Guiding Deep Probabilistic Models

by

Timur Garipov

B.S., Lomonosov Moscow State University (2017)
M.S., Lomonosov Moscow State University (2019)

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2024

Authored By:	Timur Garipov Department of Electrical Engineering and Computer Science July 15, 2024
Certified By:	Tommi S. Jaakkola Professor of Electrical Engineering and Computer Science Thesis Supervisor
Accepted By:	Leslie A. Kolodziejski Professor of Electrical Engineering and Computer Science Chair, Department Committee on Graduate Students

Guiding Deep Probabilistic Models

by

Timur Garipov

Submitted to the Department of Electrical Engineering and Computer Science
on July 15, 2024, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Deep probabilistic models utilize deep neural networks to learn probability distributions in high-dimensional data spaces. Learning and inference in these models are complicated due to the difficulty of direct evaluation of the differences between the model distribution and the target. This thesis addresses this challenge and develops novel algorithms for learning and inference based on the guidance of complex parameterized distributions towards desired configurations via signals from auxiliary discriminative models.

In the first part of the thesis, we develop novel stable training objectives for Generative Adversarial Networks (GANs). We show that under standard unary-discriminator objectives, most of the valid solutions, where the learned distribution is aligned with the target, are unstable. We propose training objectives based on pairwise discriminators that provably preserve distribution alignment and demonstrate improved training stability in image generation tasks.

In the second part of the thesis, we introduce distribution support alignment as an alternative to the distribution alignment objective and develop a learning algorithm that guides distributions towards support alignment. We demonstrate the effectiveness of our approach in unsupervised domain adaptation under label distribution shift. Recent works have shown that under cross-domain label distribution shift, optimizing for distribution alignment is excessively restrictive and causes performance degradation. Our algorithm, which is based on support alignment, alleviates this issue.

In the third part of the thesis, we develop a novel approach to compositional generation in iterative generative processes: diffusion models and Generative Flow Networks (GFlowNets). Motivated by the growing prominence of generative models pre-trained at scale and the high training costs, we propose composition operations and guidance-based sampling algorithms that enable the combination of multiple pre-trained iterative generative processes. We offer empirical results on image and molecular generation tasks.

Acknowledgments

I am deeply grateful to many people in my life, far more than I can list here.

I am very fortunate to have received my PhD training in Tommi Jaakkola's research group at MIT. I am grateful to Tommi for his insight, guidance, unconditional support, and encouragement to seek a broader perspective and think creatively. I sincerely appreciate the chance to pursue my interests while being reassured by Tommi's belief in me, even when I am facing doubts or challenges in my research.

I am profoundly grateful l to my undergraduate research advisor, Dmitry Vetrov. Dmitry taught me the fundamentals of probabilistic machine learning, and his enthusiasm got me excited about machine learning research. I made my first steps as a researcher working in the Bayesian Methods Research Group, supervised by Dmitry. I am also very grateful to Andrew Gordon Wilson, whose collaboration and support during my Master's program led to the publication of my first conference papers.

I was incredibly lucky to work with many amazing colleagues during my PhD. I would like to especially thank Shangyuan Tong, Ge Yang, Sebastiaan De Peuter, Abhi Gupta, Vikas Garg, Samuel Kaski, Yang Zhang, Shiyu Chang, Semyon Savkin, and Egor Lifar, who have been amazing collaborators. I learned a lot from each of you. I am also very grateful to my industry internship mentors and friends, who made a huge impact on my research interests and career: Chiyuan Zhang, Mike Mozer, Ekin Dogus Cubuk, David Hayden, Zhao Chen, and Yuning Chai. I am very grateful to my thesis committee members, Samuel Kaski and Phillip Isola, for their invaluable guidance and insightful feedback that helped shape my work. I want to thank Russ Tedrake for the amazing robotics classes that were enlightening and inspiring and fueled my interest in the field. I also want to thank my friend Richard Li for being a fantastic collaborator on our class projects and for memorable research discussions.

To my wonderful wife, Tatiana, your boundless love and support have made this journey possible. Your faith in me, even during the toughest times, has been my guiding light. I am eternally grateful for your presence in my life and for sharing this journey with me.

I would like to express my deepest gratitude to my parents, Ismagil and Galina, for their unwavering support, love, and encouragement throughout my studies. Your sacrifices and belief in me have been the foundation of my success. I am forever grateful. I am sincerely thankful to my brothers Emil and Roman for their consistent support and encouragement. Your companionship and belief in me have played a significant role in helping me achieve this milestone. With deep appreciation, I thank my grandparents for instilling in me the love for learning and perseverance.

