linear regressors show that high self influence samples have large losses.

C.2.5 Self Influences (CIFAR)

In CIFAR and CIFAR subclass experiments, we compare self influences and losses across different hyperparameters. Similar to Appendix C.2.4, we demonstrate scatter and density plots, and report R^2 scores of linear regression models fit to these data. Comparisons on CIFAR are shown in Fig. C.6. In both settings high self influence samples have large losses.

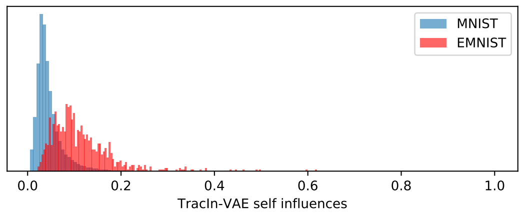


(a) $d_{\text{latent}} = 64$ **(b)** $d_{\text{latent}} = 128$ **(c)** $d_{\text{latent}} = 64$ **(d)** $d_{\text{latent}} = 128$

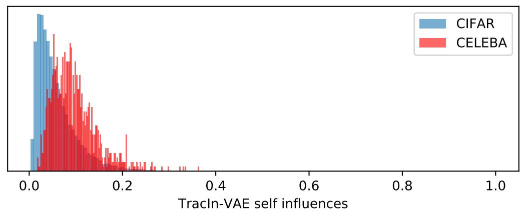
Figure C.6. Scatter and density plots of self influences versus negative losses of all training samples in CIFAR. The linear regressors show that high self influence samples have large losses.

C.2.6 Application to Unsupervised Data Cleaning

We plot the distribution of self influences of extra samples (EMNIST or CelebA) and original samples (MNIST or CIFAR) in Fig. C.7. We plot the detection curves in Fig. C.8, where the horizontal axis is the fraction of all samples checked when they are sorted in the self influence order, and the vertical axis is the fraction of extra samples found. The area under these detection curves (AUC) are reported in Table C.4. These experiments are repeated five times to reduce randomness. The results indicate that extra samples have higher self influences than original samples. This justifies the potential to apply VAE-TracIn to unsupervised data cleaning.

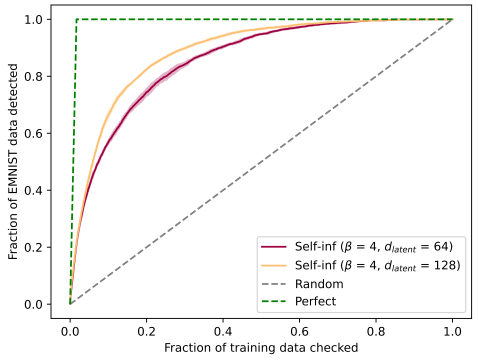


(a) EMNIST versus MNIST

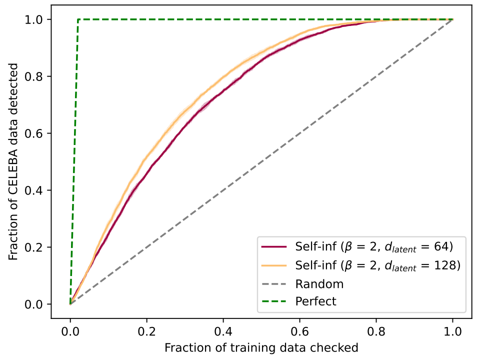


(b) CelebA versus CIFAR

Figure C.7. Distributions of self influences of 1k extra samples versus samples from the original dataset. Distributions are normalized as densities for better visualization. It is shown that extra samples have higher self influences.



(a) EMNIST versus MNIST



(b) CelebA versus CIFAR

Figure C.8. Detection curves (fraction of extra samples detected versus fraction of training data checked in the self influence order) with standard errors. It is shown that extra samples can be detected by sorting self influences.

In order to understand whether high self influence samples indeed harm the model, we

Table C.4. AUC of detection curves in Fig. C.8. The results indicate detection on the simple MNIST + EMNIST datasets is better than the more complicated CIFAR + CelebA datasets. In addition, a higher d_{latent} leads to slightly better AUC.

Original dataset	Extra samples	d_{latent}	AUC
MNIST	EMNIST	64	0.858±0.003
MNIST	EMNIST	128	0.887±0.002
CIFAR	CelebA	64	0.735±0.002
CIFAR	CelebA	128	0.760±0.001

remove a certain number of high self influence samples and retrain the β-VAE model under the MNIST + EMNIST setting. We report the results in Table C.5. We can observe consistent loss drop after deletion of high self influence samples.

Table C.5. Average loss of the MNIST test set when performing retraining after removing a certain number of high self influence samples under the MNIST + EMNIST setting.

# removed	0	4	8	16	32	64	128	256	512	1024
Loss ($\times 10^{-3}$)	4.24	4.21	4.18	4.19	4.18	4.21	4.17	4.17	4.18	4.18

C.2.7 Influences over Test Data (MNIST)

In Fig. C.9, we plot the distributions of influences of training i (red distributions) and non-i (blue distributions) over test i for label $i = 0, \cdots, 9$. For most labels including 0, 2, 4, 6, 7, and 9, the strongest proponents and opponents are very likely from the same class. For the rest of the labels including 1, 3, 5, and 8, the strongest opponents seem equally likely from the same or a different class.

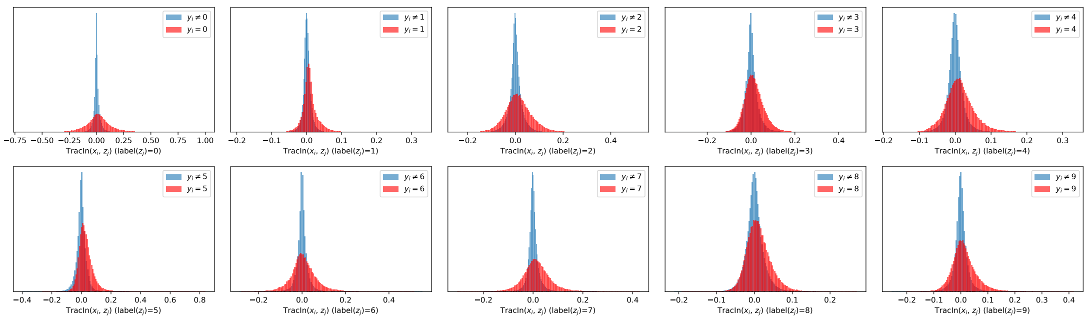
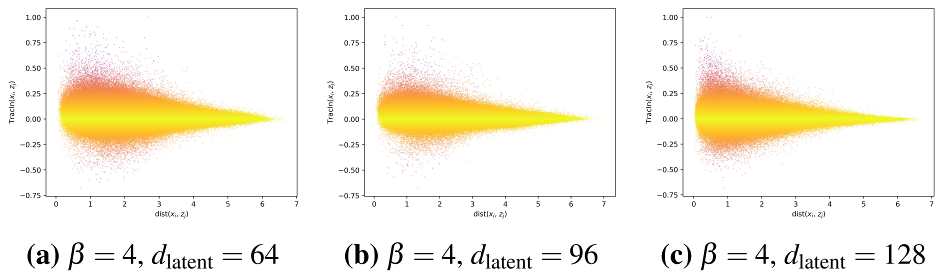


Figure C.9. Distributions of influences of training samples (x_i) over test samples (z_j). The red distributions are x_i in the same class as z_j, and the blue distributions are x_i in a different class as z_j. Many strongest opponents are from the same class as strongest proponents.

We then compare the influences of training over test samples to the distances between them in the latent space in Fig. C.10. We observe that both proponents and opponents are very close to test samples in the latent space, which indicates strong similarity between them.

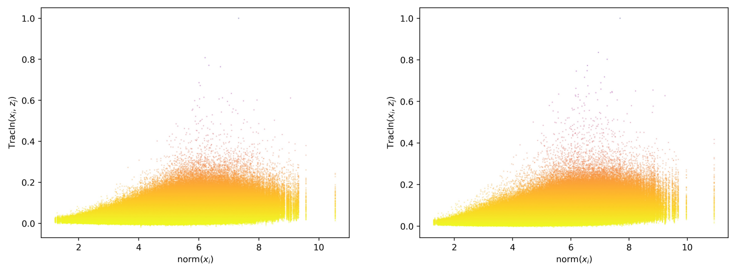


(a) $\beta = 4$, $d_{\text{latent}} = 64$ **(b)** $\beta = 4$, $d_{\text{latent}} = 96$ **(c)** $\beta = 4$, $d_{\text{latent}} = 128$

Figure C.10. Influences of training over test samples versus pairwise distances between them in the latent space. It is shown that both proponents and opponents are very close to test samples in the latent space.

C.2.8 Influences over Test Data (CIFAR)

We compare the influences of training over test samples to the norms of training samples in the latent space. Results for CIFAR are shown in Fig. C.11. We observe that strong proponents tend to have very large norms. This indicates they are high-contrast or very bright samples. This phenomenon occurs to CIFAR and all CIFAR subclasses.



(a) $\beta = 2, d_{\text{latent}} = 64$ **(b)** $\beta = 2, d_{\text{latent}} = 128$

Figure C.11. Influences of training samples over test samples (CIFAR) versus norms of training samples in the latent space. It is shown that strong proponents have large norms.

For 128 test samples in each CIFAR subclass, we report the statistics of the latent space norms of their strongest proponents, strongest opponents, and all training samples in Table C.6.

Table C.6. The means $\pm$ standard errors of latent space norms of training samples in CIFAR subclasses. Strong proponents tend to have large norms.

Dataset	top-0.1% strong proponents	top-0.1% strong opponents	all training samples
CIFAR$_0$	4.73 ± 0.78	4.26 ± 0.91	4.07 ± 0.83
CIFAR$_1$	5.30 ± 0.71	4.54 ± 0.65	4.65 ± 0.64
CIFAR$_2$	4.89 ± 0.78	4.18 ± 0.88	4.09 ± 0.93
CIFAR$_3$	5.09 ± 0.75	4.47 ± 0.78	4.42 ± 0.79
CIFAR$_4$	5.06 ± 0.72	3.96 ± 1.00	4.01 ± 0.89
CIFAR$_5$	5.25 ± 0.75	4.33 ± 0.94	4.54 ± 0.81
CIFAR$_6$	4.73 ± 0.66	3.99 ± 0.80	3.95 ± 0.82
CIFAR$_7$	5.11 ± 0.76	4.42 ± 0.73	4.43 ± 0.74
CIFAR$_8$	5.10 ± 0.72	4.19 ± 0.88	4.22 ± 0.84
CIFAR$_9$	5.48 ± 0.62	4.62 ± 0.69	4.79 ± 0.64

Appendix D

Data Redaction from Pre-trained GANs

D.1 Proof of Theorem 4.3.1 and Extension to f-GAN

Background of f-GAN [Nowozin et al., 2016].

Let ϕ be a convex, lower-semicontinuous function such that $\phi(1) = 0$. In f-GAN, the following ϕ-divergence is minimized:

$$D_\phi(P\|Q) = \int_{x\in\mathbb{R}^d} Q(x)\phi\left(\frac{P(x)}{Q(x)}\right)dx.$$

According to the variational characterization of ϕ-divergence [Nguyen et al., 2010],

$$D_\phi(P\|Q) = \sup_T \left[\mathbb{E}_{x\sim P}T(x) - \mathbb{E}_{x\sim Q}\phi^*(T(x))\right],$$

where the optimal T is obtained by $T = \phi'\left(\frac{P}{Q}\right)$.

The objective function (4.2) **corresponds to an f-GAN.**

Let $\alpha = \alpha_- + \alpha_+$. We can rewrite (4.2) as

$$L(G,D) = \alpha \cdot \mathbb{E}_{x\sim P}\log D(x) + (2-\alpha)\cdot \mathbb{E}_{x\sim Q}\log(1-D(x)),$$

where

$$P = \frac{\alpha_+}{\alpha} p_{\text{data}}|_{\bar{\Omega}} + \frac{\alpha_-}{\alpha} p_{\text{fake}}; \;\; Q = \frac{1-\alpha_+}{2-\alpha} p_{\text{data}}|_{\bar{\Omega}} + \frac{1-\alpha_-}{2-\alpha} p_{\text{fake}}.$$

Let

$$C = \alpha \log \alpha + (2-\alpha)\log(2-\alpha) - 2\log 2,$$

$$\phi(u) = (\alpha u)\log(\alpha u) - (\alpha u - \alpha + 2)\log(\alpha u - \alpha + 2) + (2-\alpha)\log(2-\alpha) - C.$$

Then, $\phi(1) = 0$, and $\phi''(u) = \frac{\alpha(2-\alpha)}{u(\alpha u - \alpha + 2)} > 0$ so ϕ is convex. Its convex conjugate function ϕ^* is

$$\phi^*(t) := \sup_u (ut - \phi(u)) = -(2-\alpha)\log\left(1 - e^{\frac{t}{\alpha}}\right) + C.$$

Let $T(x) = \alpha \log D(x)$. Then,

$$\max_D L(G,D) = \sup_T \left[\mathbb{E}_{x\sim P} T(x) - \mathbb{E}_{x\sim Q}\phi^*(T(x))\right] + C = D_\phi(P\|Q) + C.$$

Optimal D**.**

We have

$$\phi'(u) = \alpha \log \frac{\alpha u}{\alpha u - \alpha + 2}.$$

Therefore, the optimal discriminator is

$$\alpha \log D = \phi'\left(\frac{P}{Q}\right),$$

or

$$D = \frac{\alpha P}{\alpha P + (2-\alpha) Q} = \frac{\alpha_+ p_{\text{data}}|_{\bar{\Omega}} + \alpha_- p_{\text{fake}}}{p_{\text{data}}|_{\bar{\Omega}} + p_{\text{fake}}}.$$

Finally, the optimal discriminator in (4.3) is obtained by inserting (4.1) into the above equation.

Optimal G.

For conciseness, we let

$$P_1 = p_{\text{data}}|_{\bar{\Omega}}, P_2 = p_G, P_3 = p_\Omega,$$

$$\beta_1 = \frac{\alpha_+}{\alpha}, \beta_2 = \frac{\alpha_-\lambda}{\alpha}, \beta_3 = \frac{\alpha_-(1-\lambda)}{\alpha},$$

$$\gamma_1 = \frac{1-\alpha_+}{2-\alpha}, \gamma_2 = \frac{(1-\alpha_-)\lambda}{2-\alpha}, \gamma_3 = \frac{(1-\alpha_-)(1-\lambda)}{2-\alpha}.$$

Then, we have

$$P = \sum_{i=1}^{3} \beta_i P_i, Q = \sum_{i=1}^{3} \gamma_i P_i.$$

We also have

$$\frac{\beta_1}{\gamma_1} > \frac{\beta_2}{\gamma_2} = \frac{\beta_3}{\gamma_3}.$$

Because **supp** $(P_1) \cap$ **supp** (P_3) is the empty set, we have

$$\begin{aligned}
D_\phi(P\|Q) &= \int_{x\in\mathbb{R}^d} \left(\sum_{i=1}^{3} \gamma_i P_i\right) \phi\left(\frac{\sum_{i=1}^{3} \beta_i P_i}{\sum_{i=1}^{3} \gamma_i P_i}\right) dx \\
&= \int_{x\notin\Omega} (\gamma_1 P_1 + \gamma_2 P_2) \phi\left(\frac{\beta_1 P_1 + \beta_2 P_2}{\gamma_1 P_1 + \gamma_2 P_2}\right) dx \\
&+ \int_{x\in\Omega} (\gamma_2 P_2 + \gamma_3 P_3) \phi\left(\frac{\beta_2 P_2 + \beta_3 P_3}{\gamma_2 P_2 + \gamma_3 P_3}\right) dx
\end{aligned}$$

Let

$$\int_{x\in\Omega} P_2 dx = \eta.$$

We have

$$\int_{x\in\Omega} (\gamma_2 P_2 + \gamma_3 P_3) \phi\left(\frac{\beta_2 P_2 + \beta_3 P_3}{\gamma_2 P_2 + \gamma_3 P_3}\right) dx = (\gamma_2 \eta + \gamma_3)\, \phi\left(\frac{\beta_3}{\gamma_3}\right).$$

Let

$$\zeta = \frac{\beta_2(\gamma_1+\gamma_2)}{\gamma_2(\beta_1+\beta_2)}.$$

According to Jensen's inequality,

$$\begin{aligned}
&\int_{x\notin\Omega}(\gamma_1 P_1+\gamma_2 P_2)\phi\left(\frac{\beta_1 P_1+\beta_2 P_2}{\gamma_1 P_1+\gamma_2 P_2}\right)dx \\
&= (\gamma_1+\gamma_2(1-\zeta\eta))\int_{x\notin\Omega}\left(\frac{\gamma_1 P_1+\gamma_2 P_2}{\gamma_1+\gamma_2(1-\zeta\eta)}\right)\phi\left(\frac{\beta_1 P_1+\beta_2 P_2}{\gamma_1 P_1+\gamma_2 P_2}\right)dx \\
&\geq (\gamma_1+\gamma_2(1-\zeta\eta))\phi\left(\int_{x\notin\Omega}\frac{\beta_1 P_1+\beta_2 P_2}{\gamma_1+\gamma_2(1-\zeta\eta)}dx\right) \\
&= (\gamma_1+\gamma_2(1-\zeta\eta))\phi\left(\frac{\beta_1+\beta_2(1-\eta)}{\gamma_1+\gamma_2(1-\zeta\eta)}\right) \\
&= (\gamma_1+\gamma_2(1-\zeta\eta))\phi\left(\frac{\beta_1+\beta_2}{\gamma_1+\gamma_2}\right).
\end{aligned}$$

Therefore, we have

$$D_\phi(P\|Q) \geq (\gamma_1+\gamma_2)\phi\left(\frac{\beta_1+\beta_2}{\gamma_1+\gamma_2}\right)+\gamma_3\phi\left(\frac{\beta_3}{\gamma_3}\right)+\left[\gamma_2\phi\left(\frac{\beta_3}{\gamma_3}\right)-\frac{\beta_2(\gamma_1+\gamma_2)}{\beta_1+\beta_2}\phi\left(\frac{\beta_1+\beta_2}{\gamma_1+\gamma_2}\right)\right]\eta.$$

Now, we show the η term is non-negative. We write

$$\begin{aligned}
\gamma_2\phi\left(\frac{\beta_3}{\gamma_3}\right)-\frac{\beta_2(\gamma_1+\gamma_2)}{\beta_1+\beta_2}\phi\left(\frac{\beta_1+\beta_2}{\gamma_1+\gamma_2}\right) &= \beta_2\left(\frac{\gamma_2}{\beta_2}\phi\left(\frac{\beta_3}{\gamma_3}\right)-\frac{(\gamma_1+\gamma_2)}{\beta_1+\beta_2}\phi\left(\frac{\beta_1+\beta_2}{\gamma_1+\gamma_2}\right)\right) \\
&= \beta_2\left(\frac{\gamma_3}{\beta_3}\phi\left(\frac{\beta_3}{\gamma_3}\right)-\frac{1-\gamma_3}{1-\beta_3}\phi\left(\frac{1-\beta_3}{1-\gamma_3}\right)\right).
\end{aligned}$$

It suffices to prove the function $\psi(u)=\phi(u)/u$ satisfies

$$\psi\left(\frac{\beta_3}{\gamma_3}\right)\geq\psi\left(\frac{1-\beta_3}{1-\gamma_3}\right).$$

We use the Mathematica software [Inc.] to compute the difference:

$$\begin{aligned}
&\psi\left(\tfrac{\beta_3}{\gamma_3}\right)-\psi\left(\tfrac{1-\beta_3}{1-\gamma_3}\right)\\
&=-\tfrac{\alpha}{\alpha_-}\log\tfrac{2-\alpha}{1-\alpha_-}+\alpha\alpha_-\log\tfrac{\alpha_-(2-\alpha)}{1-\alpha_-}+\tfrac{\alpha(1-\alpha_-)}{2-\alpha}(\log 4-\alpha\log\alpha)\\
&-\tfrac{\alpha}{2-\alpha}\left(\tfrac{\lambda+1}{\lambda\alpha_-+\alpha_+}-1\right)(\log 4-\alpha\log\alpha)\\
&-\alpha\log\tfrac{(2-\alpha)(\lambda\alpha_-+\alpha_+)}{\lambda(1-\alpha_-)+1-\alpha_+}+\tfrac{\alpha(\lambda+1)}{\lambda\alpha_-+\alpha_+}\log\tfrac{(\lambda+1)(2-\alpha)}{\lambda(1-\alpha_-)+1-\alpha_+}.
\end{aligned}$$

The minimum value of the above difference for $\alpha_-\in[0,\frac{1}{2}]$, $\alpha_+\in[0,\frac{1}{2}]$, and $\lambda\in[0,1]$ is obtained at $\alpha_-=\alpha_+=\frac{1}{2}$, where the difference equals zero. This makes us able to conclude

$$D_\phi(P\|Q)\geq(\gamma_1+\gamma_2)\phi\left(\frac{\beta_1+\beta_2}{\gamma_1+\gamma_2}\right)+\gamma_3\phi\left(\frac{\beta_3}{\gamma_3}\right).$$

Finally, we let $P_2=P_1$. In this case,

$$\begin{aligned}
D_\phi(P\|Q)&=\int_{x\in\mathbb{R}^d}\left(\sum_{i=1}^{3}\gamma_iP_i\right)\phi\left(\frac{\sum_{i=1}^{3}\beta_iP_i}{\sum_{i=1}^{3}\gamma_iP_i}\right)dx\\
&=\int_{x\notin\Omega}(\gamma_1P_1+\gamma_2P_2)\phi\left(\frac{\beta_1P_1+\beta_2P_2}{\gamma_1P_1+\gamma_2P_2}\right)dx\\
&+\int_{x\in\Omega}\gamma_3P_3\phi\left(\frac{\beta_3P_3}{\gamma_3P_3}\right)dx\\
&=(\gamma_1+\gamma_2)\phi\left(\frac{\beta_1+\beta_2}{\gamma_1+\gamma_2}\right)+\gamma_3\phi\left(\frac{\beta_3}{\gamma_3}\right).
\end{aligned}$$

Therefore, the optimal generator is $p_G=p_{\text{data}}|_{\bar{\Omega}}$.

Extension to f-GAN.

We can extend the objective (4.2) to any type of f-GAN. Let ϕ be a convex, lower-semicontinuous function such that $\phi(1)=0$. Let

$$P=\frac{\alpha_+}{\alpha}p_{\text{data}}|_{\bar{\Omega}}+\frac{\alpha_-}{\alpha}p_{\text{fake}};\;\; Q=\frac{1-\alpha_+}{2-\alpha}p_{\text{data}}|_{\bar{\Omega}}+\frac{1-\alpha_-}{2-\alpha}p_{\text{fake}}.$$

We jointly optimize

$$\min_G \max_D \; L(G,D) = \mathbb{E}_{x\sim P} D(x) - \mathbb{E}_{x\sim Q} \phi^*(D(x)).$$

Then, the optimal discriminator is $D = \phi'\left(\frac{P}{Q}\right)$. If $\psi\left(\frac{\beta_3}{\gamma_3}\right) \geq \psi\left(\frac{1-\beta_3}{1-\gamma_3}\right)$, then the optimal generator is $p_G = p_{\text{data}}|_{\bar{\Omega}}$.