I want to especially thank my closest friends, Pavel Izmailov, Dmitrii Podoprikhin, and Evgenii Nikishin, who have also been my collaborators since our undergraduate days and played a major role in forming my research interests. Your encouragement to pursue a PhD and constant support throughout this journey have been instrumental in my success. I am truly fortunate to have you by my side.

I am also grateful to my cousin Dina Garipova and my friends Pavel Kovalenko, Victoria Snorovikhina, Tamara Makoveeva, Eleonora Kiziv, Olga Tsvetkova, Alexei Fedoret, Anastasia Sobolina, Sergey Kolchenko, Anna Talyzina, Alina Ringaci, Evgenii Kegeles, Nicole Bens, Iryna Prekrasna, Anatoly Prekrasnyy, Artur Avkhadiev, Eldar Shakirov, Victoria Shakirova, Daniil Klyuev, George Stepaniants, Alexander Avdoshkin, David Saykin, Samat Davletshin, Gerardo Flores, Zoe De Simone, Benson Chen, Xiang Fu, Yilun Xu, Abhi Gupta, Ge Yang, Hannes Stärk, Felix Faltings, Jason Yim, Peter Holderrieth, Amit Schechter, Chenyu Wang, and many others. I cannot imagine completing my PhD without your support. I am sincerely grateful to Trevor Stricker for our guitar sessions and heavy metal jams, and to the MIT Heavy Metal 101 IAP class for connecting heavy metal fans. I want to thank the EECS graduate office, graduate officer Leslie Kolodziejski, and Jaakkola lab's administrative assistant, Teresa Cataldo, for their exceptional support and dedication to student success.

Finally, I am thankful to so many of my teachers, especially Dmitry Kropotov, Elena Andreeva, Andrei Shestimerov, Sergey Mikhailin, Kamil Hadiev, Ravil Hadiev, Liudmila Bamburkina, Alexei Zabolotskiy, Natalia Zadorina, and teachers at Russian IOI Camp and Summer Informatics School.

Epigraph

Heaven and Hell

Sing me a song, you're a singer
Do me a wrong, you're a bringer of evil
The Devil is never a maker
The less that you give, you're a taker
So it's on and on and on,
it's Heaven and Hell, oh well

The lover of life's not a sinner
The ending is just a beginner
The closer you get to the meaning
The sooner you'll know that you're dreaming
So it's on and on and on,
oh it's on and on and on
It goes on and on and on, Heaven and Hell
I can tell, fool, fool!

Well if it seems to be real, it's illusion
For every moment of truth,
there's confusion in life
Love can be seen as the answer,
but nobody bleeds for the dancer
And it's on and on, on and on and on ...

They say that life's a carousel
Spinning fast, you've got to ride it well
The world is full of Kings and Queens
Who blind your eyes and steal your dreams
It's Heaven and Hell, oh well
And they'll tell you black is really white
The moon is just the sun at night
And when you walk in golden halls
You get to keep the gold that falls
It's Heaven and Hell, oh no!
Fool, fool!
You've got to bleed for the dancer!
Fool, fool!
Look for the answer!
Fool, fool, fool!

Lyrics of "Heaven and Hell" from the album
"Heaven and Hell" (1980) by Black Sabbath.
Lyrics by Ronnie James Dio (1942 – 2010).

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Artificial intelligence systems based on deep probabilistic models [174, 175] have powered advances across applied and scientific domains, including image synthesis [21], text generation [184], robotic manipulation [41], and drug discovery [257]. The ability of deep probabilistic models to capture complex statistical dependencies in high-dimensional and structured spaces and generate realistic samples has made them indispensable tools in modern machine learning.

Traditional training and inference tasks in machine learning involve well-defined objectives and direct evaluation metrics, such as optimization with explicitly specified training signals on large datasets typically collected, labeled, or produced by people. These tasks typically benefit from clear goals and a relatively predictable training process. In contrast, frontier problems — such as cross-domain adaptation, multi-objective design, generation of complex structured objects, solving multistep reasoning tasks, generation of creative content, human-computer interaction, and scientific innovation — present significant challenges. Direct evaluation of model outputs in these contexts is often costly or impossible due to the complex and dynamic nature of the tasks. Both model training and evaluation with explicitly specified signals have limited scaling potential.

These challenges motivate a modular approach that introduces learnable helper models that are trained with auxiliary objectives and then used to guide the primary (possibly large and pre-trained) model. Automatically learned signals can help guide the learning process in problems where direct supervision signals are scarce or expensive, facilitate adaption to drifting data distributions, manage trade-offs between multiple objectives, steer multi-step inference processes, and provide a basis for model evaluation.

A prominent trend in recent years is the use of large-scale models pre-trained on extensive datasets [27]. Models such as GPTs [33, 184, 202], CLIP [203], DINO [185], and Stable Diffusion [68] are trained on massive amounts of data and discover broadly generalizable representations. These models can then be fine-tuned or adapted for downstream tasks, leveraging the knowledge distilled in the model to achieve task-specific goals. This approach of large-scale pre-training followed by adaptation has demonstrated immense capacity to improve the performance and efficiency of machine learning systems, as it reduces the need for task-specific training from scratch and facilitates rapid deployment.

As progress on many problems in engineering and science benefits from multidisciplinary collaboration, complex artificial intelligence systems will inevitably involve cooperation among multiple models with different capabilities and expertise. Building such model consortia necessitates learning helper models that serve as interfaces for effective interaction and coordination among the models.

Development of structured models involving multiple components and governed by automatically learned signals introduces a range of methodological challenges. For example,

- When one model relies on feedback from another model and the states of both models are evolving, it is crucial to ensure stable interaction throughout the training. Training dynamics instability hinders convergence and reduces the effectiveness of the learning process. Understanding and managing the interaction between the primary model and the feedback model is necessary to ensure reliable and consistent training outcomes.

- The nature of a training signal and underlying assumptions have a significant effect on the obtained solution. In particular, directing models for adaptation to new domains under multi-faceted distribution shifts and imbalances requires taking into account the properties of the training signal and associated optimal solutions. Developing new measures and learning signals that can direct models effectively under these conditions is essential. These measures must be flexible enough to ensure that models can generalize across different domains and maintain high performance.

- Large pre-trained generative models are the results of extensive training on vast datasets and, therefore, require novel techniques for effective adaptation to new tasks. Combination of multiple models is a powerful approach for model reuse and adaptation. Extending model capacity to represent composite probability distributions requires the development of informative signals for the coordination of multiple generative processes.

This thesis addresses the outlined challenges and develops novel methods for training and inference in deep probabilistic models. At the core of these methods are improved techniques for guiding complex parameterized distributions towards desired configurations via signals from auxiliary models. Our contributions are focused on method development, are agnostic to neural network architectures, and can be applied to a wide range of application domains.

This thesis is organized as follows. Section 1.1 summarizes the motivation, the research questions, and the contributions of the thesis on a technical level. Chapter 2 covers background on deep probabilistic models. Chapters 3, 4, and 5 present the contributions of the thesis. Chapter 6 provides a discussion of the contributions and their implications, and concludes the thesis.

1.1 Motivation, Research Questions, Contributions

Despite advances in deep probabilistic models (see Chapter 2), learning and inference with probabilistic models in high-dimensional spaces remain highly complicated. Several specific challenges underpin these complexities:

- **Complexity of Specifying Optimality Criteria.** As learning problems become more complex, specifying the notion of optimality and training objectives excplitly becomes increasingly impractical. Explicitly specified training signals are often inadequate or excessively restrictive for capturing the nuanced requirements of complex models. Greater flexibility in optimization criteria is required to adapt to the specific needs of the model and the task. As learning high-dimensional probability distributions is an inherently ill-defined problem, extending the training criteria beyond existing objectives is essential for exploring inductive biases and implicit training dynamics regularization strategies for learning complex probability distributions.

- **Infeasibility of Direct Likelihood Evaluation and Divergence Computation.** In many models, the evaluation of the likelihood for maximum-likelihood estimation or divergence estimation is challenging or infeasible. Consequently, the distance to the target distribution cannot be directly assessed, necessitating alternative approaches.

  Objectives such as optimal transport distances measure the differences between distributions at the population level. These population-level objectives require significant computational resources and are not well-suited for deep neural networks that rely on stochastic estimators to enable more efficient algorithms for large networks and large datasets in high-dimensional spaces.

- **Insufficiently Informative Objectives from Distance-Based and Likelihood-Based Signals.** Many training objectives are based on model likelihood or distances between data points. In high-dimensional spaces, such objectives often lack informativeness and fail to capture the true structure of the data distribution. Thus, they do not provide strong learning signals. This insufficiency complicates training dynamics and hinders the effectiveness of the learning process.