Remark D.1.1. *When $\alpha_- = 0$ and $\alpha_+ = 1$ (i.e. there is no label smoothing), Theorem 1 in Sinha et al. [2020] implies the above optimal generator. Our theorem also extends their theorem to the label smoothing setting.*

D.2 Theoretical Analysis of a Simplified Dynamical System on Invalidity

In this section, we provide theoretical analysis to a simplified, ideal dynamical system that corresponds to Alg. 2 and Section 4.3.2. In this dynamical system, we assume there are only two types of invalid samples: those easy to redact, and those hard to redact. We assume after each iteration, the generator will generate a less but positive fraction of invalid samples. Formally, let $\{\Omega_{\text{easy}}, \Omega_{\text{hard}}\}$ be a split of Ω, where Ω_{easy} is the set of invalid samples that are easy to redact, and Ω_{hard} is the set of invalid samples that are hard to redact. We let

$$\begin{aligned} m_{\text{easy}} &= \int_{\Omega_{\text{easy}}} p_G(x)dx, \\ m_{\text{hard}} &= \int_{\Omega_{\text{hard}}} p_G(x)dx, \\ m_{\text{ratio}} &= \frac{m_{\text{easy}}}{m_{\text{easy}} + m_{\text{hard}}}. \end{aligned}$$

Then, m_{easy} is the fraction of invalid generated samples that are easy to redact, and m_{hard} is the fraction of invalid generated samples that are hard to redact. $m_{\text{easy}} + m_{\text{hard}}$ is the fraction of invalid generated samples over all generated ones, which we call **invalidity**. We use superscript to represent each iteration. We consider the following dynamical system:

$$\begin{aligned} m_{\text{easy}}^{i+1} &= m_{\text{easy}}^{i} \cdot \eta_{\text{easy}}(m_{\text{ratio}}^{i}, T), \\ m_{\text{hard}}^{i+1} &= m_{\text{hard}}^{i} \cdot \eta_{\text{hard}}(m_{\text{ratio}}^{i}, T). \end{aligned}$$

In other words, the improvement of m_{easy} and m_{hard} (in terms of multiplication factor) is only affected by m_{ratio} and T. We make this assumption because in practice, the number of invalid samples to optimize the loss function is always fixed. As for boundary conditions, we assume $m_{\text{easy}}^0 > m_{\text{hard}}^0$. We assume for $\eta \in \{\eta_{\text{easy}}, \eta_{\text{hard}}\}$, $0 < \eta(m,T) \leq 1$, where equality holds only in these situations:

$$\eta(m,0) = 1,\ \eta_{\text{easy}}(0,T) = 1,\ \eta_{\text{hard}}(1,T) = 1.$$

We also assume a larger T leads to smaller η, but this effect degrades as T increases:

$$\frac{\partial}{\partial T}\eta(m,T) < 0,\ \frac{\partial^2}{\partial T^2}\eta(m,T) > 0.$$

To distinguish between samples that are easy or hard to redact, we assume

$$\frac{1}{m}\cdot\frac{\partial}{\partial T}\eta_{\text{easy}}(m,T) < \frac{1}{1-m}\cdot\frac{\partial}{\partial T}\eta_{\text{hard}}(m,T) < 0.$$

We can now draw some conclusions below.

As $i \to \infty$, invalidity converges to 0.

Because $\eta_{\text{easy}}(T) < 1$ and $\eta_{\text{hard}}(T) < 1$ when $T > 0$, we have $m_{\text{easy}}^{i+1} \le m_{\text{easy}}^{i}$ and $m_{\text{hard}}^{i+1} \le m_{\text{hard}}^{i}$. According to the monotone convergence theorem, there exists $m_{\text{easy}}^{\infty} \ge 0$ and $m_{\text{hard}}^{\infty} \ge 0$ such that

$$\lim_{i\to\infty} m_{\text{easy}}^{i} = m_{\text{easy}}^{\infty},\ \lim_{i\to\infty} m_{\text{hard}}^{i} = m_{\text{hard}}^{\infty}.$$

We now prove $m_{\text{easy}}^{\infty} = m_{\text{hard}}^{\infty} = 0$. If otherwise, there exists $m_{\text{ratio}}^{\infty} = \frac{m_{\text{easy}}^{\infty}}{m_{\text{easy}}^{\infty}+m_{\text{hard}}^{\infty}}$ such that $m_{\text{ratio}}^{i} \to m_{\text{ratio}}^{\infty}$. We then have

$$m_{\text{easy}}^{\infty} = m_{\text{easy}}^{\infty}\cdot\eta_{\text{easy}}(m_{\text{ratio}}^{\infty},T),$$
$$m_{\text{hard}}^{\infty} = m_{\text{hard}}^{\infty}\cdot\eta_{\text{hard}}(m_{\text{ratio}}^{\infty},T).$$

If $m_{\text{easy}}^{\infty} > 0$, then $m_{\text{ratio}}^{\infty} > 0$, and $\eta_{\text{easy}}(m_{\text{ratio}}^{\infty},T) < 1$, contradiction. Similarly, if $m_{\text{hard}}^{\infty} > 0$, then $m_{\text{ratio}}^{\infty} < 1$, and $\eta_{\text{hard}}(m_{\text{ratio}}^{\infty},T) < 1$, contradiction. Therefore, we conclude both m_{easy}^{i} and m_{hard}^{i} converge to 0. This indicates the invalidity converges to zero.

Simplifying the dynamical system.

To further simplify the problem, we make a strong assumption that η is linear in m. Then, we must have

$$\begin{aligned}
\eta_{\text{easy}}(m,T) &= 1-\xi_{\text{easy}}(T)\cdot m,\\
\eta_{\text{hard}}(m,T) &= 1-\xi_{\text{hard}}(T)\cdot(1-m),
\end{aligned}$$

where $\xi \in [0,1], \xi(0)=0, \xi' > 0, \xi'' < 0$ for $\xi \in \{\xi_{\text{easy}}, \xi_{\text{hard}}\}$. We also have $\xi'_{\text{easy}} > \xi'_{\text{hard}}$ and therefore $\xi_{\text{easy}} > \xi_{\text{hard}}$.

Optimal T and R from bounds.

We have

$$m^{i+1}_{\text{easy}} + m^{i+1}_{\text{hard}} = m^{i}_{\text{easy}} + m^{i}_{\text{hard}} - \frac{\xi_{\text{easy}}(T)(m^{i}_{\text{easy}})^2 + \xi_{\text{hard}}(T)(m^{i}_{\text{hard}})^2}{m^{i}_{\text{easy}} + m^{i}_{\text{hard}}}.$$

Because $\xi_{\text{easy}}(T) \geq \xi_{\text{hard}}(T)$, we have

$$\begin{aligned}
\frac{\xi_{\text{easy}}(T)\xi_{\text{hard}}(T)}{\xi_{\text{easy}}(T)+\xi_{\text{hard}}(T)}(m^{i}_{\text{easy}} + m^{i}_{\text{hard}}) &\leq \frac{\xi_{\text{easy}}(T)(m^{i}_{\text{easy}})^2 + \xi_{\text{hard}}(T)(m^{i}_{\text{hard}})^2}{m^{i}_{\text{easy}} + m^{i}_{\text{hard}}}\\
&\leq \xi_{\text{easy}}(T)(m^{i}_{\text{easy}} + m^{i}_{\text{hard}}).
\end{aligned}$$

This leads to

$$1-\xi_{\text{easy}}(T) \leq \frac{m^{i+1}_{\text{easy}} + m^{i+1}_{\text{hard}}}{m^{i}_{\text{easy}} + m^{i}_{\text{hard}}} \leq 1 - \frac{\xi_{\text{easy}}(T)\xi_{\text{hard}}(T)}{\xi_{\text{easy}}(T)+\xi_{\text{hard}}(T)},$$

and therefore

$$\left(1-\xi_{\text{easy}}(T)\right)^R \leq \frac{m^{R}_{\text{easy}} + m^{R}_{\text{hard}}}{m^{0}_{\text{easy}} + m^{0}_{\text{hard}}} \leq \left(1 - \frac{\xi_{\text{easy}}(T)\xi_{\text{hard}}(T)}{\xi_{\text{easy}}(T)+\xi_{\text{hard}}(T)}\right)^R.$$

Assume the number of queries, $T \times R$, is fixed. Then, the optimal T from the lower bound is

$$T^*_{\text{low}} = \arg\min_T \frac{1}{T} \log(1 - \xi_{\text{easy}}(T)).$$

By setting the derivative to be zero, we have T^*_{low} is the solution to

$$-T\xi'_{\text{easy}}(T) = (1 - \xi_{\text{easy}}(T)) \log(1 - \xi_{\text{easy}}(T)).$$

Similarly, the optimal T from the upper bound is

$$T^*_{\text{upp}} = \arg\min_T \frac{1}{T} \log\left(1 - \frac{\xi_{\text{easy}}(T)\xi_{\text{hard}}(T)}{\xi_{\text{easy}}(T) + \xi_{\text{hard}}(T)}\right).$$

By setting the derivative to be zero, we have T^*_{upp} is the solution to

$$\begin{aligned}&-T \cdot \frac{\xi'_{\text{easy}}(T)\xi_{\text{hard}}(T)^2 + \xi'_{\text{hard}}(T)\xi_{\text{easy}}(T)^2}{(\xi_{\text{easy}}(T) + \xi_{\text{hard}}(T))^2} \\ &= \left(1 - \frac{\xi_{\text{easy}}(T)\xi_{\text{hard}}(T)}{\xi_{\text{easy}}(T) + \xi_{\text{hard}}(T)}\right) \log\left(1 - \frac{\xi_{\text{easy}}(T)\xi_{\text{hard}}(T)}{\xi_{\text{easy}}(T) + \xi_{\text{hard}}(T)}\right).\end{aligned}$$

D.3 Feasibility of Discriminator in the Classifier-based Setting

The solution to (4.4) and (4.5) is:

$$D^*(x) = \begin{cases} \frac{\alpha_+ p_{\text{data}}|_{\bar{\Omega}} + \alpha_-(\lambda p_G + (1-\lambda)p_\Omega)}{p_{\text{data}}|_{\bar{\Omega}} + \lambda p_G + (1-\lambda)p_\Omega} & \text{if } \mathbf{f}(x) \geq \tau \\ \alpha_- & \text{if } \mathbf{f}(x) < \tau \end{cases},$$

which satisfies $D^* \in [0,1]$. Therefore, (4.4) is feasible with the `guide` function defined in (4.5).

D.4 Experimental Setup

Pre-training. We use DCGAN [Radford et al., 2015] with latent dimension = 128 as the model. The pre-trained model is trained with label smoothing ($\alpha_+ = 0.9, \alpha_- = 0.1$):

$$\begin{aligned}\min_G \max_D \quad & \mathbb{E}_{x\sim X}\left[\alpha_+ \log D(x) + (1-\alpha_+)\log(1-D(x))\right] \\ & + \mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[\alpha_- \log D(G(z)) + (1-\alpha_-)\log(1-D(G(z)))\right].\end{aligned}$$

We use Adam optimizer with learning rate $= 2\times 10^{-4}, \beta_1 = 0.5, \beta_2 = 0.999$ to optimize both the generator and the discriminator. The networks are trained for 200 epochs with a batch size of 64. For each iteration over one mini-batch, we let K_D be the number of times to update the discriminator, and K_G the number of times to update the generator. We use $K_D = 1$ and $K_G = 5$ to train.

Data redaction. The setup is similar to the pre-training except for two differences. The number of epochs is much smaller: 8 for MNIST, 30 for CIFAR, and 40 for STL-10. We let $K_G = 1$ for MNIST and CIFAR and $K_G = 5$ for STL-10.

Evaluation. To measure invalidity, we generate 50K samples, and compute the fraction of these samples that are not valid (e.g., classified as the label to be redacted, or with pre-defined biases). It is the lower the better. The invalidity for redacting labels is measured based on label classifiers. We use pre-trained classifiers on these datasets. [1]

The other evaluation metric is generation quality. The inception score (`IS`) [Salimans et al., 2016] is computed based on logit distributions from the above pre-trained classifiers. It is the higher the better. The Frechet Inception Distance (`FID`) [Heusel et al., 2017] is computed based on an open-sourced PyTorch implementation. [2] It is the lower the better.

When computing these quality metrics, we generate 50K samples, and compare to the set of valid training samples: $\{x \in X : x \notin \Omega\}$. Therefore, when $X \cap \Omega$ is not the empty set (such

[1]https://github.com/aaron-xichen/pytorch-playground (MIT license)

[2]https://github.com/mseitzer/pytorch-fid (Apache-2.0 license)

as redacting labels in Section 4.4.1), the quality measure of the model after data redaction is not directly comparable to the pre-trained model, but these scores among different redaction algorithms are comparable and give intuition to the generation quality. When $X \cap \Omega$ is the empty set (such as de-biasing in Section 4.4.2), the quality measures of the pre-trained model and the model after data redaction are directly comparable.

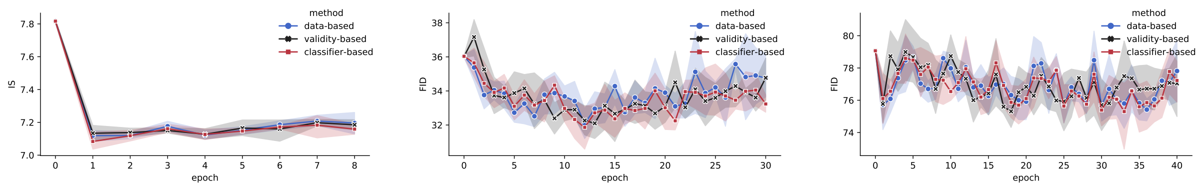
D.5 Redacting Labels

D.5.1 Redacting Label 0

We include additional results for redacting label 0 in this section. We discuss quality during redaction, results after one epoch, the effect of λ.

Quality during data redaction

We plot quality measure of different data redaction algorithms on different datasets during the redaction process, complementary to the invalidity in Fig. 4.2. We find the variances of quality measure is higher than the invalidity, but different redaction algorithms are generally comparable.



(a) MNIST **(b)** CIFAR-10 **(c)** STL-10

Figure D.1. Quality measure during data redaction. Mean and standard errors are plotted for five random seeds.

Invalidity after one epoch

We compare invalidity after only one epoch of data redaction. These redaction algorithms are highly comparable to each other. We hypothesis that the classifier-based algorithm performs the best on MNIST because a label classifier on MNIST (and its gradient information) can be very accurate, while this may not be true for CIFAR-10 and STL-10.

Table D.1. Invalidity after one epoch of data redaction.

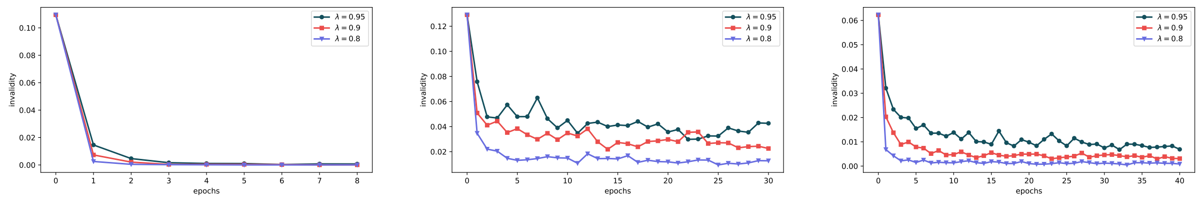
Dataset	Scale	Pre-trained	Data-based	Validity-based	Classifier-based
MNIST	$\times 10^{-3}$	1.1×10^{2}	4.7 ± 0.8	5.6 ± 0.9	$\mathbf{3.9 \pm 0.9}$
CIFAR-10	$\times 10^{-2}$	1.3×10^{1}	$\mathbf{3.7 \pm 0.5}$	3.7 ± 0.8	3.8 ± 0.3
STL-10	$\times 10^{-3}$	6.2×10^{1}	9.1 ± 0.9	$\mathbf{8.6 \pm 0.9}$	10.6 ± 1.2

Trade-off by alternating λ

We study the effect of λ (hyper-parameter in (4.1)) in Table D.2 and Fig. D.2. There is a trade-off by alternating λ: a larger λ (less fake data from the redaction set) leads to better quality measure, and a smaller λ (more fake data from the redaction set) leads to better invalidity.

Table D.2. Invalidity after data redaction for different λ in the classifier-based redaction algorithm.

λ	MNIST		CIFAR-10		STL-10	
	`Inv`($\downarrow$)	`IS`($\uparrow$)	`Inv`($\downarrow$)	`FID`($\downarrow$)	`Inv`($\downarrow$)	`FID`($\downarrow$)
0.8	0.6×10^{-4}	7.15	1.28×10^{-2}	33.7	0.86×10^{-3}	75.4
0.9	0.8×10^{-4}	7.18	2.25×10^{-2}	28.6	3.10×10^{-3}	77.2
0.95	7.2×10^{-4}	7.24	4.26×10^{-2}	26.9	6.89×10^{-3}	76.2



(a) MNIST **(b)** CIFAR-10 **(c)** STL-10

Figure D.2. Invalidity during data redaction for different λ in the classifier-based redaction algorithm.

D.5.2 Redacting Other Labels

We also demonstrate results for redacting other labels with our data redaction algorithms. We use the base set of hyper-parameters in Appendix D.5.1. Similar to redacting label 0, all redaction algorithms can largely reduce invalidity, and they are highly comparable to each other. The classifier-based redaction algorithm achieves slightly better generation quality on MNIST and CIFAR-10. In terms of different labels, we find some labels are harder to redact in the sense that the invalidity scores for these labels are higher than other scores, such as label 9 in MNIST, and label 3 in CIFAR-10 and STL-10.

Table D.3. Redacting other labels on MNIST.

Label	Pre-trained		Data-based		Validity-based		Classifier-based	
	`Inv`(↓)	`IS`(↑)	`Inv`(↓)	`IS`(↑)	`Inv`(↓)	`IS`(↑)	`Inv`(↓)	`IS`(↑)
1	10.2%	7.81	0.002%	7.01	**0.000**%	**7.21**	0.008%	7.13
2	8.6%	7.81	0.022%	7.22	**0.012**%	7.20	0.028%	**7.28**
3	11.5%	7.81	**0.126**%	7.20	0.136%	**7.24**	0.134%	7.19
4	9.9%	7.81	0.138%	7.19	**0.092**%	7.21	0.104%	**7.26**
5	8.7%	7.81	0.048%	7.22	**0.046**%	7.21	0.056%	**7.24**
6	9.0%	7.81	0.020%	7.04	0.022%	7.07	**0.010**%	**7.12**
7	11.4%	7.81	0.114%	7.24	0.124%	**7.34**	**0.088**%	7.32
8	9.1%	7.81	**0.198**%	7.48	0.248%	7.35	0.302%	**7.51**
9	10.7%	7.81	0.486%	7.30	**0.414**%	**7.36**	0.545%	7.26

Table D.4. Redacting other labels on CIFAR-10.

Label	Pre-trained		Data-based		Validity-based		Classifier-based	
	`Inv`(↓)	`FID`(↓)	`Inv`(↓)	`FID`(↓)	`Inv`(↓)	`FID`(↓)	`Inv`(↓)	`FID`(↓)
1	1.5%	36.24	0.032%	35.06	**0.014**%	35.23	0.082%	**33.40**
2	11.0%	36.24	**1.311**%	31.67	1.537%	31.65	1.564%	**28.34**
3	15.8%	36.24	3.013%	30.10	3.491%	31.01	**2.534**%	**28.06**
4	16.8%	36.24	1.752%	30.36	1.754%	31.26	**1.590**%	**29.72**
5	6.7%	36.24	**0.799**%	**30.76**	0.985%	30.90	1.461%	31.36
6	9.3%	36.24	0.797%	29.81	1.071%	31.65	**0.755**%	**29.64**
7	8.6%	36.24	0.789%	33.48	**0.496**%	**33.40**	1.325%	34.15
8	10.3%	36.24	**0.218**%	38.96	1.451%	38.59	0.496%	**34.56**
9	7.1%	36.24	**0.138**%	38.13	0.186%	37.74	0.216%	**36.85**

Table D.5. Redacting other labels on STL-10.

Label	Pre-trained		Data-based		Validity-based		Classifier-based	
	`Inv`($\downarrow$)	`FID`($\downarrow$)	`Inv`($\downarrow$)	`FID`($\downarrow$)	`Inv`($\downarrow$)	`FID`($\downarrow$)	`Inv`($\downarrow$)	`FID`($\downarrow$)
1	9.0%	79.00	**1.273%**	74.89	2.168%	**73.91**	1.900%	75.34
2	6.2%	79.00	0.158%	**72.22**	**0.132%**	72.39	0.176%	75.75
3	14.9%	79.00	3.772%	77.24	**3.732%**	76.80	4.412%	**75.19**
4	8.2%	79.00	1.634%	**81.91**	**1.345%**	82.82	1.425%	83.25
5	15.1%	79.00	**2.072%**	**76.85**	3.383%	80.40	5.041%	77.74
6	8.7%	79.00	**0.462%**	80.82	0.518%	**78.17**	0.745%	79.63
7	10.7%	79.00	2.973%	**77.53**	**1.838%**	78.57	2.180%	77.58
8	9.5%	79.00	0.304%	79.56	**0.272%**	78.06	0.352%	**77.07**
9	11.6%	79.00	**0.817%**	76.70	0.947%	78.37	0.941%	**76.37**

D.5.3 Detailed Setup of Redacting Multiple Sets

We use 30K images from CelebA-64 as the training set. All other hyper-parameters are the same as the base set for STL-10 in Appendix D.5.1, except that we run data redaction algorithms for only 5 epochs. We train attribute classifiers for each attribute separately. The attribute classifiers are fine-tuned from open-sourced pre-trained ResNet [He et al., 2016a]. [3] We fine-tune the network for 20 epochs using the SGD optimizer with learning rate $= 1 \times 10^{-3}$, momentum $= 0.9$, and a batch size of 64.

[3]https://pytorch.org/vision/stable/models.html

D.6 Model De-biasing

D.6.1 Boundary Artifacts

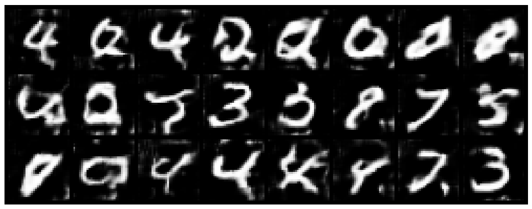
Let the image size be $W \times H$ (the number of channels is 1 for MNIST). For an integer margin, the boundary pixels are defined as

$$\{(i, j) : 1 \leq i \leq \text{margin or } W - \text{margin} < i \leq W, 1 \leq j \leq \text{margin or } H - \text{margin} < j \leq H\}.$$

Then, the validity function for boundary artifacts is defined as

$$\mathbf{v}(x) = 1\left\{\sum_{(i,j)\in\text{boundary pixels}} x_{ij} < \tau_b\right\},$$

where $\tau_b = 4.25$ for margin $= 1$ and 10.0 for margin $= 2$. For these values, no training data has the boundary artifact. Quantitative results are in Tabel 4.5. We visualize some samples with boundary artifacts in Fig. D.3a. We run the validity-based redaction algorithm with $\lambda = 0.98, \alpha_+ = 0.95, \alpha_- = 0.05$ for 4 epochs. After de-biasing via data redaction, these samples have less boundary pixels, as shown in Fig. D.3b.


(a) Samples with boundary artifacts.

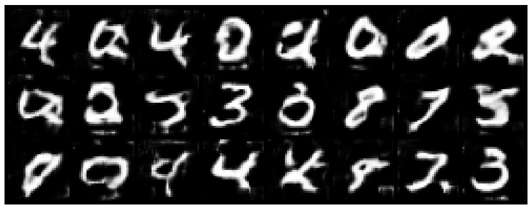
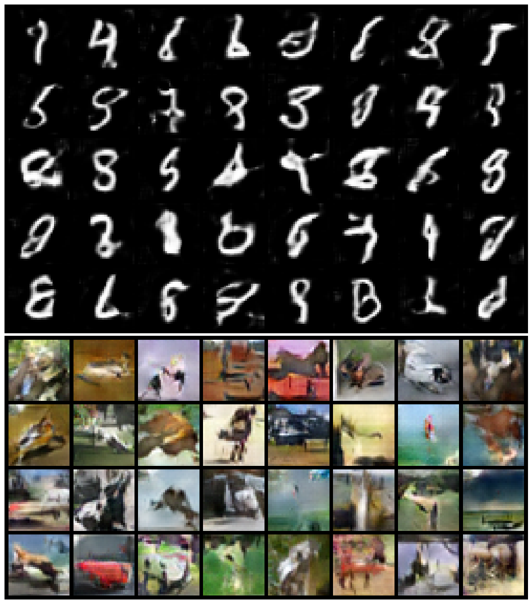

(b) Samples after de-biasing via data redaction.

Figure D.3. De-biasing boundary artifacts with the validity-based data redaction algorithm. Margin $= 1$ and $T = 40$K.

D.6.2 Label Biases

We use classifier-based redaction algorithm to de-bias label biases. For MNIST, we use $\lambda = 0.8, \alpha_+ = 0.95, \alpha_- = 0.05$ and run for 8 epochs. For CIFAR-10, we use $\lambda = 0.9, \alpha_+ =$

$0.9, \alpha_{-} = 0.05$ and run for 30 epochs. Quantitative results are in Table 4.6 and 4.7. We visualize semantically ambiguous samples generated by the pre-trained model in Fig. D.4a. After de-biasing via data redaction, these samples become less semantically ambiguous, as shown in Fig. D.4b.


(a) Samples with label-biases.

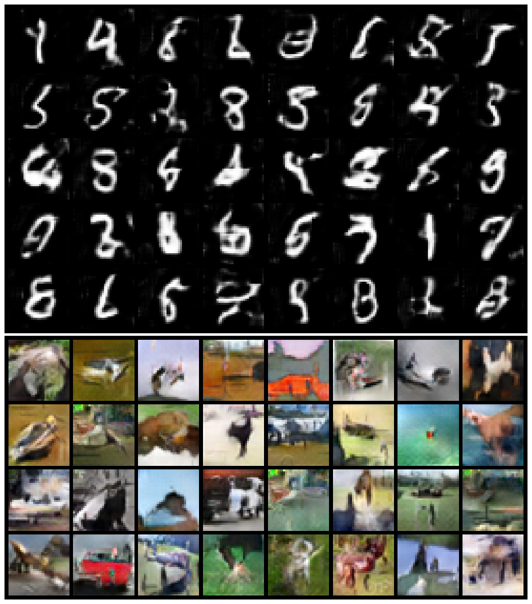
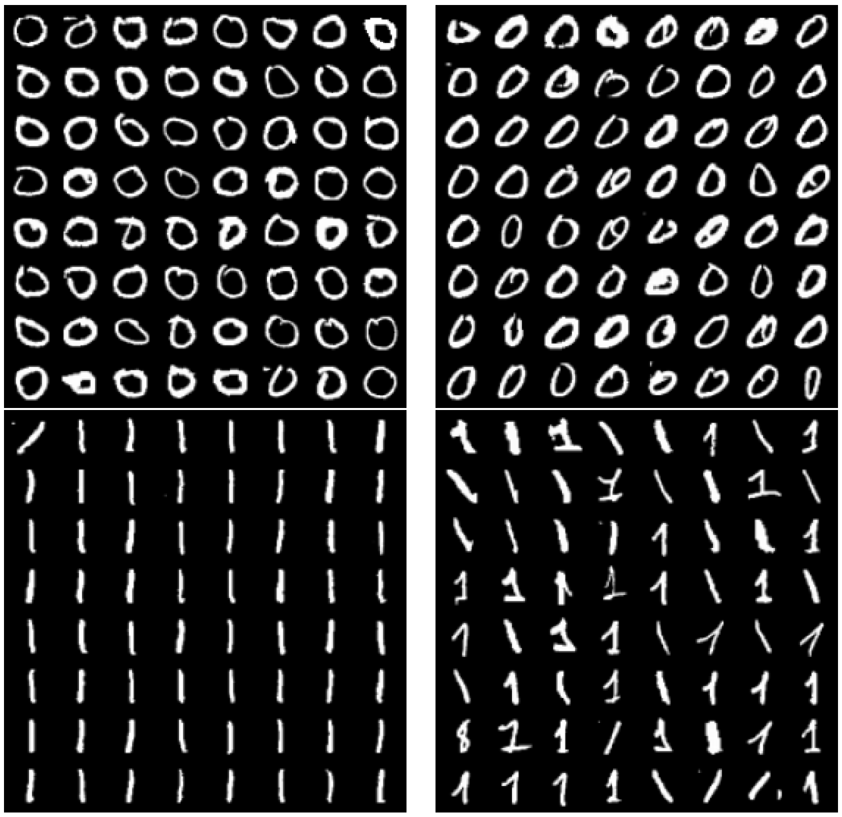

(b) Samples after de-biasing via data redaction.

Figure D.4. De-biasing label biases with the classifier-based data redaction algorithm ($\tau = 0.7$).

D.7 Understanding Training Data

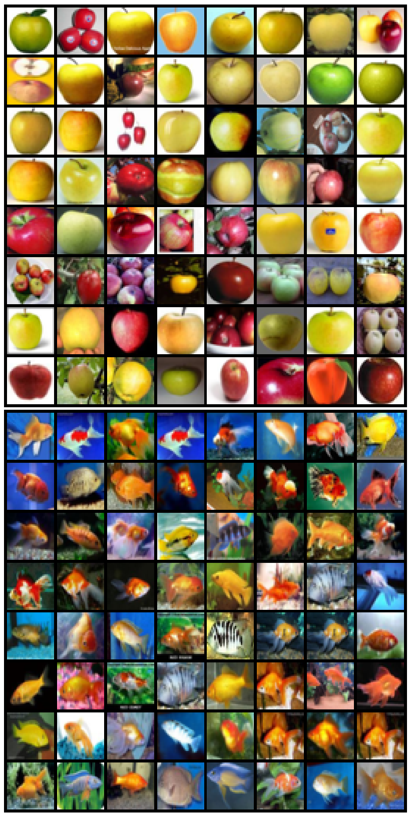
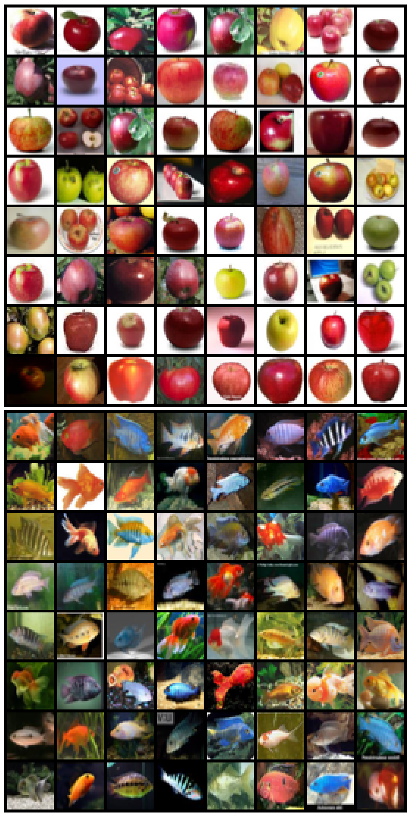
D.7.1 Sample-level redaction difficulty

We visualize some most and least difficult-to-redact samples according to the redaction scores in Fig. D.5 and Fig. D.6. We find the most difficult-to-redact samples are visually atypical, while the least difficult-to-redact samples are visually more common.



(a) Samples that are easiest to redact. **(b)** Samples that are hardest to redact.

Figure D.5. Samples that are most and least difficult-to-redact in MNIST.

(a) Samples that are easiest to redact. **(b)** Samples that are hardest to redact.

Figure D.6. Samples that are most and least difficult-to-redact in CIFAR-100.

D.7.2 Label-level redaction difficulty

We sort all labels according to their average redaction scores. This tells us which labels are easier or harder to redact. The results for MNIST are in Fig. D.7. Consistent with Table D.3, label 9 is the most difficult label to redact. The most and least difficult-to-redact labels for CIFAR-100 are shown in Fig. D.8a and D.8b.

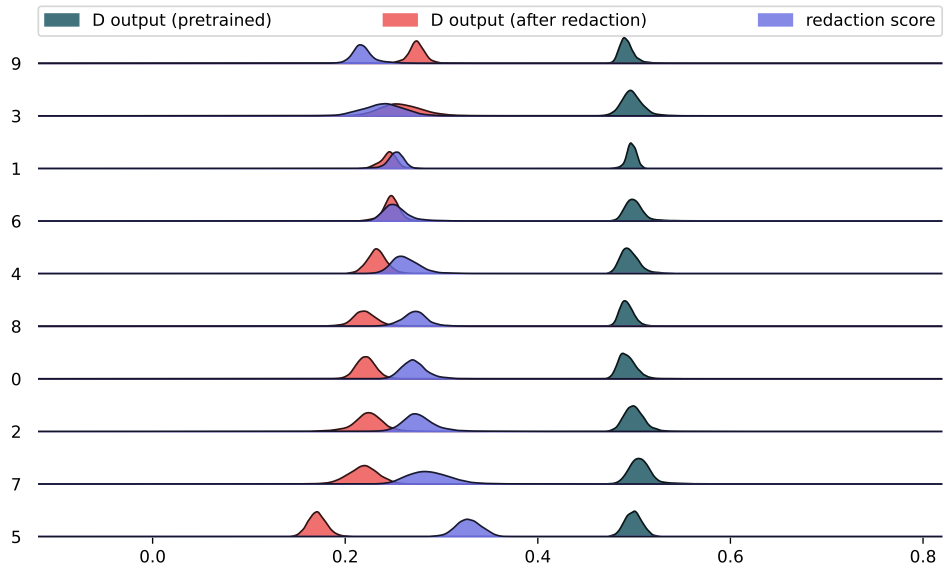
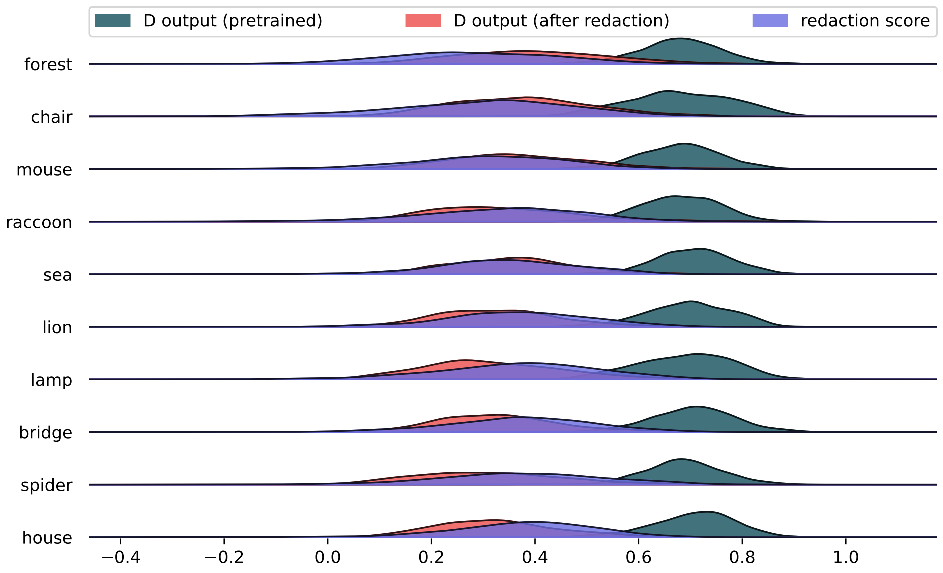
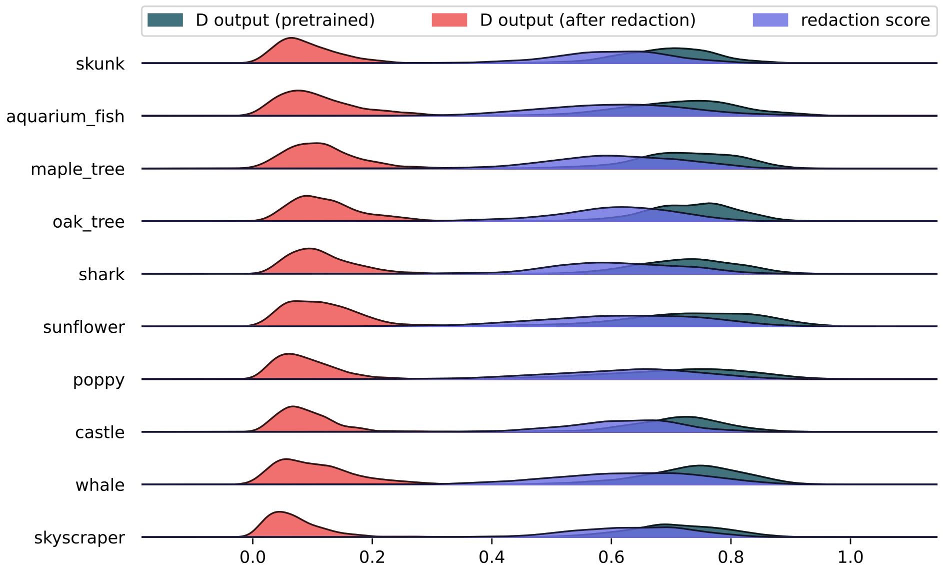


Figure D.7. Label-level redaction difficulty for MNIST. Top: the most difficult to redact. Bottom: the least difficult to redact. A large redaction score means a label is easier to be redacted. We find some labels are more difficult to redact than others.



(a) Label-level redaction difficulty for CIFAR-100 (10 most difficult-to-redact labels). Top: the most difficult to redact. Bottom: the least difficult to redact.



(b) Label-level redaction difficulty for CIFAR-100 (10 least difficult-to-redact labels). Top: the most difficult to redact. Bottom: the least difficult to redact.

Figure D.8. Label-level redaction difficulty for CIFAR-100. A large redaction score means a label is easier to be redacted. We find some labels are more difficult to redact than others.

Appendix E

Data Redaction from Conditional Generative Models

E.1 Redacting Models Conditioned on Discrete Labels

Redacting multiple labels. Suppose there are multiple labels $\{1,\cdots,J\}$ ($J<k$) to be redacted. The M' matrix needs to satisfy $M'v_i = Mv_i$ for $i>J$, and $M'v_j = MV_{-J}\eta_{-j}$ for $j\leq J$, where $V_{-J}=[v_{J+1},\cdots,v_k]\in\mathbb{R}^{k\times(k-J)}$. For $j\leq J$, let u_j be the basis vector of the null space of $\{v_i\}_{i\neq j}$. Each row of $M'-M$ is in the null space of $\{v_i\}_{i>J}$, which can be written as a linear combination of $\{u_j\}_{j=1}^J$. Therefore, we can represent $M'-M$ as

$$M'-M=\sum_{j=1}^{J}\omega_j u_j^\top = WU^\top,$$

where the j-th column of W (U) is ω_j (u_j). Let $V_J=[v_1,\cdots,v_J]$ and $Y_{-J}=[\eta_{-1},\cdots,\eta_{-J}]$. We have $M'V_J=MV_{-J}Y_{-J}$. This simplifies to

$$WU^\top V_J = M(V_{-J}Y_{-J}-V_J).$$

Notice that $U^\top V_J$ is a diagonal matrix with j-th diagonal element $u_j^\top v_j\neq 0$. Therefore, we have

$$W = M(V_{-J}Y_{-J}-V_J)(U^\top V_J)^{-1}. \tag{E.1}$$

Simplified formula when embedding vectors are one-hot. Let $v_i = \mathbf{e}_i$ for each i. Then, we have $u_i = v_i = \mathbf{e}_i$, and therefore $U^\top V_J = I$. We also have $U = V_J = [I_J|\mathbf{0}]^\top$ and $V_{-J} = [\mathbf{0}|I_{k-J}]$, where I_J is the J-dimensional identity matrix. Then,

$$WU^\top = M([\mathbf{0}|I_{k-J}]Y_{-J} - [I_J|\mathbf{0}]^\top)[I_J|\mathbf{0}] = M \begin{pmatrix} -I_J & \mathbf{0} \\ Y_{-J} & \mathbf{0} \end{pmatrix}.$$

As a result,

$$M' = M + WU^\top = M \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ Y_{-J} & I_{k-J} \end{pmatrix}.$$

Higher embedding dimension. Because of linear independence, the null space of $\{v_i\}_{i\neq j}$ has 1 dimension higher than the null space of $\{v_i\}_{i=1}^k$. Therefore, we can pick $u_j \in \mathbf{null}(\{v_i\}_{i\neq j}) \setminus \mathbf{null}(\{v_i\}_{i=1}^k)$.

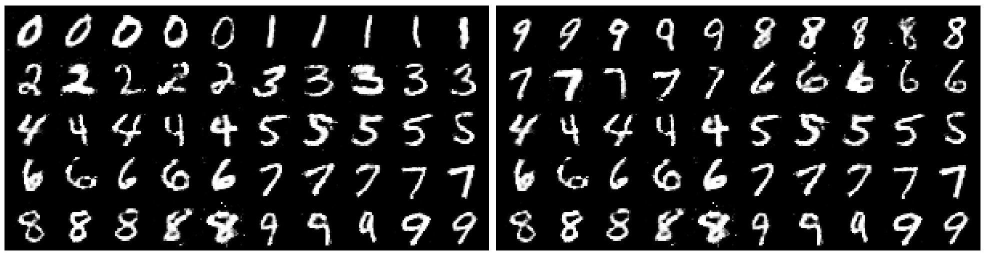


Figure E.1. Redacting labels `0,1,2,3` in cGAN on MNIST. Left: samples generated from the pre-trained model. Right: samples generated from the redacted model. Redacted conditionals (first two rows) are edited as expected, and other conditionals (last three rows) remain unchanged.

E.2 Additional Details and Experiments for Redaction from DM-GAN

E.2.1 Details of the Pre-trained Model

The high-level architecture of DM-GAN is shown in Fig. E.2 and E.3. The first conditioning network H_1 takes the sentence embedding $v_s(c)$ as input and outputs two vectors: a mean vector, and the square root of the variance vector. A re-parameterization similar to variational auto-encoders is applied to these two vectors, and the output is concatenated to the latent code. The other two conditioning networks H_2 and H_3, called the memory writing module, take two inputs: the word embeddings $v_w(c)$, and the image features of the previously generated low resolution images. The output of H_2 or H_3 then goes through the rest of the modules in the main generative network. We use the pre-trained model and code from https://github.com/MinfengZhu/DM-GAN under MIT license. The pre-trained model takes days to train on 1 or more GPUs.

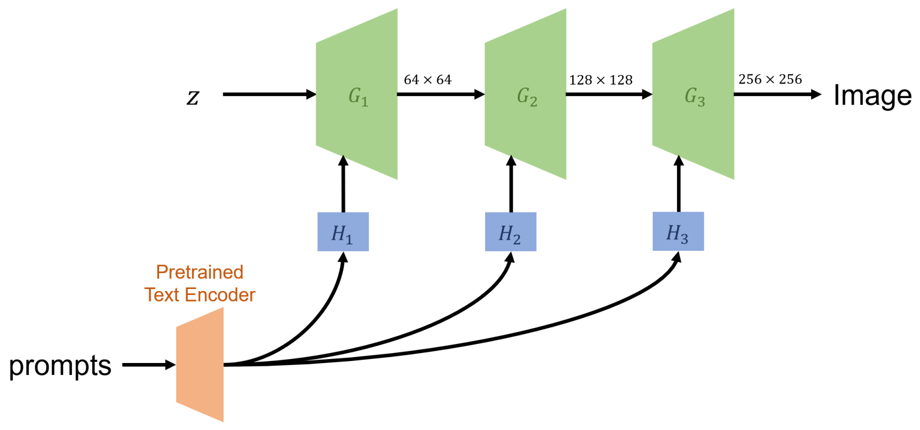


Figure E.2. High-level architecture of DM-GAN.

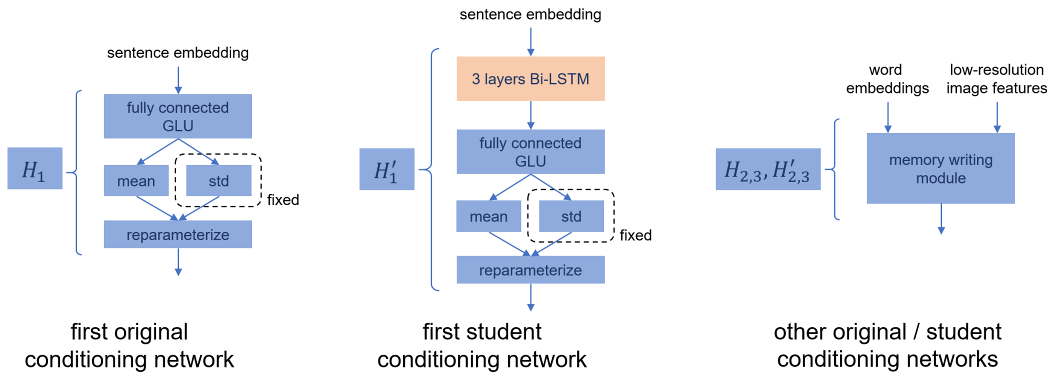


Figure E.3. High-level architecture of original and higher-capacity conditioning networks of DM-GAN.

E.2.2 Redaction Setup

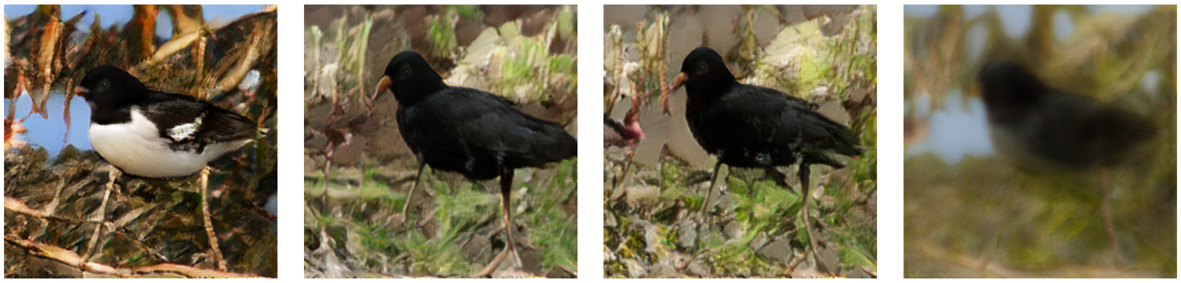
We use one NVIDIA 3080 GPU to train networks. For each H_i, $i = 1,2,3$, we use the Adam optimizer [Kingma and Ba, 2014] with a learning rate 0.005 to optimize the mean square error loss. The redaction algorithm terminates at 1000 iterations. For H_1 we use a batch size of 128, and for H_2 and H_3 we reduce the batch size to 32 in order to fit into GPU memory. The architecture of student conditioning networks with improved capacity is shown in Fig. E.3. Table E.1 includes the number of training and test prompts that are redacted in each experiment. Note that when we redact `blue wings` and `red wings`, we also redact phrases `wings that are blue` and `wings that are red`.

Table E.1. Number of redacted training and test prompts. There are 88550 training prompts and 29330 test prompts in total.

Redaction prompts	# redacted training prompts	# redacted test prompts
`long beak, white belly`	10377	3369
`blue / red wings`	732	303
`blue`	6113	2175
`yellow, red`	29514	9319

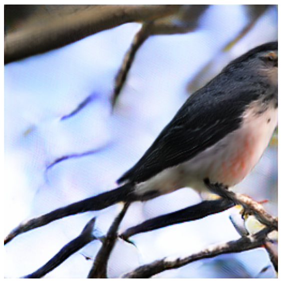
E.2.3 Visualization

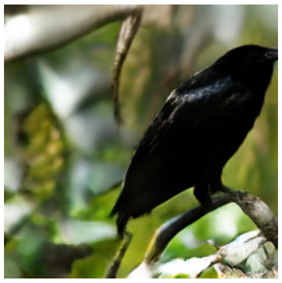
In Fig. E.4 - Fig. E.7 , we visualize examples where we redact prompts that contain `long beak` or `white belly`. In Fig. E.8 - Fig. E.11 , we visualize examples where we redact prompts that contain `blue wings` or `red wings`. In Fig. E.12 - Fig. E.15 , we visualize examples where we redact prompts that contain `blue`. In Fig. E.16 - Fig. E.19 , we visualize examples where we redact prompts that contain `yellow` or `red`.

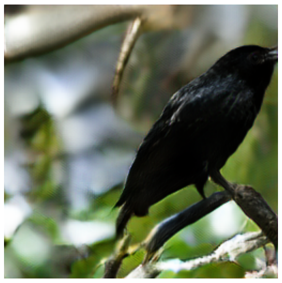


(a) Pre-trained $G(\cdot|c)$ **(b)** Reference $G(\cdot|\hat{c})$ **(c)** Redaction $G'(\cdot|c)$ **(d)** Rewriting Baseline

Figure E.4. Redacted prompt: "this particular bird has a white belly and breasts and black head and back". Reference prompt: "this particular bird has a black belly and breasts and black head and back".


(a) Pre-trained $G(\cdot|c)$


(b) Reference $G(\cdot|\hat{c})$


(c) Redaction $G'(\cdot|c)$

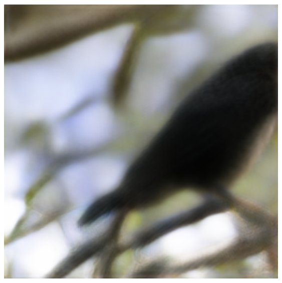
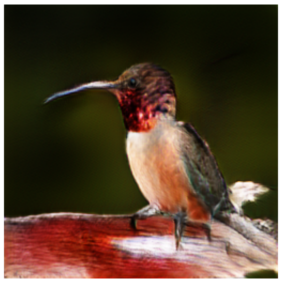
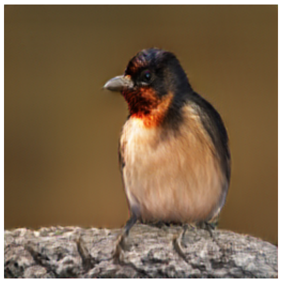
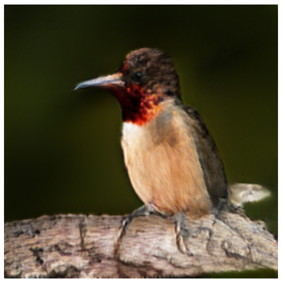

(d) Rewriting Baseline

Figure E.5. Redacted prompt: "this bird has feathers that are black and has a white belly". Reference prompt: "this bird has feathers that are black and has a black belly".


(a) Pre-trained $G(\cdot|c)$


(b) Reference $G(\cdot|\hat{c})$


(c) Redaction $G'(\cdot|c)$

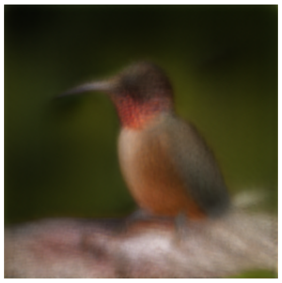
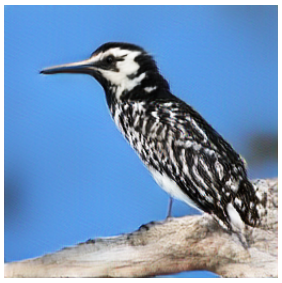
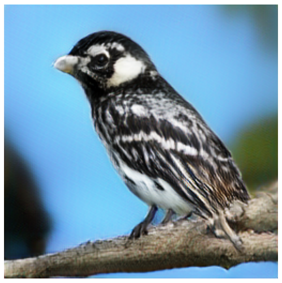
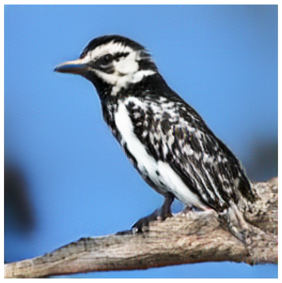

(d) Rewriting Baseline

Figure E.6. Redacted prompt: "a small bird with an orange throat and long beak". Reference prompt: "a small bird with an orange throat and short beak".


(a) Pre-trained $G(\cdot|c)$


(b) Reference $G(\cdot|\hat{c})$


(c) Redaction $G'(\cdot|c)$

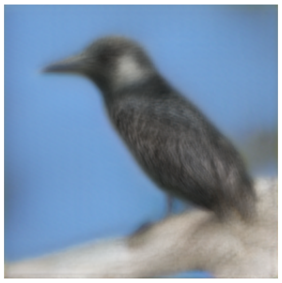
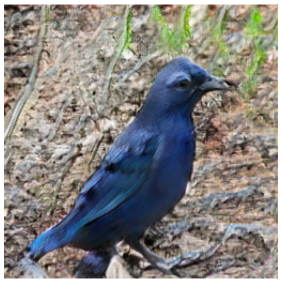
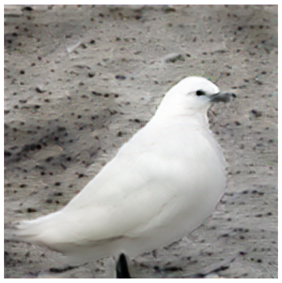
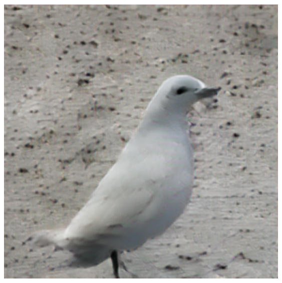

(d) Rewriting Baseline

Figure E.7. Redacted prompt: "the black and white bird has a sharp long beak". Reference prompt: "the black and white bird has a sharp short beak".


(a) Pre-trained $G(\cdot|c)$


(b) Reference $G(\cdot|\hat{c})$


(c) Redaction $G'(\cdot|c)$

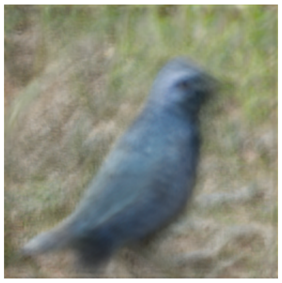
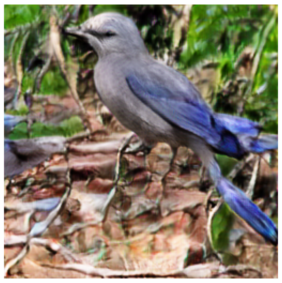
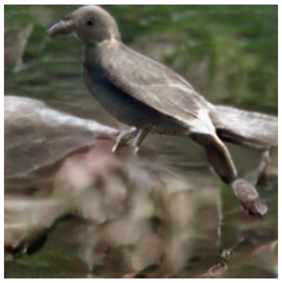
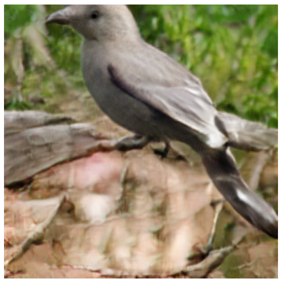

(d) Rewriting Baseline

Figure E.8. Redacted prompt: "this bird has wings that are blue and has black feet". Reference prompt: "this bird has wings that are white and has black feet".


(a) Pre-trained $G(\cdot|c)$


(b) Reference $G(\cdot|\hat{c})$


(c) Redaction $G'(\cdot|c)$

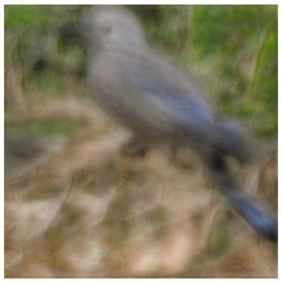
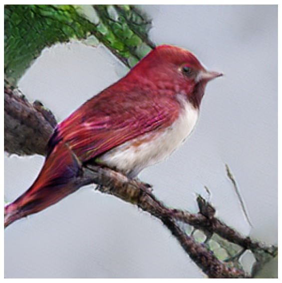
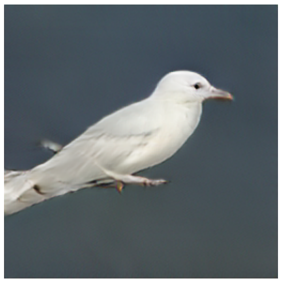
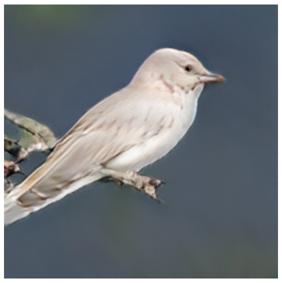

(d) Rewriting Baseline

Figure E.9. Redacted prompt: "this is a grey bird with blue wings and a pointy beak". Reference prompt: "this is a grey bird with white wings and a pointy beak".


(a) Pre-trained $G(\cdot|c)$


(b) Reference $G(\cdot|\hat{c})$


(c) Redaction $G'(\cdot|c)$

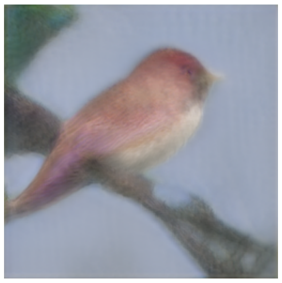
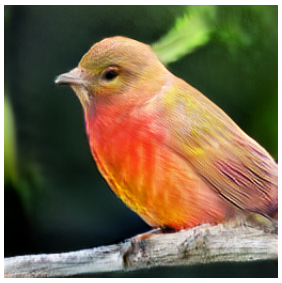
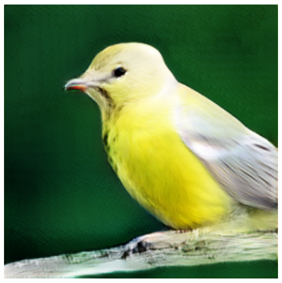
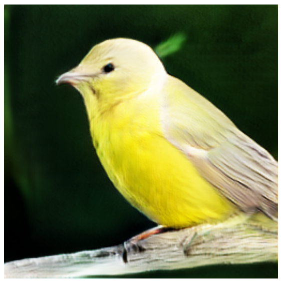

(d) Rewriting Baseline

Figure E.10. Redacted prompt: "this bird has wings that are red and has a white belly". Reference prompt: "this bird has wings that are white and has a white belly".


(a) Pre-trained $G(\cdot|c)$


(b) Reference $G(\cdot|\hat{c})$


(c) Redaction $G'(\cdot|c)$

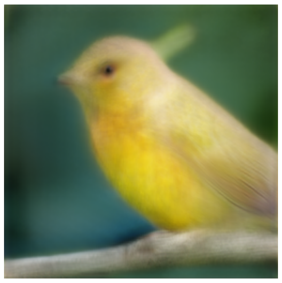
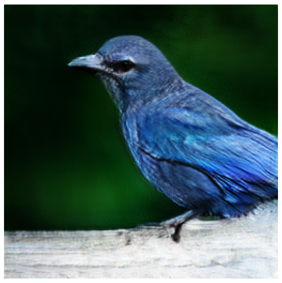
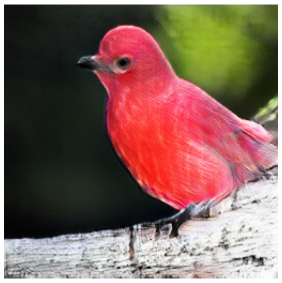
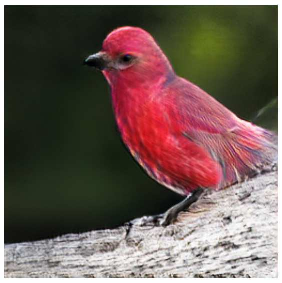

(d) Rewriting Baseline

Figure E.11. Redacted prompt: "this bird has wings that are red and has a yellow belly". Reference prompt: "this bird has wings that are white and has a yellow belly".


(a) Pre-trained $G(\cdot|c)$


(b) Reference $G(\cdot|\hat{c})$


(c) Redaction $G'(\cdot|c)$

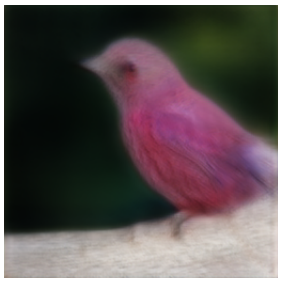
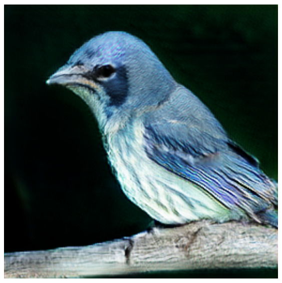
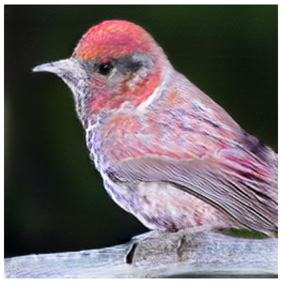
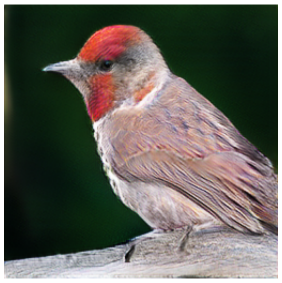

(d) Rewriting Baseline

Figure E.12. Redacted prompt: "this bird has wings that are blue and has black feet". Reference prompt: "this bird has wings that are red and has black feet".


(a) Pre-trained $G(\cdot|c)$


(b) Reference $G(\cdot|\hat{c})$


(c) Redaction $G'(\cdot|c)$

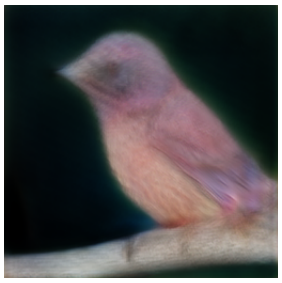
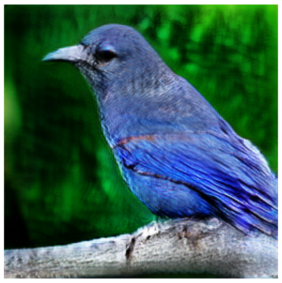
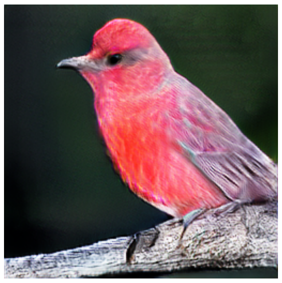
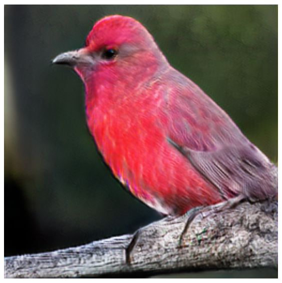

(d) Rewriting Baseline

Figure E.13. Redacted prompt: "this bird has small wings and blue grey nape". Reference prompt: "this bird has small wings and red grey nape".


(a) Pre-trained $G(\cdot|c)$


(b) Reference $G(\cdot|\hat{c})$


(c) Redaction $G'(\cdot|c)$

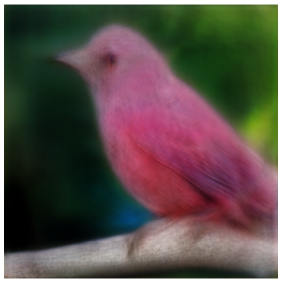
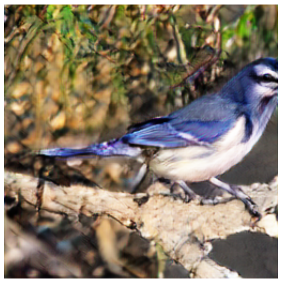
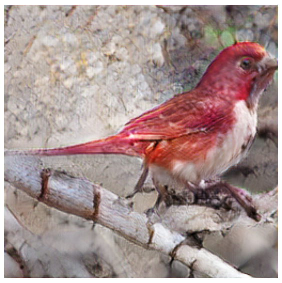
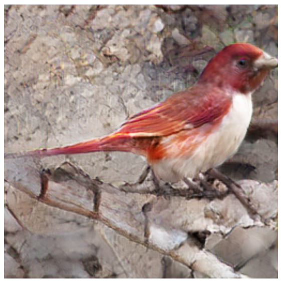

(d) Rewriting Baseline

Figure E.14. Redacted prompt: "the bird is blue with gray wins and tail". Reference prompt: "the bird is red with gray wins and tail".


(a) Pre-trained $G(\cdot|c)$


(b) Reference $G(\cdot|\hat{c})$


(c) Redaction $G'(\cdot|c)$

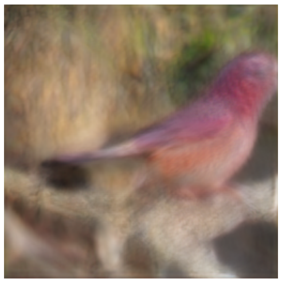
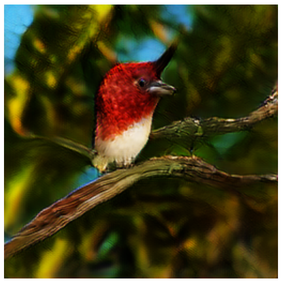
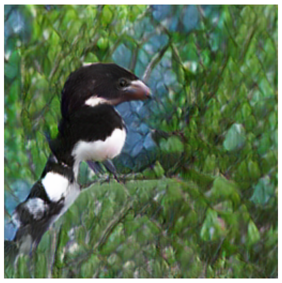
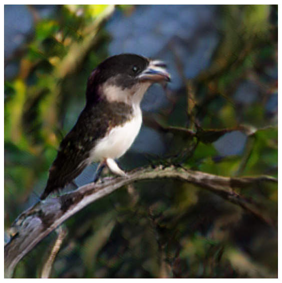

(d) Rewriting Baseline

Figure E.15. Redacted prompt: "this bird has wings that are blue and has a white belly". Reference prompt: "this bird has wings that are red and has a white belly".


(a) Pre-trained $G(\cdot|c)$


(b) Reference $G(\cdot|\hat{c})$


(c) Redaction $G'(\cdot|c)$

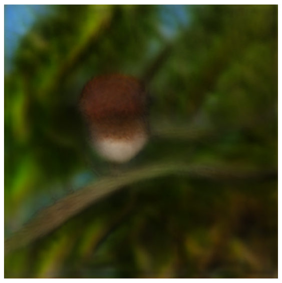
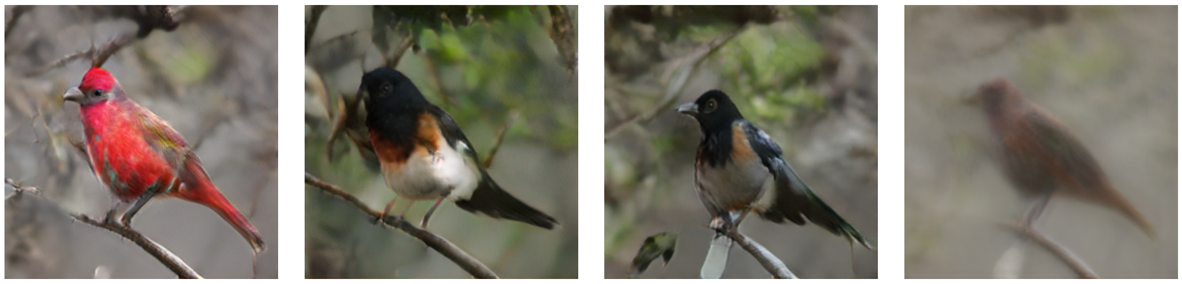
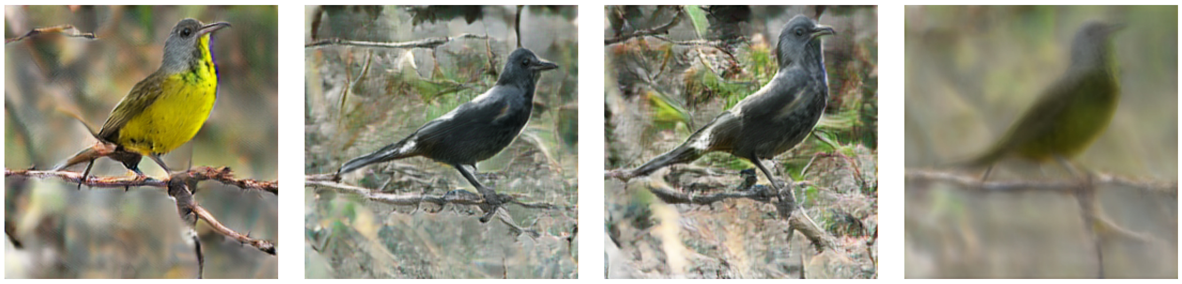

(d) Rewriting Baseline

Figure E.16. Redacted prompt: "this is a red bird with a white belly and a large beak". Reference prompt: "this is a black bird with a white belly and a large beak".



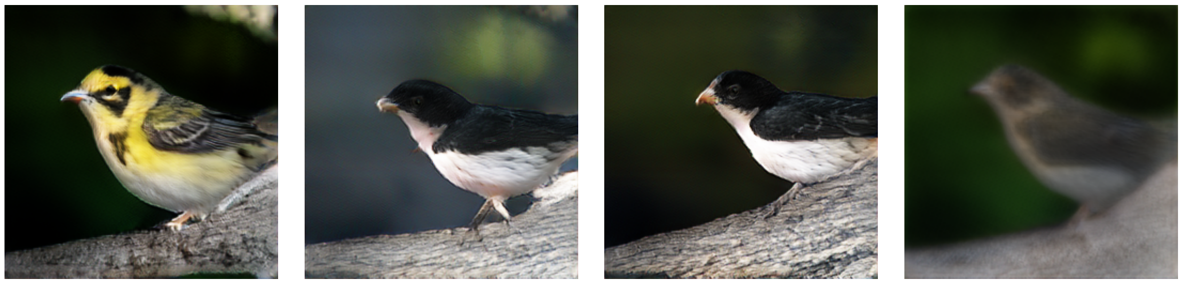
(a) Pre-trained $G(\cdot|c)$ **(b)** Reference $G(\cdot|\hat{c})$ **(c)** Redaction $G'(\cdot|c)$ **(d)** Rewriting Baseline

Figure E.17. Redacted prompt: "a bird with thick short beak red crown red breast that fades into a pink and white belly and red coverts". Reference prompt: "a bird with thick short beak black crown black breast that fades into a pink and white belly and black coverts".



(a) Pre-trained $G(\cdot|c)$ **(b)** Reference $G(\cdot|\hat{c})$ **(c)** Redaction $G'(\cdot|c)$ **(d)** Rewriting Baseline

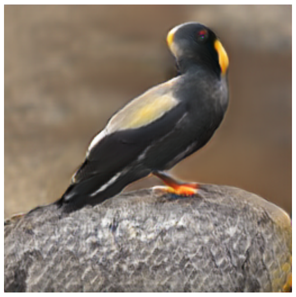
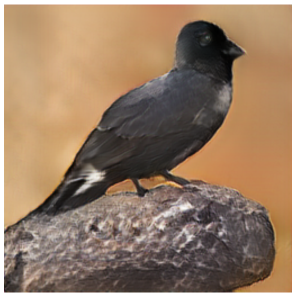
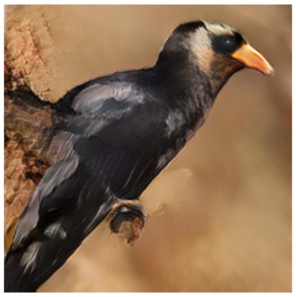
Figure E.18. Redacted prompt: "this yellow breasted bird has a dark gray head and chest a thin beak and a long tail". Reference prompt: "this black breasted bird has a dark gray head and chest a thin beak and a long tail".



(a) Pre-trained $G(\cdot|c)$ **(b)** Reference $G(\cdot|\hat{c})$ **(c)** Redaction $G'(\cdot|c)$ **(d)** Rewriting Baseline

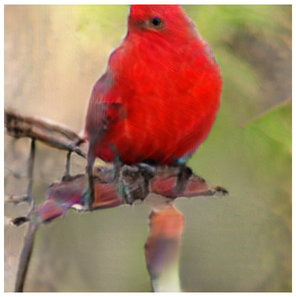
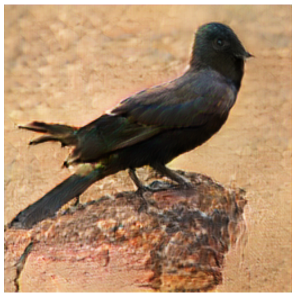
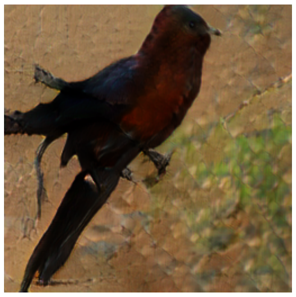
Figure E.19. Redacted prompt: "the yellow and black bird has a white belly". Reference prompt: "the black and black bird has a white belly".

E.2.4 Adversarial Prompting Attack

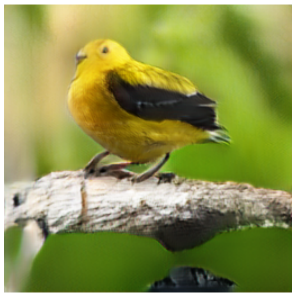
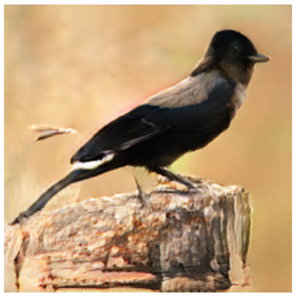
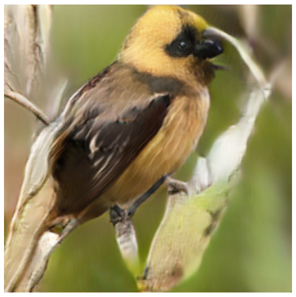
(a) Pre-trained $G(\cdot|c)$ **(b)** Redaction $G'(\cdot|c)$ **(c)** Adversarial $G'(\cdot|c_{\text{adv}})$

Figure E.20. Redacted prompt: "a short yellow beak on an otherwise black bird bright irises around black pupils". Adversarial prompt: "a short yellow wingtip on yhellow otherwise wa bird bright irises around dramatic pupils".

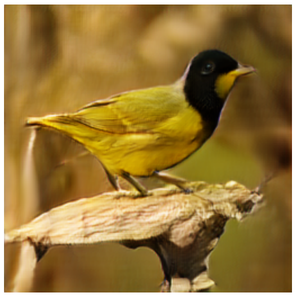
(a) Pre-trained $G(\cdot|c)$ **(b)** Redaction $G'(\cdot|c)$ **(c)** Adversarial $G'(\cdot|c_{\text{adv}})$

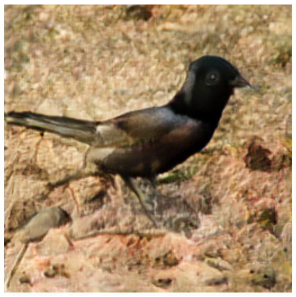
Figure E.21. Redacted prompt: "this is a colorful bird boasting a coral red color strongest at head and fading towards tail". Adversarial prompt: "wingtip is a colorful bird boasting the coral red color strongest at primary and fading towards tail".

(a) Pre-trained $G(\cdot|c)$ **(b)** Redaction $G'(\cdot|c)$ **(c)** Adversarial $G'(\cdot|c_{\text{adv}})$

Figure E.22. Redacted prompt: "this small bird has a light yellow belly and a dark yellow head with black wings and tail". Adversarial prompt: "this small bird the the light yellow belly and a dark yellow head with black remaining and mahogany".


(a) Pre-trained $G(\cdot|c)$


(b) Redaction $G'(\cdot|c)$

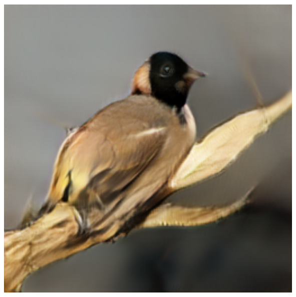
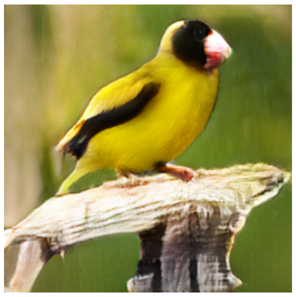
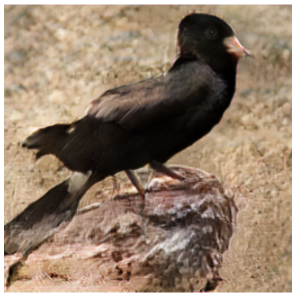

(c) Adversarial $G'(\cdot|c_{\text{adv}})$

Figure E.23. Redacted prompt: "this bird is golden yellow with a black head". Adversarial prompt: "through mahogany wingtip golden yellow bed a black head".


(a) Pre-trained $G(\cdot|c)$


(b) Redaction $G'(\cdot|c)$

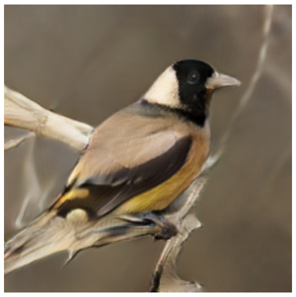
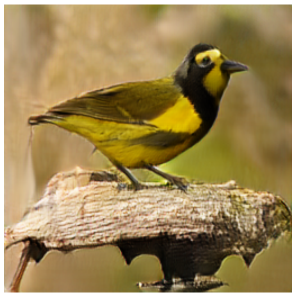
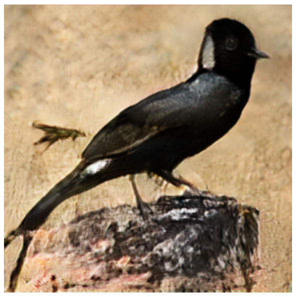

(c) Adversarial $G'(\cdot|c_{\text{adv}})$

Figure E.24. Redacted prompt: "this bird has a golden yellow body a black head a pink bill that curves downward". Adversarial prompt: "mahogany the has blacker golden yellow basically a black head a primary bill that through hughed".


(a) Pre-trained $G(\cdot|c)$


(b) Redaction $G'(\cdot|c)$

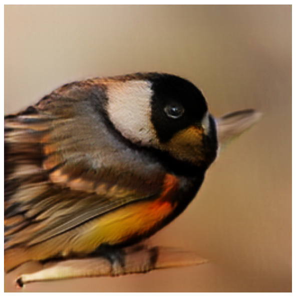

(c) Adversarial $G'(\cdot|c_{\text{adv}})$

Figure E.25. Redacted prompt: "this bird has a black beak a black crown and a belly that is golden yellow". Adversarial prompt: "mahogany love has a riangular love bed black crown and a dramatic hjas breadt golden yellow".

E.3 Additional Details and Experiments for Redaction from DiffWave

E.3.1 Details of the Pre-trained Model

The high-level architecture of DiffWave is shown in Fig. E.26. We select the base (64 channels) version of the model. The model is conditioned on 80-band Mel-spectrogram with FFT size= 1024, hop size= 256, and window size= 1024. Each conditioning network has two up-sampling layers that up-sample the spectrogram, and a one-dimensional convolution layer that maps the number of channels to 128. We use the pre-trained model and code from https://github.com/philsyn/DiffWave-Vocoder under MIT license, which is trained on all LJSpeech samples except for LJ001 and LJ002, which is used as the test set. The pre-trained model takes days to train on 8 GPUs.

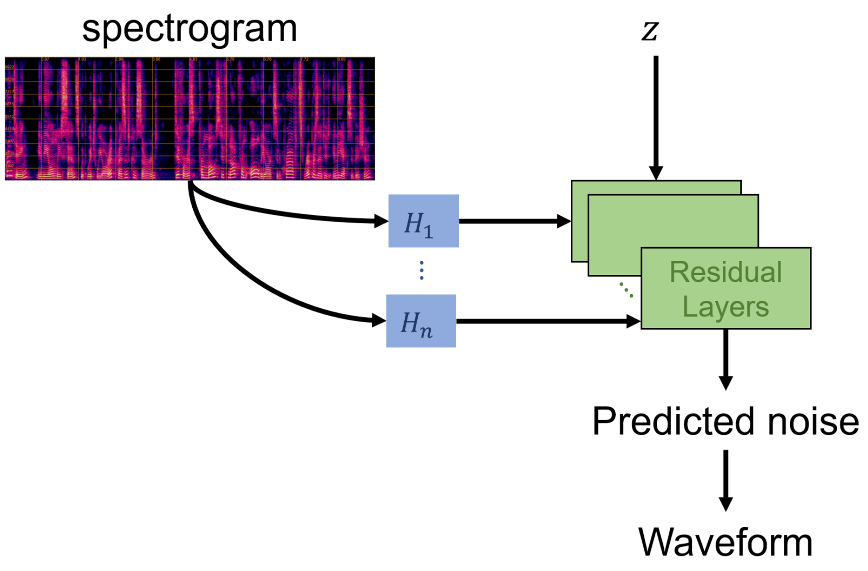


Figure E.26. High-level architecture of DiffWave.

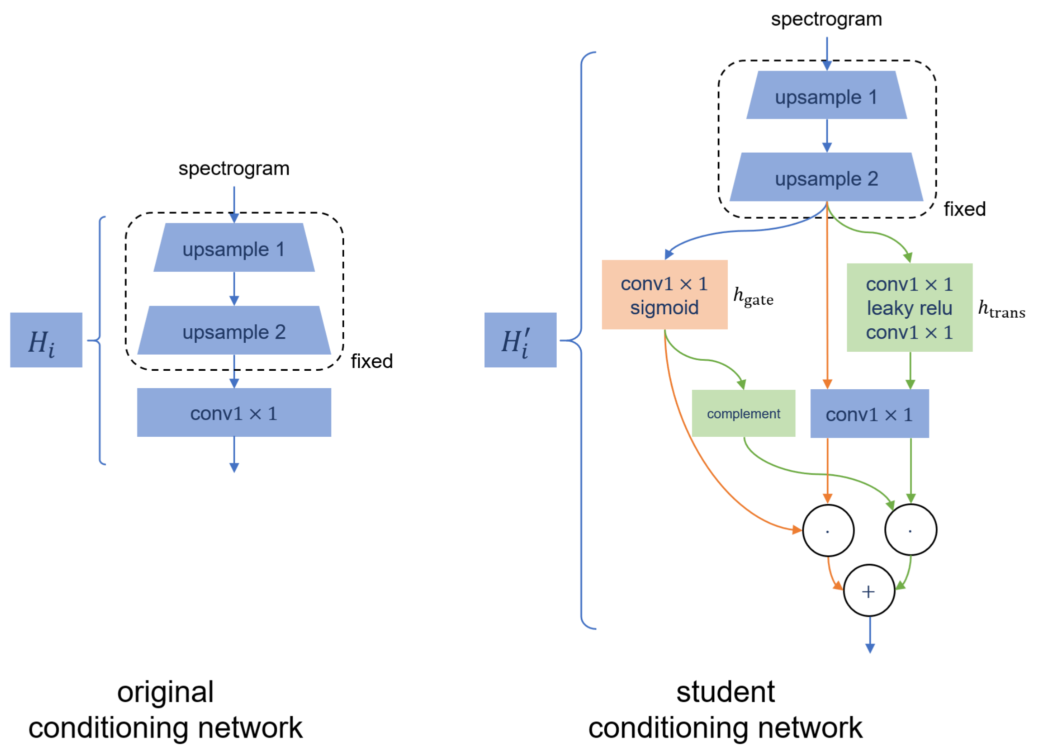


Figure E.27. High-level architecture of original and higher-capacity conditioning networks of DiffWave.

E.3.2 Redaction Setup

We use one NVIDIA 3080 GPU to run experiments. We use the Adam optimizer with a learning rate 0.001 to optimize the ℓ_1 loss. The redaction algorithm terminates at 80000 iterations. We use a batch size of 32. Table E.2 includes the specific train-test splits of the LibriTTS voices. Note that for `speaker 1040` there is only one chapter id, so we split based on the segment id shown in columns.

We use the code from https://github.com/jackaduma/CycleGAN-VC2 under MIT license to train CycleGAN-VC2. The training data for CycleGAN-VC2 is the training data of a LibriTTS voice and the first 100 samples of LJ003 from LJSpeech. We train CycleGAN-VC2 for 1000 iterations with a batch size of 8. We use the Whisper (medium-sized English-only) model from https://github.com/openai/whisper under MIT license. We use the Tortoise-TTS model from

Table E.2. Specific train-test splits of the LibriTTS voices, and their total lengths measured in minutes.

Redaction voices	training		test	
	chapter id	length	chapter id	length
`speaker 125`	121124	5.89	121342	2.30
`speaker 1578`	140045, 140049	4.81	6379	1.30
`speaker 1737`	142397, 148989, 142396	3.75	146161	2.51
`speaker 1926`	147979, 147987	5.44	143879	1.98
`speaker 1040`	133433 (0-98)	4.65	133433 (100-168)	2.35

https://github.com/neonbjb/tortoise-tts under Apache-2.0 license. To sample from Tortoise-TTS we use two 10-second utterances from LJSpeech.

For the additional layers in the improved capacity configuration, all convolutions are one-dimensional with kernel size $= 1$. h_{trans} includes two convolutions that keep the channels ($= 80$) and a leaky ReLU activation with negative slope $= 0.4$ between. h_{gate} includes one zero-initialized convolution that changes channels from 80 to 128 followed by a sigmoid activation. The architecture of student conditioning networks with improved capacity is shown in Fig. E.27.

E.3.3 Evaluation Metrics

The metrics for speech quality are as follows.

1. Perceptual Evaluation of Speech Quality [Recommendation, 2001], or PESQ, measures the quality of generated speech. It is between -0.5 and 4.5 and is higher for better quality.

2. Short-Time Objective Intelligibility [Taal et al., 2011], or STOI, measures the intelligibility of generated speech. It ranges between 0% and 100% and is higher for better intelligibility.

The voice classifier is trained and tested on audio clips with 0.7256 second. For each audio clip, we extract 20-dimensional Mel-frequency cepstral coefficients [Xu et al., 2005], 7-dimensional spectral contrast [Jiang et al., 2002], and 12-dimensional chroma features [Ellis, 2007]. The classifier is a support vector classifier with the radial basis function kernel with regularization coefficient $= 1$.

Appendix F

Approximate Data Deletion in Generative Models

F.1 Notation Tables

Table F.1. Abbreviations used in Chapter 6 and Appendix F.

DRE	density ratio estimator
MMD^2	squared MMD metric
$\widehat{\mathrm{MMD}}_u^2$	unbiased MMD estimator
LR	likelihood ratio
D_ϕ	the ϕ-divergence
$\widehat{\mathrm{ASC}}_\phi$	ASC statistic, or the ϕ-divergence estimator
KDE	kernel density estimator
KBC	kernel-based classifier
kNN	k nearest neighbour classifier

Table F.2. Notations used in Chapter 6 and Appendix F.

p_*	data distribution
X	training set: N i.i.d. samples from p_*
X'	deletion set: N' samples from X
$\mathscr{A}$	algorithm of the generative model
$\hat{p}$	pretrained generative model on X
$\hat{p}'$	retrained generative model on $X \setminus X'$
p'_*	distribution s.t. $X \setminus X'$ are i.i.d. samples from p'_*
ρ_*	density ratio p'_*/p_*
$\hat{\rho}$	density ratio $\hat{p}'/\hat{p}$
$\hat{\rho}_{\mathscr{E}}$	abbreviation for DRE$(X, X \setminus X')$; DRE between X and $X \setminus X'$
$\mathscr{D}(\hat{p}, X, X')$	approximate deletion $\hat{\rho}_{\mathscr{E}} \cdot \hat{p}$
q	the distribution to be tested
Y	m i.i.d. samples from q
$Y_{\mathscr{H}_i}$	m i.i.d. samples from q under $\mathscr{H}_i$, $i = 1, 2$
$\hat{Y}$	m i.i.d. samples from the pretrained model $\hat{p}$
$Y_{\mathscr{D}}$	m i.i.d. samples from the approximate deletion $\mathscr{D}(\hat{p}, X, X')$

F.2 Theory for the Framework in Section 6.2

F.2.1 Omitted Proofs in Section 6.2.2

Proof of Thm. 6.2.2

Proof. Notice that

$$\hat{\rho} = \frac{\hat{p}'}{\hat{p}} = \frac{\hat{p}'}{p'_*} \cdot \frac{p'_*}{p_*} \cdot \frac{p_*}{\hat{p}}.$$

With probability at least $1 - \delta_N$

$$\frac{1}{c_N} \le \frac{p_*}{\hat{p}} \le c_N.$$

With probability at least $1 - \delta_{N-N'}$

$$\frac{1}{c_{N-N'}} \le \frac{\hat{p}'}{p'_*} \le c_{N-N'}.$$

Therefore, with probability at least $1-\delta_N-\delta_{N-N'}$,

$$\begin{aligned}\int_{\mathbb{R}^d}\hat{p}\,|\hat{\rho}-\rho_*|\,dx &= \int_{\mathbb{R}^d}p'_*\left|\frac{\hat{p}'}{p'_*}-\frac{\hat{p}}{p_*}\right|dx\\ &\leq \max\left(c_N-\frac{1}{c_{N-N'}},c_{N-N'}-\frac{1}{c_N}\right)\\ &\leq 2(c_N+c_{N-N'}-2).\end{aligned}$$

Now, we choose a fixed RC algorithm $\mathscr{A}_0$, and define $\hat{\rho}_{\mathscr{E}}(Z_1,Z_2)=\rho\left(p_{\mathscr{A}_0(Z_1)},p_{\mathscr{A}_0(Z_2)}\right)$. Then, with probability at least $1-\delta_N-\delta_{N-N'}$,

$$\int_{\mathbb{R}^d}\hat{p}\,|\hat{\rho}_{\mathscr{E}}-\rho_*|\,dx\leq 2(c_N+c_{N-N'}-2).$$

Therefore, with probability at least $1-2\delta_N-2\delta_{N-N'}$,

$$\|\hat{\rho}_{\mathscr{E}}\cdot\hat{p}-\hat{p}'\|_1=\int_{\mathbb{R}^d}\hat{p}\,|\hat{\rho}_{\mathscr{E}}-\hat{\rho}|\,dx\leq 4(c_N+c_{N-N'}-2).$$

□

Proof of Thm. 6.2.4

Proof. Notice that

$$\begin{aligned}\int_{\mathbb{R}^d}\hat{p}^2\,|\hat{\rho}-\rho_*|\,dx &= \int_{\mathbb{R}^d}\hat{p}\,\left|\hat{p}'-\rho_*\hat{p}\right|dx\\ &= \int_{\mathbb{R}^d}\hat{p}\,\left|\hat{p}'-p'_*+p'_*-\rho_*(\hat{p}-p_*+p_*)\right|dx\\ &= \int_{\mathbb{R}^d}\hat{p}\,\left|\hat{p}'-p'_*-\rho_*(\hat{p}-p_*)\right|dx.\end{aligned}$$

With probability at least $1-\delta_N$, $|\hat{p}-p_*|\leq\varepsilon_N$, and with probability at least $1-\delta_{N-N'}$, $|\hat{p}'-p'_*|\leq\varepsilon_{N-N'}$. Therefore, with probability at least $1-\delta_N-\delta_{N-N'}$,

$$\int_{\mathbb{R}^d}\hat{p}^2\,|\hat{\rho}-\rho_*|\,dx\leq\varepsilon_{N-N'}+\|\rho_*\|_\infty\varepsilon_N.$$

Now, we choose a fixed TVC algorithm $\mathscr{A}_0$, and define $\hat{\rho}_{\mathscr{E}}(Z_1,Z_2)=\rho(p_{\mathscr{A}_0(Z_1)},p_{\mathscr{A}_0(Z_2)})$. Then, with probability at least $1-\delta_N-\delta_{N-N'}$,

$$\int_{\mathbb{R}^d}\hat{p}^2\,|\hat{\rho}_{\mathscr{E}}-\rho_*|\,dx\leq\varepsilon_{N-N'}+\|\rho_*\|_\infty\varepsilon_N.$$

Therefore, with probability at least $1-2\delta_N-2\delta_{N-N'}$,

$$\|\hat{\rho}_{\mathscr{E}}\cdot\hat{p}-\hat{p}'\|_{1,\hat{p}}=\int_{\mathbb{R}^d}\hat{p}^2\,|\hat{\rho}-\hat{\rho}_{\mathscr{E}}|\,dx\leq2\left(\varepsilon_{N-N'}+\|\rho_*\|_\infty\varepsilon_N\right).$$

□

F.2.2 Omitted Proofs in Section 6.2.3

Proof of Thm. 6.2.8

Proof. By taking $Z_0=Z$, $Z_1=Z\setminus\{z\}$, and $\hat{Z}=\{\hat{z}\}$, we conclude ε-DP implies ε-US. By taking one side of the ε-US bound, we conclude ε-US implies ε-LBLI.

Define

$$\hat{\rho}_k=\frac{p_{\mathscr{A}(X\setminus X'_{1:k-1})}}{p_{\mathscr{A}(X\setminus X'_{1:k})}}$$

for $k=1,\cdots,N'$. Then, ε-LBLI indicates $\log\|\hat{\rho}_k\|_\infty\leq\varepsilon$. Notice that

$$\hat{\rho}=\prod_{k=1}^{N'}\hat{\rho}_k.$$

Therefore, we have $\log\|\hat{\rho}\|_\infty\leq N'\varepsilon$. □

Proof of Thm. 6.2.10

Proof. With probability at least $1-\delta_N$,

$$-\log c_N \leq \log \rho(\mu_i, p_{\mathscr{A}(Z_i)}) \leq \log c_N.$$

Therefore, with probability at least $1-2\delta_N$,

$$\left\|\log \rho(\mu_1, p_{\mathscr{A}(Z_1)}) - \log \rho(\mu_2, p_{\mathscr{A}(Z_2)})\right\|_\infty \leq 2\log c_N.$$

□

Proof of Thm. 6.2.11

Proof. Define $Z_k = X \setminus X'_{1:k}$ and μ_k be the distribution such that Z_k contains i.i.d. samples from μ_k. Specifically, $\mu_0 = p_*$ and $\mu_{N'} = p'_*$. Then, we have

$$\begin{aligned}\log \rho_* - \log \hat{\rho} &= \log \frac{\mu_{N'}}{\mu_0} - \log \frac{p_{\mathscr{A}(Z_{N'})}}{p_{\mathscr{A}(Z_0)}} \\ &= \sum_{k=1}^{N'} \left(\log \frac{\mu_k}{\mu_{k-1}} - \log \frac{p_{\mathscr{A}(Z_k)}}{p_{\mathscr{A}(Z_{k-1})}} \right) \\ &= \sum_{k=1}^{N'} \left(\log \rho(\mu_{k-1}, p_{\mathscr{A}(Z_{k-1})}) - \log \rho(\mu_k, p_{\mathscr{A}(Z_k)}) \right).\end{aligned}$$

Therefore, with probability at least $1-N'\delta$, we have

$$\|\log \rho_* - \log \hat{\rho}\|_\infty \leq N'\varepsilon,$$

which indicates $\log \|\hat{\rho}\|_\infty \leq N'\varepsilon + \log \|\rho_*\|_\infty$. □

Proof of Thm. 6.2.13

Proof. By rewriting ES for $\hat{p}$ and $\hat{p}'$, we have

$$\mathbb{E}_{x\sim\hat{p}} \log\hat{\rho} = -\mathrm{KL}\left(\hat{p}\|\hat{p}'\right),$$

$$\mathbb{E}_{x\sim\hat{p}}(\log\hat{\rho})^2 \leq \varepsilon.$$

Because

$$\mathbb{E}_{x\sim\hat{p}}(\log\hat{\rho})^2 \geq \left(\mathbb{E}_{x\sim\hat{p}} \log\hat{\rho}\right)^2,$$

we have $\mathrm{KL}\left(\hat{p}\|\hat{p}'\right) \leq \sqrt{\varepsilon}$. Then, according to Cantelli's inequality Cantelli [1910], for any positive a,

$$\mathrm{Prob}\left(\log\hat{\rho} \geq -\mathrm{KL}\left(\hat{p}\|\hat{p}'\right) + a\right) \leq \frac{\mathbb{VAR}(\log\hat{\rho})}{\mathbb{VAR}(\log\hat{\rho}) + a^2}.$$

By letting

$$a = \sqrt{\frac{1-\delta}{\delta} \cdot \mathbb{VAR}(\log\hat{\rho})},$$

we have with probability at least $1-\delta$ for samples $x \sim \hat{p}$,

$$\begin{aligned}\log\hat{\rho}(x) &\leq \sqrt{\frac{1-\delta}{\delta} \cdot \left(\mathbb{E}_{x\sim\hat{p}}(\log\hat{\rho})^2 - \mathrm{KL}\left(\hat{p}\|\hat{p}'\right)^2\right)} - \mathrm{KL}\left(\hat{p}\|\hat{p}'\right) \\ &\leq \sqrt{\frac{\varepsilon(1-\delta)}{\delta}}.\end{aligned}$$

$\square$

F.3 Statistical Tests in Section 6.4

F.3.1 Likelihood Ratio Tests

Proof of Thm. 6.4.1

Proof. By definition of RC, we have with probability at least $1-\delta_N$,

$$|\log \hat{p} - \log p_*| \leq \log c_N,$$

and with probability at least $1-\delta_{N-N'}$,

$$|\log \hat{p}' - \log p'_*| \leq \log c_{N-N'}.$$

Therefore, with probability at least $1-\delta_N-\delta_{N-N'}$,

$$|\log \hat{\rho} - \log \rho_*| \leq \log c_{N-N'} + \log c_N.$$

Now, we choose a fixed RC algorithm $\mathscr{A}_0$, and define $\hat{\rho}_{\mathscr{E}}(Z_1,Z_2) = \rho(p_{\mathscr{A}_0(Z_1)}, p_{\mathscr{A}_0(Z_2)})$. Then, we also have with probability at least $1-\delta_N-\delta_{N-N'}$,

$$|\log \hat{\rho}_{\mathscr{E}} - \log \rho_*| \leq \log c_{N-N'} + \log c_N.$$

Therefore, with probability at least $1-2(\delta_N+\delta_{N-N'})$,

$$|\log \hat{\rho} - \log \hat{\rho}_{\mathscr{E}}| \leq 2(\log c_{N-N'} + \log c_N),$$

and the conclusion follows. □

Proof of Thm. 6.4.2

Proof. (1) Notice that

$$\begin{aligned}|\mathrm{LR}(Y,\hat{p},\hat{p}') - \mathrm{LR}(Y,\hat{p},\hat{\rho}_{\mathscr{E}}\cdot\hat{p})| &= \frac{1}{m}\sum_{y\in Y}|\log\hat{\rho}(y) - \log\hat{\rho}_{\mathscr{E}}(y)| \\ &\leq \frac{1}{m}\cdot m\varepsilon \\ &= \varepsilon.\end{aligned}$$

(2) If $\mathscr{H}_0$ is true, then $Y \sim \hat{p}$. Then,

$$\begin{aligned}\mathbb{E}_Y|\mathrm{LR}(Y,\hat{p},\hat{p}') - \mathrm{LR}(Y,\hat{p},\hat{\rho}_{\mathscr{E}}\cdot\hat{p})| &= \mathbb{E}_Y\left|\frac{1}{m}\sum_{y\in Y}(\log\hat{\rho}(y) - \log\hat{\rho}_{\mathscr{E}}(y))\right| \\ &\leq \mathbb{E}_Y\left(\frac{1}{m}\sum_{y\in Y}|\log\hat{\rho}(y) - \log\hat{\rho}_{\mathscr{E}}(y)|\right) \\ &= \mathbb{E}_{y\sim\hat{p}}|\log\hat{\rho}(y) - \log\hat{\rho}_{\mathscr{E}}(y)| \\ &= \|\log\hat{\rho} - \log\hat{\rho}_{\mathscr{E}}\|_{1,\hat{p}} \\ &\leq \varepsilon.\end{aligned}$$

By Markov's inequality, we have with probability at least $1-\delta$, $|\mathrm{LR}(Y,\hat{p},\hat{p}') - \mathrm{LR}(Y,\hat{p},\hat{\rho}_{\mathscr{E}}\cdot\hat{p})| \leq \varepsilon/\delta$. The proof for $\mathscr{H}_1$ is similar. □

Statistical properties of LR **statistics.**

Let $\phi(t) = \log(t)^2$. When $\mathscr{H}_0$ is true, we have

$$\begin{aligned}\mathbb{E}_{Y\sim\hat{p}}\ \mathrm{LR}(Y,\hat{p},\hat{p}') &= \mathbb{E}_{\hat{p}}\log\frac{\hat{p}'}{\hat{p}} = -\mathrm{KL}\left(\hat{p}\|\hat{p}'\right), \\ \mathbb{VAR}_{Y\sim\hat{p}}\ \mathrm{LR}(Y,\hat{p},\hat{p}') &= \frac{1}{m}\left(\mathbb{E}_{\hat{p}}\left(\log\frac{\hat{p}'}{\hat{p}}\right)^2 - \left(\mathbb{E}_{\hat{p}}\log\frac{\hat{p}'}{\hat{p}}\right)^2\right) \\ &= \frac{1}{m}\left(D_{\log^2}(\hat{p}\|\hat{p}') - \mathrm{KL}\left(\hat{p}\|\hat{p}'\right)^2\right).\end{aligned}$$

When $\mathscr{H}_1$ is true, we have

$$\begin{aligned}
\mathbb{E}_{Y\sim\hat{p}'}\ \mathrm{LR}(Y,\hat{p},\hat{p}') &= \mathbb{E}_{\hat{p}'}\log\frac{\hat{p}'}{\hat{p}} = \mathrm{KL}\left(\hat{p}'\|\hat{p}\right),\\
\mathbb{VAR}_{Y\sim\hat{p}'}\ \mathrm{LR}(Y,\hat{p},\hat{p}') &= \frac{1}{m}\left(\mathbb{E}_{\hat{p}'}\left(\log\frac{\hat{p}'}{\hat{p}}\right)^2 - \left(\mathbb{E}_{\hat{p}'}\log\frac{\hat{p}'}{\hat{p}}\right)^2\right)\\
&= \frac{1}{m}\left(D_{\log^2}(\hat{p}'\|\hat{p}) - \mathrm{KL}\left(\hat{p}'\|\hat{p}\right)^2\right).
\end{aligned}$$

F.3.2 ASC Tests

Proof of Thm. 6.4.3

Proof. Take expectations $Y\sim q$ and $\hat{Y}\sim\hat{p}$. Then, we have

$$\begin{aligned}
&\mathbb{E}|\mathrm{A\hat{S}C}_\phi(\hat{Y},Y,\hat{\rho}) - \mathrm{A\hat{S}C}_\phi(\hat{Y},Y,\hat{\rho}_{\mathscr{E}})|\\
&= \mathbb{E}\left|\frac{1}{m}\left(\sum_{y\in\hat{Y}} + \sum_{y\in Y}\right)(\psi(\hat{\rho}(y)) - \psi(\hat{\rho}_{\mathscr{E}}(y)))\right|\\
&\le \mathbb{E}\left(\frac{1}{m}\sum_{y\in\hat{Y}}|\psi(\hat{\rho}(y)) - \psi(\hat{\rho}_{\mathscr{E}}(y))|\right)\\
&+\mathbb{E}\left(\frac{1}{m}\sum_{y\in Y}|\psi(\hat{\rho}(y)) - \psi(\hat{\rho}_{\mathscr{E}}(y))|\right)\\
&= \mathbb{E}_{y\sim\hat{p}}|\psi(\hat{\rho}(y)) - \psi(\hat{\rho}_{\mathscr{E}}(y))| + \mathbb{E}_{y\sim q}|\psi(\hat{\rho}(y)) - \psi(\hat{\rho}_{\mathscr{E}}(y))|\\
&= \|\psi(\hat{\rho}) - \psi(\hat{\rho}_{\mathscr{E}})\|_{1,\hat{p}} + \|\psi(\hat{\rho}) - \psi(\hat{\rho}_{\mathscr{E}})\|_{1,q}\\
&\le 2\varepsilon.
\end{aligned}$$

By Markov's inequality, we have with probability at least $1-\delta$, it holds that $|\mathrm{A\hat{S}C}_\phi(\hat{Y},Y,\hat{\rho}) - \mathrm{A\hat{S}C}_\phi(\hat{Y},Y,\hat{\rho}_{\mathscr{E}})| \le 2\varepsilon/\delta$.

□

Statistical properties of ASC **statistics.**

When $\mathscr{H}_0$ is true, we have

$$\mathbb{E}_{Y\sim\hat{p},\hat{Y}\sim\hat{p}}\ \ \mathrm{A\hat{S}C}_\phi(\hat{Y},Y,\hat{\rho}) = \mathbb{E}_{\hat{p}}\left(\frac{2\phi(\hat{\rho}(y))}{1+\hat{\rho}(y)}\right).$$

When $\mathscr{H}_1$ is true, we have

$$\begin{aligned}\mathbb{E}_{Y\sim\hat{p}',\hat{Y}\sim\hat{p}}\ \ \mathrm{A\hat{S}C}_\phi(\hat{Y},Y,\hat{\rho}) &= (\mathbb{E}_{\hat{p}}+\mathbb{E}_{\hat{p}'})\frac{\phi(\hat{\rho})}{1+\hat{\rho}} \\ &= \mathbb{E}_{\hat{p}}(1+\hat{\rho})\cdot\frac{\phi(\hat{\rho})}{1+\hat{\rho}} \\ &= \mathbb{E}_{\hat{p}}\left(\phi(\hat{\rho}(y))\right).\end{aligned}$$

F.3.3 MMD Tests

Definition of MMD.

Let $K_{\mathrm{MMD}}(\cdot,\cdot)$ be a kernel function. The Maximum Mean Discrepancy (MMD) Gretton et al. [2012] between $\hat{p}$ and q is defined as

$$\mathrm{MMD}^2(q,\hat{p}) = \left(\mathbb{E}_{x,y\sim\hat{p}} - 2\mathbb{E}_{x\sim\hat{p},y\sim q} + \mathbb{E}_{x,y\sim q}\right)K_{\mathrm{MMD}}(x,y).$$

Given m i.i.d. samples $\hat{Y}\sim\hat{p}$ and m i.i.d. samples $Y\sim q$, an unbiased estimator of MMD^2 is

$$\mathrm{M\hat{M}D}_u^2(Y,\hat{Y}) = \frac{1}{m(m-1)}\sum_{i\neq j}(K_{\mathrm{MMD}}(y_i,y_j)+K_{\mathrm{MMD}}(\hat{y}_i,\hat{y}_j)) - \frac{2}{m^2}\sum_{i,j}K_{\mathrm{MMD}}(y_i,\hat{y}_j).$$

Asymptotic and concentration propertiesSerfling [2009], Gretton et al. [2012].

Define

$$h((y_i,\hat{y}_i),(y_j,\hat{y}_j)) = K_{\mathrm{MMD}}(y_i,y_j)+K_{\mathrm{MMD}}(\hat{y}_i,\hat{y}_j)-K_{\mathrm{MMD}}(y_i,\hat{y}_j)-K_{\mathrm{MMD}}(y_j,\hat{y}_i).$$

Then, we have

$$\hat{\text{MMD}}_u^2(Y,\hat{Y}) = \frac{1}{m(m-1)}\sum_{i\neq j}^{m} h((y_i,\hat{y}_i),(y_j,\hat{y}_j)).$$

Define

$$\begin{aligned}\sigma_u^2 &= 4\left(\mathbb{E}_{\substack{y\sim q\\ \hat{y}\sim\hat{p}}}\left[\mathbb{E}_{\substack{y'\sim q\\ \hat{y}'\sim\hat{p}}} h((y,\hat{y}),(y',\hat{y}'))\right]^2 - \left[\mathbb{E}_{\substack{y,y'\sim q\\ \hat{y},\hat{y}'\sim\hat{p}}} h((y,\hat{y}),(y',\hat{y}'))\right]^2\right)\\ &= 4\cdot\mathbb{E}_{\substack{y\sim q\\ \hat{y}\sim\hat{p}}}\mathbb{VAR}_{\substack{y'\sim q\\ \hat{y}'\sim\hat{p}}}\ h((y,\hat{y}),(y',\hat{y}')).\end{aligned}$$

Then, it holds that

$$\sqrt{m}\left(\hat{\text{MMD}}_u^2(Y,\hat{Y}) - \text{MMD}^2(q,\hat{p})\right)\to\mathcal{N}(0,\sigma_u^2)\text{ in distribution}$$

As for concentration properties, with probability at least $1-\delta$, it holds that

$$\text{MMD}_u^2(Y,\hat{Y}) - \text{MMD}^2(q,\hat{p}) \leq 4\sqrt{\frac{1}{m}\log\frac{1}{\delta}}\cdot\sup_{x,y} K_{\text{MMD}}(x,y),$$

with have the same bound on the other side.

Asymptotic and concentration properties in the context of deletion test.

Now, we look at these properties in the context of deletion test. If $\mathscr{H}_0$ is true,

$$\begin{aligned}\mathbb{E}_{Y\sim\hat{p}}\ \hat{\text{MMD}}_u^2(Y,\hat{Y}) &= 0,\\ \mathbb{VAR}_{Y\sim\hat{p}}\ \hat{\text{MMD}}_u^2(Y,\hat{Y}) &= \frac{4}{m}\cdot\mathbb{E}_{\substack{y\sim\hat{p}\\ \hat{y}\sim\hat{p}}}\mathbb{VAR}_{\substack{y'\sim\hat{p}\\ \hat{y}'\sim\hat{p}}}\ h((y,\hat{y}),(y',\hat{y}')).\end{aligned}$$

And with probability at least $1-\delta$,

$$\left|\hat{\text{MMD}}_u^2(Y,\hat{Y})\right| \leq 4\sqrt{\frac{1}{m}\log\frac{2}{\delta}}\cdot\sup_{x,y} K_{\text{MMD}}(x,y).$$

If $\mathscr{H}_1$ is true,

$$\begin{aligned}
\mathbb{E}_{Y\sim\hat{p}'} \ \ \hat{\mathrm{MMD}}_u^2(Y,\hat{Y}) &= \mathrm{MMD}^2(\hat{p}',\hat{p}),\\
\mathbb{VAR}_{Y\sim\hat{p}} \ \ \hat{\mathrm{MMD}}_u^2(Y,\hat{Y}) &= \frac{4}{m}\cdot \mathbb{E}_{\substack{y\sim q\\ \hat{y}\sim\hat{p}}} \mathbb{VAR}_{\substack{y'\sim q\\ \hat{y}'\sim\hat{p}}} \ \ h((y,\hat{y}),(y',\hat{y}')).
\end{aligned}$$

And with probability at least $1-\delta$,

$$\left|\hat{\mathrm{MMD}}_u^2(Y,\hat{Y}) - \mathrm{MMD}^2(\hat{p}',\hat{p})\right| \le 4\sqrt{\frac{1}{m}\log\frac{2}{\delta}}\cdot \sup_{x,y} K_{\mathrm{MMD}}(x,y).$$

Example F.3.1 (KDE). *Now, we compute* $\mathrm{MMD}(\hat{p}',\hat{p})^2$ *for KDE with the standard Gaussian kernel. We let* K_{MMD} *be the standard RBF kernel:* $K_{\mathrm{MMD}}(x,y)=\exp(-\|x-y\|^2/2)$*. Let* $x,x'\sim q$*,* $y,y'\sim\hat{p}$*, and* $z_i,z_i'\sim\mathcal{N}(x_i,I)$*. Then,*

$$\begin{aligned}
\mathbb{E}_{x,x'}K_{\mathrm{MMD}}(x,x') &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{E}_{z_i,z_j'}K_{\mathrm{MMD}}(z_i,z_j')\\
\mathbb{E}_{y,y'}K_{\mathrm{MMD}}(y,y') &= \frac{1}{(N-N')^2}\sum_{i=N'+1}^{N}\sum_{j=N'+1}^{N}\mathbb{E}_{z_i,z_j'}K_{\mathrm{MMD}}(z_i,z_j')\\
\mathbb{E}_{x,y}K_{\mathrm{MMD}}(x,y) &= \frac{1}{N(N-N')}\sum_{i=1}^{N}\sum_{j=N'+1}^{N}\mathbb{E}_{z_i,z_j'}K_{\mathrm{MMD}}(z_i,z_j').
\end{aligned}$$

By rearranging, we have

$$\begin{aligned}
&\mathrm{MMD}^2(\hat{p}',\hat{p})\\
&=\left(\frac{N'^2}{N^2(N-N')^2}\sum_{i=N'+1}^{N}\sum_{j=N'+1}^{N} - \frac{N+N'}{N^2(N-N')}\sum_{i=1}^{N'}\sum_{j=N'+1}^{N} + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N'}\right)\mathbb{E}_{z_i,z_j'}K_{\mathrm{MMD}}(z_i,z_j').
\end{aligned}$$

We then compute $\mathbb{E}_{z_i,z'_j} K_{\mathrm{MMD}}(z_i, z'_j)$*.*

$$\begin{aligned}
\mathbb{E}_{z_i,z'_j} K_{\mathrm{MMD}}(z_i, z'_j) &= \int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \mathcal{N}(z_i; x_i, I)\mathcal{N}(z'_j; x_j, I) K_{\mathrm{MMD}}(z_i, z'_j) dz_i dz'_j \\
&= \int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \frac{1}{(2\pi)^d} \exp\left(-\frac{\|z_i - x_i\|^2 + \|z'_j - x_j\|^2}{2} - \frac{\|z_i - z'_j\|^2}{2}\right) dz_i dz'_j.
\end{aligned}$$

We apply a change-of-variable formula:

$$\begin{aligned}
z_i &= -\frac{v_i}{\sqrt{2}} - \frac{v'_j}{\sqrt{6}} + \frac{2}{3}x_i + \frac{1}{3}x_j, \\
z_j &= -\frac{v_i}{\sqrt{2}} + \frac{v'_j}{\sqrt{6}} + \frac{1}{3}x_i + \frac{2}{3}x_j.
\end{aligned}$$

Then,

$$\frac{\|z_i - x_i\|^2 + \|z'_j - x_j\|^2}{2} + \frac{\|z_i - z'_j\|^2}{2} = \frac{1}{2}\left(\|v_i\|^2 + \|v'_j\|^2 + \frac{\|x_i - x_j\|^2}{3}\right).$$

Therefore,

$$\begin{aligned}
&\mathbb{E}_{z_i,z'_j} K_{\mathrm{MMD}}(z_i, z'_j) \\
&= \int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \frac{1}{(2\pi)^d} \exp\left(-\frac{\|v_i\|^2 + \|v'_j\|^2}{2} - \frac{\|x_i - x_j\|^2}{6}\right) \left|\det\left(\frac{\partial(z_i, z'_j)}{\partial(v_i, v'_j)}\right)\right|^d dv_i dv'_j \\
&= 3^{-\frac{d}{2}} \exp\left(-\frac{\|x_i - x_j\|^2}{6}\right).
\end{aligned}$$

Summing up, we have

$$\begin{aligned}
&\mathrm{MMD}^2(\hat{p}', \hat{p}) \\
&= 3^{-\frac{d}{2}} \left(\frac{N'^2}{N^2(N-N')^2} \sum_{i=N'+1}^{N} \sum_{j=N'+1}^{N} - \frac{N+N'}{N^2(N-N')} \sum_{i=1}^{N'} \sum_{j=N'+1}^{N} + \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N'}\right) e^{-\frac{\|x_i - x_j\|^2}{6}}.
\end{aligned}$$

F.4 Experiments on Synthetic Datasets

F.4.1 MoG-8

Setup.

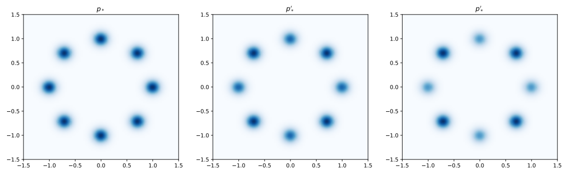
The MoG-8 data distribution is defined as

$$p_*(x) = \frac{1}{8}\sum_{i=1}^{8} \mathscr{N}(x; (\cos\theta_i, \sin\theta_i), 0.1I),$$

where $\theta_i = \frac{2\pi i}{8}$. The modified distribution p'_* with weight λ is defined as

$$p'_*(x) = \frac{1}{4(1+\lambda)}\sum_{i=1}^{8} w_i \mathscr{N}(x; (\cos\theta_i, \sin\theta_i), 0.1I),$$

where $w_i = 1$ for even i and λ for odd i. The construction algorithm for X is randomly sampling a cluster id between 1 and 8 and randomly drawing a sample from the corresponding Gaussian distribution. The construction algorithm for X' is to include a sample $x \in X$ with probability $1-\lambda$ if x is from i-th Gaussian for odd i. The distributions and data with different λ are shown in Fig. F.1.



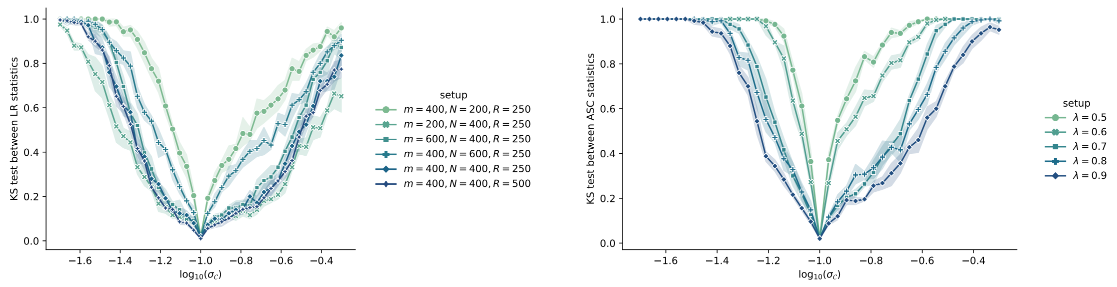
(a) p_* **(b)** $p'_*(\lambda = 0.8)$ **(c)** $p'_*(\lambda = 0.6)$

Figure F.1. Visualization of the experimental setup of MoG-8. (a) Data distribution p_*. (b) - (f) p'_* with different λ values. A larger λ means less data is deleted.

Other hyperparameters are set as follows. The number of training samples $N = 400$ unless specified. The number of samples for the deletion test $m = 400$ unless specified. The number of repeats for each setup is $R = 250$ unless specified. The learning algorithm KDE has bandwidth $\sigma_{\mathscr{A}} = 0.1$ unless specified.

Question 1 (DRE Approximations).

We visualize KS test results for KBC with different bandwidth $\sigma_{\mathscr{C}}$ in Fig. F.2 (extension of Fig. 6.4a). When $\sigma_{\mathscr{C}} \approx \sigma_{\mathscr{A}} = 0.1$, the KS values are small, indicating KBC with these $\sigma_{\mathscr{C}}$ can lead to classifier-based DRE $\hat{\rho}_{\mathscr{E}}$ that is close to $\hat{\rho}$. In terms of statistics, the estimation is most accurate under KL and least accurate under Hellinger distance. In terms of λ, the estimation is more accurate for larger λ, where less data are deleted, as expected.



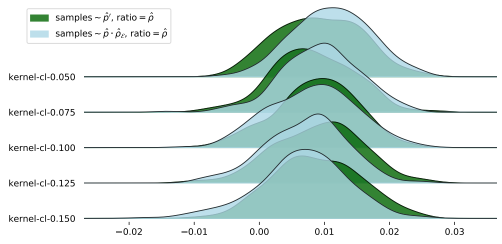
(a) LR statistics with $\lambda = 0.8$ and different m, N, R **(b)** ASC statistics with $\phi(t) = \log(t)$

Figure F.2. KS tests between distributions of statistics for KBC with different $\sigma_{\mathscr{C}}$. (a) $\text{LR}(Y_{\mathscr{H}_0}, \hat{\rho})$ vs $\text{LR}(Y_{\mathscr{H}_0}, \hat{\rho}_{\mathscr{E}})$ with $\lambda = 0.8$ and different m, N, R, complementary to Fig. 6.4a. (b) $\hat{\text{ASC}}_\phi(\hat{Y}, Y_{\mathscr{H}_0}, \hat{\rho})$ vs $\hat{\text{ASC}}_\phi(\hat{Y}, Y_{\mathscr{H}_0}, \hat{\rho}_{\mathscr{E}})$. Smaller values indicate the two compared distributions are closer.

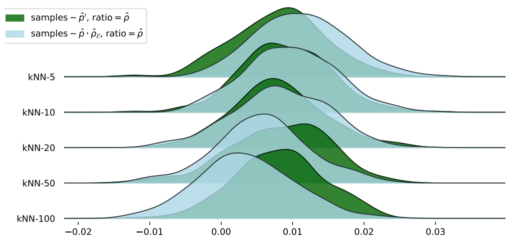
Question 2 (Fast Deletion).

We visualize distributions of LR statistics between $Y_{\mathscr{H}_1}$ and $Y_{\mathscr{D}}$ in Fig. F.3 (extension of Fig. 6.5a). The more overlapping between the distributions, the less distinguishable between the approximated and re-trained models. KBC is generally better than kNN. For kNN a moderate k (e.g. between 10 and 50) has better overlapping.

We visualize KS test results for KBC with different bandwidth $\sigma_{\mathscr{C}}$ in Fig. F.4 (extension of Fig. 6.4b). The KS values are small for a wide range of $\sigma_{\mathscr{C}}$, indicating KBC with these $\sigma_{\mathscr{C}}$ can lead to approximated models indistinguishable from the re-trained model. There is no clear difference between LR and ASC statistics. In terms of λ, the models are less distinguishable when λ is larger, as expected.

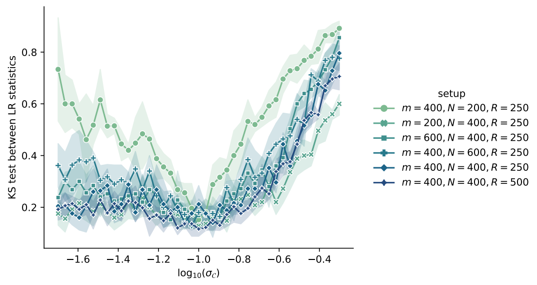


(a) LR for KBC-based DRE ($\lambda = 0.8$)

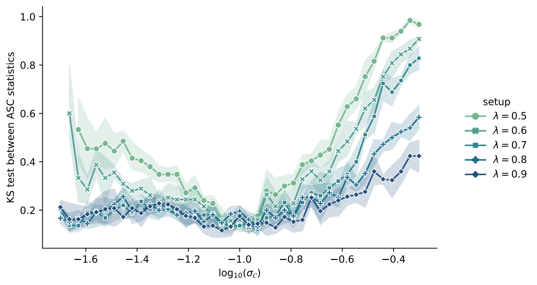


(b) LR for kNN-based DRE ($\lambda = 0.8$)

Figure F.3. Distributions of $\mathrm{LR}(Y_{\mathscr{H}_1}, \hat{\rho})$ vs $\mathrm{LR}(Y_{\mathscr{D}}, \hat{\rho})$.



(a) LR statistics with $\lambda = 0.8$ and different m, N, R



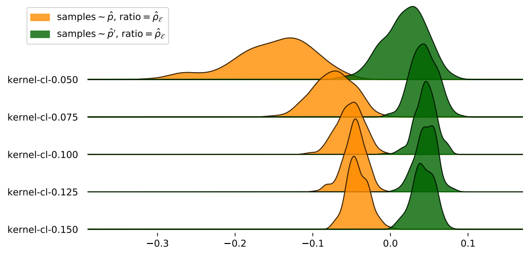
(b) ASC statistics with $\phi(t) = \log(t)$

Figure F.4. KS tests between distributions of statistics for KBC with different $\sigma_\mathscr{C}$. (a) $\mathrm{LR}(Y_{\mathscr{H}_1}, \hat{\rho})$ vs $\mathrm{LR}(Y_{\mathscr{D}}, \hat{\rho})$ with $\lambda = 0.8$ and different m, N, R, complementary to Fig. 6.4b. (b) $\hat{\mathrm{ASC}}_\phi(\hat{Y}, Y_{\mathscr{H}_1}, \hat{\rho})$ vs $\hat{\mathrm{ASC}}_\phi(\hat{Y}, Y_{\mathscr{D}}, \hat{\rho})$. Smaller values indicate the two compared distributions are closer.

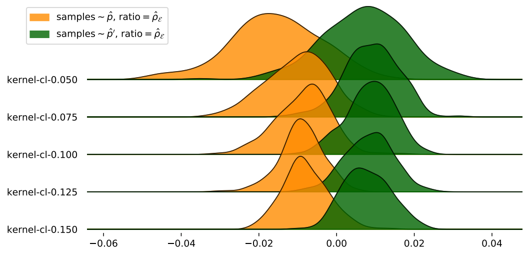
Question 3 (Hypothesis Test).

We visualize distributions of LR statistics between $Y_{\mathscr{H}_0}$ and $Y_{\mathscr{H}_1}$ in Fig. F.5 (extension of Fig. 6.5b). The separation between the distributions indicates how the DRE can distinguish samples between pre-trained and re-trained models. We observe separation for a wide range of classifiers, and KBC is generally comparable to kNN. In terms of λ, larger λ makes the two models less distinguishable.

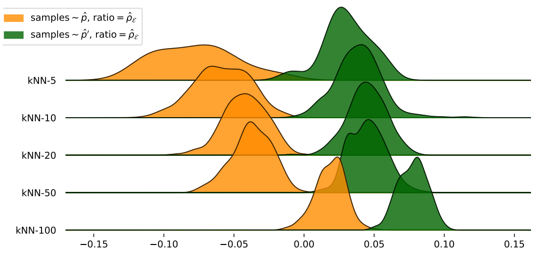
We visualize KS test results for KBC with different bandwidth $\sigma_\mathscr{C}$ in Fig. F.6 (extension of Fig. 6.4c). The KS values are large for a wide range of $\sigma_\mathscr{C}$, indicating KBC with these $\sigma_\mathscr{C}$ can nicely distinguish pre-trained and re-trained model. LR statistics are slightly better than ASC statistics. In terms of λ, the models can be more easily distinguished when λ is small, as expected.



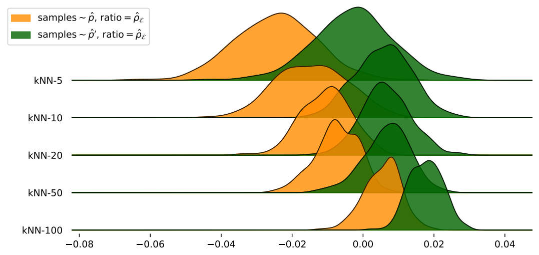
(a) LR for KBC-based DRE ($\lambda = 0.6$)



(b) LR for KBC-based DRE ($\lambda = 0.8$)

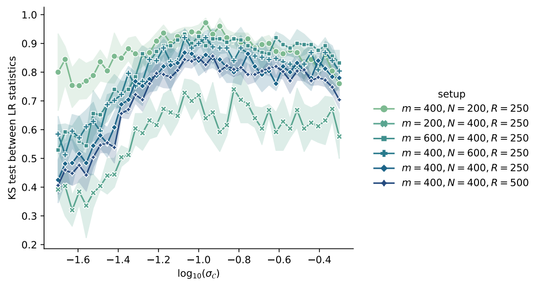


(c) LR for kNN-based DRE ($\lambda = 0.6$)

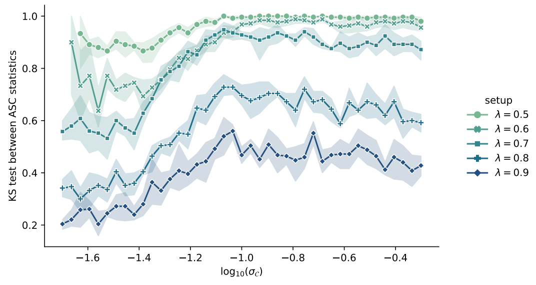


(d) LR for kNN-based DRE ($\lambda = 0.8$)

Figure F.5. Distributions of $\mathrm{LR}(Y_{\mathscr{H}_0}, \hat{\rho})$ vs $\mathrm{LR}(Y_{\mathscr{H}_1}, \hat{\rho})$.



(a) LR statistics with $\lambda = 0.8$ and different m, N, R



(b) ASC statistics with $\phi(t) = \log(t)$

Figure F.6. KS tests between distributions of statistics for KBC with different $\sigma_{\mathscr{C}}$. (a) $\mathrm{LR}(Y_{\mathscr{H}_0}, \hat{\rho})$ vs $\mathrm{LR}(Y_{\mathscr{H}_1}, \hat{\rho})$ with $\lambda = 0.8$ and different m, N, R, complementary to Fig. 6.4c. (b) $\hat{\mathrm{ASC}}_\phi(\hat{Y}, Y_{\mathscr{H}_0}, \hat{\rho})$ vs $\hat{\mathrm{ASC}}_\phi(\hat{Y}, Y_{\mathscr{H}_1}, \hat{\rho})$. Smaller values indicate the two compared distributions are closer.

F.5 Experiments on GAN

F.5.1 Setup

We run experiments on MNIST LeCun et al. [2010] and Fashion-MNIST Xiao et al. [2017]. Both datasets contain gray-scale 28×28 images with 10 labels $\{0, 1, \cdots, 9\}$. We define the even-λ setting as the subset containing all samples with odd labels and a λ fraction of samples with even labels randomly selected from the training set. The rest $1-\lambda$ fraction of samples with even labels form the deletion set X'. We have similar definition for `odd-`λ. In experiments, we let $\lambda \in \{0.6, 0.7, 0.8, 0.9\}$.

The learner is a DCGAN Radford et al. [2015]. For pre-trained and re-trained models, we train each of them for 200 epochs. To obtain DRE, we optimize (6.5), where the network T has the same architecture as the discriminator and is trained for 40 epochs. The learning rate is halved for stability.

All experiments were run on a single machine with one i9-9940X CPU (3.30GHz), one 2080Ti GPU, and 128GB memory.

F.5.2 Results on MNIST

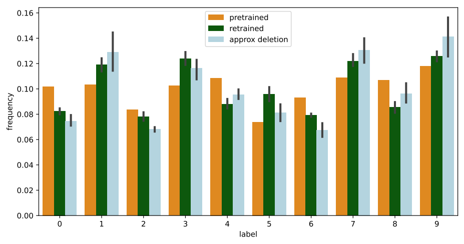
Question 2 (Fast Deletion).

We generate $m = 50\text{K}$ samples from pre-retrained, re-trained, and approximated models (with rejection sampling bound $B = 10$). We then compute the label distributions of these samples based on pre-trained classifiers. [1] Results for each deletion set (including means and standard errors for five random seeds) are shown in Fig. F.7. We find the approximated model generates less (even or odd) labels some data with these labels are deleted from the training set. The variances for deleting odd labels are higher than deleting even labels.

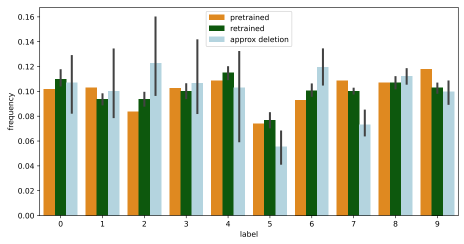
Question 3 (Hypothesis Test).

We generate $m = 1000$ samples for each $Y_{\mathcal{H}_i}$, $i = 1, 2$, and $\hat{Y}$. We visualize distributions of LR and ASC statistics between $Y_{\mathcal{H}_0}$ and $Y_{\mathcal{H}_1}$ in Fig. F.8. The separation between the distributions indicates how the DRE can distinguish samples between pre-trained and re-trained models. The separation for `odd-`λ is better than `even-`λ. In terms of statistics, the LR is slightly better than ASC. In terms of λ, a smaller λ does not lead to more separation.

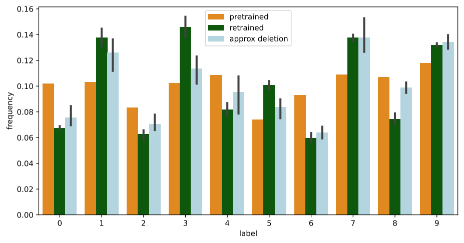
[1]https://github.com/aaron-xichen/pytorch-playground (MIT license)



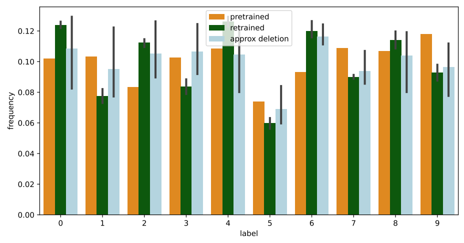
(a) `even-0.8`



(b) `odd-0.8`

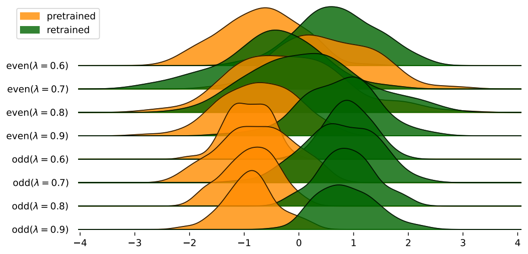


(c) `even-0.6`

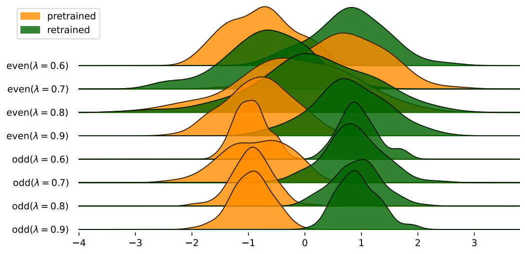


(d) `odd-0.6`

Figure F.7. Label distributions of samples from pre-trained, re-trained, and approximated models. The closeness between green and light blue distributions indicate how well the fast deletion performs.



(a) ASC for VDM-based DRE ($\phi(t) = \log(t)$)



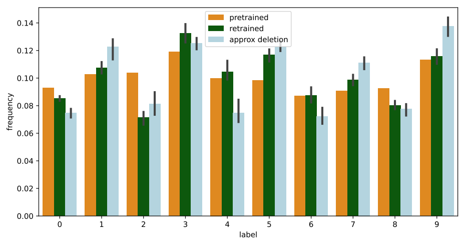
(b) LR for VDM-based DRE

Figure F.8. (a) $\hat{\mathrm{ASC}}_{\phi}(\hat{Y}, Y_{\mathscr{H}_0}, \hat{\rho})$ vs $\hat{\mathrm{ASC}}_{\phi}(\hat{Y}, Y_{\mathscr{H}_1}, \hat{\rho})$. (b) $\mathrm{LR}(Y_{\mathscr{H}_0}, \hat{\rho})$ vs $\mathrm{LR}(Y_{\mathscr{H}_1}, \hat{\rho})$

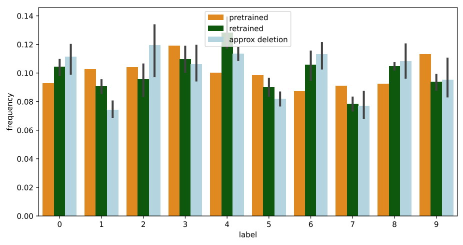
F.5.3 Results on Fashion-MNIST

Question 2 (Fast Deletion).

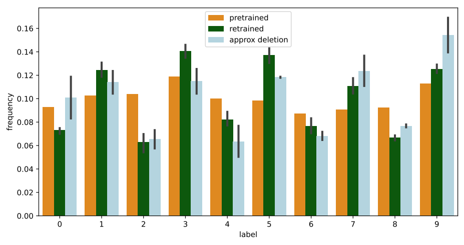
Label distributions for each deletion set (including means and standard errors for five random seeds) are shown in Fig. F.9. Similar to MNIST, we find the approximated model generates less (even or odd) labels some data with these labels are deleted from the training set., and the variances for deleting odd labels are slightly higher than deleting even labels.



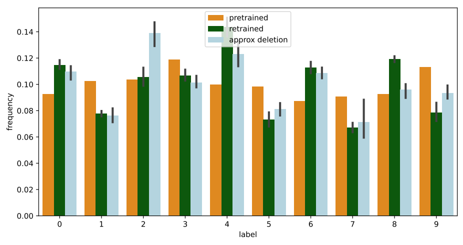
(a) `even-0.8`



(b) `odd-0.8`



(c) `even-0.6`

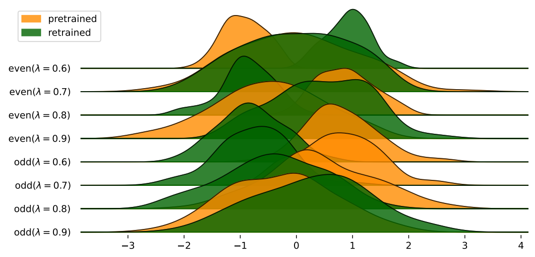


(d) `odd-0.6`

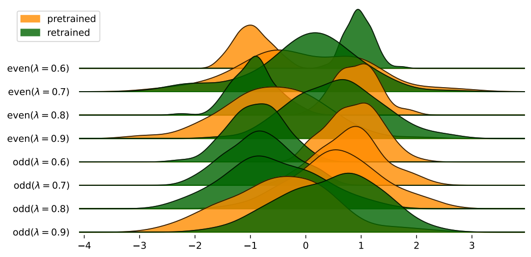
Figure F.9. Label distributions of samples from pre-trained, re-trained, and approximated models. The closeness between green and light blue distributions indicate how well the fast deletion performs.

Question 3 (Hypothesis Test).

We generate $m = 1000$ samples for each $Y_{\mathscr{H}_i}$, $i = 1, 2$, and $\hat{Y}$. We visualize distributions of LR and ASC statistics between $Y_{\mathscr{H}_0}$ and $Y_{\mathscr{H}_1}$ in Fig. F.10. The separation between the distributions indicates how the DRE can distinguish samples between pre-trained and re-trained models. The separation is good for some deletion sets (e.g. $\lambda = 0.6$) while not obvious for others (e.g. $\lambda = 0.9$), indicating performing the deletion test for Fashion-MNIST is harder than MNIST. There is no significant differences between LR and ASC statistics.



(a) ASC for VDM-based DRE ($\phi(t) = \log(t)$)



(b) LR for VDM-based DRE

Figure F.10. (a) $\hat{\mathrm{ASC}}_{\phi}(\hat{Y}, Y_{\mathscr{H}_0}, \hat{\rho})$ vs $\hat{\mathrm{ASC}}_{\phi}(\hat{Y}, Y_{\mathscr{H}_1}, \hat{\rho})$. (b) $\mathrm{LR}(Y_{\mathscr{H}_0}, \hat{\rho})$ vs $\mathrm{LR}(Y_{\mathscr{H}_1}, \hat{\rho})$

Bibliography

Abubakar Abid, Maheen Farooqi, and James Zou. Persistent anti-muslim bias in large language models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 298–306, 2021.

Andrew Aitken, Christian Ledig, Lucas Theis, Jose Caballero, Zehan Wang, and Wenzhe Shi. Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. *arXiv preprint arXiv:1707.02937*, 2017.

David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. *arXiv preprint arXiv:1806.07538*, 2018.

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII*, pages 484–501. Springer, 2020.

Cem Anil, James Lucas, and Roger Grosse. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, pages 291–301, 2019.

Mark Anthony Armstrong. *Basic topology*. Springer Science & Business Media, 2013.

Siddarth Asokan and Chandra Sekhar Seelamantula. Teaching a gan what not to learn. *arXiv preprint arXiv:2010.15639*, 2020.

Bolton Bailey and Matus J Telgarsky. Size-noise tradeoffs in generative networks. In *Advances in Neural Information Processing Systems*, pages 6489–6499, 2018.

Borja Balle, Giovanni Cherubin, and Jamie Hayes. Reconstructing training data with informed adversaries. In *NeurIPS 2021 Workshop Privacy in Machine Learning*, 2021.

Omer Bar-Tal, Dolev Ofri-Amar, Rafail Fridman, Yoni Kasten, and Tali Dekel. Text2live: Text-driven layered image and video editing. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XV*, pages 707–723. Springer, 2022.

Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. Relatif: Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics*, pages 1899–1909. PMLR, 2020.

Samyadeep Basu, Xuchen You, and Soheil Feizi. On second-order group influence functions for black-box predictions. In *International Conference on Machine Learning*, pages 715–724. PMLR, 2020.

David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In *European Conference on Computer Vision*, pages 351–369. Springer, 2020a.

David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. *arXiv preprint arXiv:2005.07727*, 2020b.

Praneeth Bedapudi. Nudenet: Neural nets for nudity detection and censoring, 2022. URL https://github.com/notAI-tech/NudeNet.

Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. *arXiv preprint arXiv:1811.00995*, 2018.

Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582, 2019.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

James Betker. TorToiSe text-to-speech, 4 2022. URL https://github.com/neonbjb/tortoise-tts.

Abeba Birhane, Vinay Uday Prabhu, and Emmanuel Kahembwe. Multimodal datasets: misogyny, pornography, and malignant stereotypes. *arXiv preprint arXiv:2110.01963*, 2021.

Michele Borassi, Alessandro Epasto, Silvio Lattanzi, Sergei Vassilvitskii, and Morteza Zadimoghaddam. Sliding window algorithms for k-clustering problems. *Advances in Neural Information Processing Systems*, 33:8716–8727, 2020.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE*

Symposium on Security and Privacy (SP), pages 141–159. IEEE, 2021.

Olivier Bousquet and André Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

Manuel Brack, Patrick Schramowski, Felix Friedrich, Dominik Hintersdorf, and Kristian Kersting. The stable artist: Steering semantics in diffusion latent space. *arXiv preprint arXiv:2212.06013*, 2022.

Francesco Paolo Cantelli. *Intorno ad un teorema fondamentale della teoria del rischio*. Tip. degli operai, 1910.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480. IEEE, 2015.

John M Chambers. Regression updating. *Journal of the American Statistical Association*, 66 (336):744–748, 1971.

Aaron Chen. Pytorch-playground. https://github.com/aaron-xichen/pytorch-playground, 2020.

Changyou Chen, Chunyuan Li, Liqun Chen, Wenlin Wang, Yunchen Pu, and Lawrence Carin. Continuous-time flows for efficient inference and density estimation. *arXiv preprint arXiv:1709.01179*, 2017.

Hongge Chen, Si Si, Yang Li, Ciprian Chelba, Sanjiv Kumar, Duane Boning, and Cho-Jui Hsieh. Multi-stage influence function. *arXiv preprint arXiv:2007.09081*, 2020.

Ricky TQ Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *arXiv preprint arXiv:1906.02735*, 2019.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *arXiv preprint arXiv:1606.03657*, 2016.

Anton Cherepkov, Andrey Voynov, and Artem Babenko. Navigating the gan parameter space for semantic image editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3671–3680, 2021.

Kristy Choi, Madeline Liao, and Stefano Ermon. Featurized density ratio estimation. In *Uncertainty in Artificial Intelligence*, pages 172–182. PMLR, 2021.

Kristy Choi, Chenlin Meng, Yang Song, and Stefano Ermon. Density ratio estimation via infinitesimal classification. In *International Conference on Artificial Intelligence and Statistics*, pages 2552–2573. PMLR, 2022.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

Jeremy EJ Cohen, Todd Huster, and Ra Cohen. Universal lipschitz approximation in bounded depth neural networks. *arXiv preprint arXiv:1904.04861*, 2019.

R Dennis Cook and Sanford Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.

Imre Csiszár and Paul C Shields. Information theory and statistics: A tutorial. 2004.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Cyprien de Masson d'Autume, Shakir Mohamed, Mihaela Rosca, and Jack Rae. Training language gans from scratch. *Advances in Neural Information Processing Systems*, 32, 2019.

Guillaume Desjardins, Aaron Courville, and Yoshua Bengio. Disentangling factors of variation via generative entangling. *arXiv preprint arXiv:1210.5474*, 2012.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*, 2018.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *arXiv preprint arXiv:1904.01681*, 2019.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

Dan Ellis. Chroma feature analysis and synthesis. *Resources of laboratory for the recognition and organization of speech and audio-LabROSA*, 5, 2007.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *arXiv preprint arXiv:2008.03703*, 2020.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Rohit Gandikota, Joanna Materzynska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models. *arXiv preprint arXiv:2303.07345*, 2023.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462*, 2020.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889, 2015.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR, 2019.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Ryan Giordano, Michael I Jordan, and Tamara Broderick. A higher-order swiss army infinitesimal jackknife. *arXiv preprint arXiv:1907.12116*, 2019a.

Ryan Giordano, William Stephenson, Runjing Liu, Michael Jordan, and Tamara Broderick. A swiss army infinitesimal jackknife. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1139–1147. PMLR, 2019b.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander

Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

Kazuaki Hanawa, Sho Yokoi, Satoshi Hara, and Kentaro Inui. Evaluation of similarity-based explanations. *arXiv preprint arXiv:2006.04528*, 2020.

Boris Hanin. Universal function approximation by deep neural nets with bounded width and relu activations. *arXiv preprint arXiv:1708.02691*, 2017.

Steve Hanneke, Adam Tauman Kalai, Gautam Kamath, and Christos Tzamos. Actively avoiding nonsense in generative models. In *Conference On Learning Theory*, pages 209–227. PMLR, 2018.

Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. Data cleansing for models trained with sgd. *arXiv preprint arXiv:1906.08473*, 2019.

Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. *Advances in Neural Information Processing Systems*, 33:9841–9850, 2020.

Hrayr Harutyunyan, Alessandro Achille, Giovanni Paolini, Orchid Majumder, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Estimating informativeness of samples with smooth unique information. *arXiv preprint arXiv:2101.06640*, 2021.

Leonard Hasenclever, Jakub M. Tomczak, Rianne van den Berg, and Max Welling. Variational inference with orthogonal normalizing flows. In *Workshop on Bayesian Deep Learning (NIPS 2017)*, Long Beach, CA, USA, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016b.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter.

Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. *arXiv preprint arXiv:1902.00275*, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. *arXiv preprint arXiv:1804.00779*, 2018.

Chin-Wei Huang, Laurent Dinh, and Aaron Courville. Solving ode with universal flows: Approximation theory for flow-based models. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Wolfram Research, Inc. Mathematica, Version 13.0.0. URL https://www.wolfram.com/mathematica. Champaign, IL, 2021.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

Keith Ito. The LJ speech dataset. 2017.

Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pages 2008–2016. PMLR, 2021.

Priyank Jaini, Kira A Selby, and Yaoliang Yu. Sum-of-squares polynomial flow. *arXiv preprint* *arXiv:1905.02325*, 2019.

Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. Music type classification by spectral contrast feature. In *Proceedings. IEEE International Conference on Multimedia and Expo*, volume 1, pages 113–116. IEEE, 2002.

Takafumi Kanamori, Taiji Suzuki, and Masashi Sugiyama. f-divergence estimation and two-sample homogeneity test under semiparametric density-ratio models. *IEEE Transactions on Information Theory*, 58(2):708–720, 2011.

Takuhiro Kaneko and Tatsuya Harada. Blur, noise, and compression robust generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13579–13589, 2021.

Takuhiro Kaneko, Hirokazu Kameoka, Kou Tanaka, and Nobukatsu Hojo. Cyclegan-vc2: Improved cyclegan-based non-parallel voice conversion. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6820–6824. IEEE, 2019.

Cemre Karakas, Alara Dirik, Eylul Yalcinkaya, and Pinar Yanardag. Fairstyle: Debiasing stylegan2 with style channel manipulations. *arXiv preprint arXiv:2202.06240*, 2022.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021.

Masahiro Kato and Takeshi Teshima. Non-negative bregman divergence minimization for deep direct density ratio estimation. In *International Conference on Machine Learning*, pages 5320–5333. PMLR, 2021.

Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. *arXiv preprint arXiv:2210.09276*, 2022.

Haidar Khan, Lara Marcuse, and Bülent Yener. Deep density ratio estimation for change point detection. *arXiv preprint arXiv:1905.09876*, 2019.

Rajiv Khanna, Been Kim, Joydeep Ghosh, and Sanmi Koyejo. Interpreting black box predictions using fisher kernels. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3382–3390. PMLR, 2019.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

Polina Kirichenko, Pavel Izmailov, and Andrew G Wilson. Why normalizing flows fail to detect out-of-distribution data. *Advances in neural information processing systems*, 33:20578–20589, 2020.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Frederic Koehler, Viraj Mehta, and Andrej Risteski. Representational aspects of depth and conditioning in normalizing flows. *arXiv preprint arXiv:2010.01155*, 2020.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR, 2017.

Pang Wei Koh, Kai-Siang Ang, Hubert HK Teo, and Percy Liang. On the accuracy of influence functions for measuring group effects. *arXiv preprint arXiv:1905.13289*, 2019.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020.

Zhifeng Kong and Scott Alfeld. Approximate data deletion in generative models. *arXiv preprint arXiv:2206.14439*, 2022.

Zhifeng Kong and Kamalika Chaudhuri. The expressive power of a class of normalizing flow models. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3599–3609, Online, 26–28 Aug 2020. PMLR.

Zhifeng Kong and Kamalika Chaudhuri. Understanding instance-based interpretability of variational auto-encoders. *Advances in Neural Information Processing Systems*, 34, 2021a.

Zhifeng Kong and Kamalika Chaudhuri. Universal approximation of residual flows in maximum mean discrepancy. *arXiv preprint arXiv:2103.05793*, 2021b.

Zhifeng Kong and Kamalika Chaudhuri. Data redaction from conditional generative models. *arXiv preprint arXiv:2305.11351*, 2023a.

Zhifeng Kong and Kamalika Chaudhuri. Data redaction from pre-trained gans. In *First IEEE Conference on Secure and Trustworthy Machine Learning*, 2023b.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=a-xFK8Ymz5J.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Gant Laborde. Deep nn for nsfw detection, 2022. URL https://github.com/GantMan/nsfw_model.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Holden Lee, Rong Ge, Tengyu Ma, Andrej Risteski, and Sanjeev Arora. On the ability of neural nets to express distributions. In *Conference on Learning Theory*, pages 1271–1296, 2017.

Sang-gil Lee, Wei Ping, Boris Ginsburg, Bryan Catanzaro, and Sungroh Yoon. Bigvgan: A universal neural vocoder with large-scale training. In *International Conference on Learning Representations*, 2023.

Qi Li, Long Mai, Michael A Alcorn, and Anh Nguyen. A cost-effective method for improving and re-purposing large, pre-trained gans by fine-tuning their class-embeddings. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

Friedrich Liese and Igor Vajda. On divergences and informations in statistics and information theory. *IEEE Transactions on Information Theory*, 52(10):4394–4412, 2006.

Hongzhou Lin and Stefanie Jegelka. Resnet with one-neuron hidden layers is a universal approximator. In *Advances in Neural Information Processing Systems*, pages 6169–6178, 2018.

Shuang Liu and Kamalika Chaudhuri. The inductive bias of restricted f-gans. *arXiv preprint arXiv:1809.04542*, 2018.

Xuejiao Liu, Yao Xu, and Xueshuang Xiang. Towards gans' approximation ability. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Yulong Lu and Jianfeng Lu. A universal approximation theorem of deep neural networks for expressing distributions. *arXiv preprint arXiv:2004.08867*, 2020.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in neural information processing systems*, pages 6231–6239, 2017.

Helmut Lütkepohl. *Handbook of matrices*, volume 1. Wiley Chichester, 1996.

Shimon Malnick, Shai Avidan, and Ohad Fried. Taming a generative model. *arXiv preprint arXiv:2211.16488*, 2022.

Natalie Maus, Patrick Chao, Eric Wong, and Jacob Gardner. Adversarial prompting for black box foundation models. *arXiv preprint arXiv:2302.04237*, 2023.

Kris McGuffie and Alex Newhouse. The radicalization risks of gpt-3 and advanced neural language models. *arXiv preprint arXiv:2009.06807*, 2020.

Casey Meehan, Kamalika Chaudhuri, and Sanjoy Dasgupta. A non-parametric test to detect data-copying in generative models. *arXiv preprint arXiv:2004.05675*, 2020.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Freeze the discriminator: a simple baseline for fine-tuning gans. *arXiv preprint arXiv:2002.10964*, 2020.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

Saemi Moon, Seunghyuk Cho, and Dongwoo Kim. Feature unlearning for generative models via implicit feedback. *arXiv preprint arXiv:2303.05699*, 2023.

George V Moustakides and Kalliopi Basioti. Training neural networks for likelihood/density ratio estimation. *arXiv preprint arXiv:1911.00405*, 2019.

Bernd Mulansky and Marian Neamtu. Interpolation and approximation from convex sets. *Journal of approximation theory*, 92(1):82–100, 1998.

Angel Muleshkov and Tan Nguyen. Easy proof of the jacobian for the n-dimensional polar coordinates, July 2017.

Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, pages 429–443, 1997.

Johannes Müller. On the space-time expressivity of resnets. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, pages 931–962. PMLR, 2021.

Edward Neuman. Inequalities and bounds for the incomplete gamma function. *Results in Mathematics*, 63(3-4):1209–1214, 2013.

Jerzy Neyman and Egon Sharpe Pearson. Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933.

XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. doi: 10.23915/distill.00003. URL http://distill.pub/2016/deconv-checkerboard.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018. doi: 10.23915/distill.00010. https://distill.pub/2018/building-blocks.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

Dmytro Perekrestenko, Stephan Müller, and Helmut Bölcskei. Constructive universal high-dimensional distribution generation through deep relu networks. *arXiv preprint arXiv:2006.16664*, 2020.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022.

Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*, 2018.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33, 2020.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*, 2022.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2847–2854. JMLR. org, 2017.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Javier Rando, Daniel Paleka, David Lindner, Lennard Heim, and Florian Tramèr. Red-teaming the stable diffusion safety filter. *arXiv preprint arXiv:2210.04610*, 2022.

ITU-T Recommendation. Perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. *Rec. ITU-T P. 862*, 2001.

Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 49–58, 2016.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015a. PMLR. URL https://proceedings.mlr.press/v37/rezende15.html.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015b.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

Benjamin Rhodes, Kai Xu, and Michael U Gutmann. Telescoping density-ratio estimation. *Advances in Neural Information Processing Systems*, 33:4905–4916, 2020.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832 – 837, 1956. doi: 10.1214/aoms/1177728190. URL https://doi.org/10.1214/aoms/1177728190.

Andrew Slavin Ross, Nina Chen, Elisa Zhao Hang, Elena L Glassman, and Finale Doshi-Velez. Evaluating the interpretability of generative models by interactive reconstruction. *arXiv preprint arXiv:2102.01264*, 2021.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Hadi Salman, Payman Yadollahpour, Tom Fletcher, and Kayhan Batmanghelich. Deep diffeomorphic normalizing flows. *arXiv preprint arXiv:1810.03256*, 2018.

Axel Sauer, Tero Karras, Samuli Laine, Andreas Geiger, and Timo Aila. Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis. *arXiv preprint arXiv:2301.09515*, 2023.

Sebastian Schelter. " amnesia"-machine learning models that can forget user data very fast. In *CIDR*, 2020.

Patrick Schramowski, Manuel Brack, Björn Deiseroth, and Kristian Kersting. Safe latent diffusion: Mitigating inappropriate degeneration in diffusion models. *arXiv preprint arXiv:2211.05105*, 2022a.

Patrick Schramowski, Cigdem Turan, Nico Andersen, Constantin A Rothkopf, and Kristian Kersting. Large pre-trained language models contain human-like biases of what is right and wrong to do. *Nature Machine Intelligence*, 4(3):258–268, 2022b.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

Katja Schwarz, Yiyi Liao, and Andreas Geiger. On the frequency bias of generative models. *Advances in Neural Information Processing Systems*, 34, 2021.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34, 2021.

Robert J Serfling. *Approximation theorems of mathematical statistics*, volume 162. John Wiley & Sons, 2009.

Jack Sherman and Winifred J Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1): 124–127, 1950.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

Abhishek Sinha, Kumar Ayush, Jiaming Song, Burak Uzkent, Hongxia Jin, and Stefano Ermon. Negative data augmentation. In *International Conference on Learning Representations*, 2020.

Abhishek Sinha, Kumar Ayush, Jiaming Song, Burak Uzkent, Hongxia Jin, and Stefano Ermon. Negative data augmentation. In *International Conference on Learning Representations*, 2021.

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

Kenji Suzuki, Yoshiyuki Kobayashi, and Takuya Narihira. Data cleansing for deep neural networks with storage-efficient approximation of influence functions. *arXiv preprint arXiv:2103.11807*, 2021.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

Cees H Taal, Richard C Hendriks, Richard Heusdens, and Jesper Jensen. An algorithm for intelligibility prediction of time–frequency weighted noisy speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2125–2136, 2011.

Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

Esteban G Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

Ahmed Taha, Abhinav Shrivastava, and Larry S Davis. Knowledge evolution in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12843–12852, 2021.

Shuhan Tan, Yujun Shen, and Bolei Zhou. Improving the fairness of deep generative models without retraining. *arXiv preprint arXiv:2012.04842*, 2020.

Ugo Tanielian, Maxime Sangnier, and Gerard Biau. Approximating lipschitz continuous functions with groupsort neural networks. *arXiv preprint arXiv:2006.05254*, 2020.

Matus Telgarsky. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*, 2015.

Naoyuki Terashita, Hiroki Ohashi, Yuichi Nonaka, and Takashi Kanemaru. Influence estimation for generative adversarial networks. *arXiv preprint arXiv:2101.08367*, 2021.

Takeshi Teshima, Isao Ishikawa, Koichi Tojo, Kenta Oono, Masahiro Ikeda, and Masashi Sugiyama. Coupling-based invertible neural networks are universal diffeomorphism approximators. *Advances in Neural Information Processing Systems*, 33, 2020.

Hoang Thanh-Tung and Truyen Tran. Catastrophic forgetting and mode collapse in gans. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2020.

Nathan Thiem, Marko Orescanin, and James Bret Michael. Reducing artifacts in gan audio synthesis. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1268–1275. IEEE, 2020.

Daniel Ting and Eric Brochu. Optimal subsampling with influence functions. In *Advances in neural information processing systems*, pages 3650–3659, 2018.

Jakub M Tomczak and Max Welling. Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630*, 2016.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018.

Enayat Ullah, Tung Mai, Anup Rao, Ryan A Rossi, and Raman Arora. Machine unlearning via algorithmic stability. In *Conference on Learning Theory*, pages 4126–4142. PMLR, 2021.

Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1): 7184–7220, 2016.

Dani Valevski, Matan Kalman, Yossi Matias, and Yaniv Leviathan. Unitune: Text-driven image editing by fine tuning an image generation model on a single image. *arXiv preprint arXiv:2210.09477*, 2022.

Rianne Van Den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 393–402. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

Gerrit JJ van den Burg and Christopher KI Williams. On memorization in probabilistic deep generative models. *NeurIPS*, 2021.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*, 2019.

Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*, 2023.

Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot... for now. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8695–8704, 2020.

David Warde-Farley and Ian Goodfellow. 11 adversarial perturbations of deep neural networks. *Perturbations, Optimization, and Statistics*, 311:5, 2016.

Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. Technical Report CNS-TR-201, Caltech, 2010. URL http://www.vision.caltech.edu/visipedia/CUB-200.html.

Dominik Wied and Rafael Weißbach. Consistency of the kernel density estimator: a survey. *Statistical Papers*, 53(1):1–21, 2012.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Min Xu, Ling-Yu Duan, Jianfei Cai, Liang-Tien Chia, Changsheng Xu, and Qi Tian. Hmm-based audio keyword generation. In *Advances in Multimedia Information Processing-PCM 2004: 5th Pacific Rim Conference on Multimedia, Tokyo, Japan, November 30-December 3, 2004. Proceedings, Part III 5*, pages 566–574. Springer, 2005.

Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1316–1324, 2018.

Makoto Yamada, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Masashi Sugiyama. Relative density-ratio estimation for robust distribution comparison. *Advances in neural information processing systems*, 24, 2011.

Haotian Ye, Chuanlong Xie, Yue Liu, and Zhenguo Li. Out-of-distribution generalization analysis via influence function. *arXiv preprint arXiv:2101.08521*, 2021.

Chih-Kuan Yeh, Joon Sik Kim, Ian EH Yen, and Pradeep Ravikumar. Representer point selection for explaining deep neural networks. *arXiv preprint arXiv:1811.09720*, 2018.

Jinsung Yoon, Sercan Arik, and Tomas Pfister. Data valuation using reinforcement learning. In *International Conference on Machine Learning*, pages 10842–10851. PMLR, 2020.

Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. Libritts: A corpus derived from librispeech for text-to-speech. *arXiv preprint arXiv:1904.02882*, 2019.

Han Zhang, Xi Gao, Jacob Unterman, and Tom Arodz. Approximation capabilities of neural ordinary differential equations. *arXiv preprint arXiv:1907.12998*, 2019a.

Junzhe Zhang, Xinyi Chen, Zhongang Cai, Liang Pan, Haiyu Zhao, Shuai Yi, Chai Kiat Yeo, Bo Dai, and Chen Change Loy. Unsupervised 3d shape completion through gan inversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1768–1777, 2021.

Linfeng Zhang and Lei Wang. Monge-ampere flow for generative modeling. *arXiv preprint arXiv:1809.10188*, 2018.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

Xu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in gan fake images. In *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2019b.

Ziqiang Zhang, Long Zhou, Chengyi Wang, Sanyuan Chen, Yu Wu, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. Speak foreign languages with your own voice: Cross-lingual neural codec language modeling. *arXiv preprint arXiv:2303.03926*, 2023.

Miaoyun Zhao, Yulai Cong, and Lawrence Carin. On leveraging pretrained gans for generation with limited data. In *International Conference on Machine Learning*, pages 11340–11351. PMLR, 2020.

Hattie Zhou, Ankit Vani, Hugo Larochelle, and Aaron Courville. Fortuitous forgetting in connectionist networks. In *International Conference on Learning Representations*, 2021.

Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *European conference on computer vision*, pages 592–608. Springer, 2020.

Minfeng Zhu, Pingbo Pan, Wei Chen, and Yi Yang. Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5802–5810, 2019.