- **Realization of Complex Distributions in Controlled Generation Tasks.** In many inference and controlled generation tasks, even if a user can explicitly describe the properties of the desired distribution, directly realizing such distributions is often not possible since DPM families impose specific structure on the generative process. It is challenging to prescribe control mechanisms and sampling schemes to realize controllable inference in DPMs.

Guiding Deep Probabilistic Models. Motivated by the challenges listed above, this thesis focuses on using auxiliary models trained on separate discriminative tasks to guide deep probabilistic models during training and inference. The key idea of the approach is to learn informative signals that can be used to guide a deep probabilistic model towards the target for training or inference. This guidance approach builds upon and expands the idea of Generative Adversarial Networks (GANs) [78]. Guidance introduces several significant benefits:

- **Flexibility in Specifying Optimality Criteria.** The use of learned objectives based on signals from auxiliary models provides enhanced flexibility. The optimization criteria can be dynamically adapted to the current state of the deep probabilistic model. This flexibility expands the range of optimization criteria beyond explicitly specified objectives. Moreover, guidance with an auxiliary model introduces additional inductive biases that influence model capacity and training dynamics.

- **Informative Signals in High-Dimensional Spaces.** Through training on samples from the distributions, auxiliary models automatically discover informative representations of high-dimensional data. The signals derived from these models can be used as learned metrics on the data space and as proxies for divergences between the model and the target distribution configuration. This approach enables models to be trained and evaluated even when direct estimates of divergences are not tractable or are not informative.

  Direct computation of population-level objectives, such as optimal transport, is computationally expensive. In certain scenarios, estimation of distribution differences via auxiliary models provides a practical workaround by offering signals that approximate these objectives with adequate compute costs. This approach aligns well with the stochastic optimization algorithms for deep neural networks and makes it feasible to train complex models on large datasets.

- **Guidance for Controlled Generation.** One can use auxiliary models to construct distributions with desired properties and guide the inference toward these targets. This approach expands the tools for model composition and control to scenarios where the direct realization of the target distribution is not feasible.

Guidance using auxiliary discriminative models offers a powerful and flexible framework for addressing the challenges inherent in training and inference with DPMs. Leveraging the power of discriminative models to learn informative representations and metrics, provide informative signals, and introduce beneficial inductive biases, this approach enhances the capability of DPMs to learn and represent complex distributions. This thesis explores the guidance methodology through three research questions, each addressing specific aspects of adversarial training, joint training dynamics of the main and auxiliary models, properties of training criteria and objectives, and guidance for inference in structured models. The thesis is focused on principled methods with guarantees on the optimality of target configurations, stability of solutions, and sampling from the target distributions. The contributions span theoretical analysis, novel practical methods, and empirical evaluations, highlighting the efficacy of using auxiliary discriminative models to guide deep probabilistic models towards target configurations.

Research Questions 1. Adversarial training methods typically align distributions by solving two-player games between a generator and a discriminator. However, in most current formulations, even if the generator aligns perfectly with data, a sub-optimal discriminator can still drive the two apart. Absent additional regularization, the instability can manifest itself as a never-ending game. **How can we guide a parameterized generative distribution towards the target so that the alignment, once reached, is preserved?**

Contributions. In Chapter 3, we introduce a family of objectives by leveraging pairwise discriminators, and show that only the generator needs to converge. The alignment, if achieved, would be preserved with any discriminator. We provide sufficient conditions for local convergence; characterize the capacity balance between parametric generators and discriminators; and construct examples of minimally sufficient discriminators. Empirically, we illustrate the theory and the effectiveness of our approach on synthetic examples. Moreover, we show that practical methods derived from our approach improve training stability in image generation tasks.

Research Question 2. Adversarial distribution alignment methods found applications in domain adaption [72]. The adversarial process aims to make the learned representations indistinguishable between domains, effectively aligning their distributions. As a result, the shared classifier can perform well on both the source and target domains, improving the model's ability to generalize across different data distributions. Recent works [143, 239, 278] have shown that optimizing for distribution alignment can be excessively restrictive under cross-domain label distribution shifts. **Can we develop an alternative notion of alignment that partially lifts distribution alignment restrictions while enabling effective learning in unsupervised domain adaptation scenarios? How do we guide distributions towards this relaxed alignment configuration?**

Contributions. In Chapter 4, we study the problem of aligning the supports of distributions. Compared to the existing work on distribution alignment, support alignment does not require the densities to be matched. We propose symmetric support difference as a divergence measure to quantify the mismatch between supports. We show that select discriminators (e.g. discriminator trained for Jensen–Shannon divergence) are able to map support differences as support differences in their one-dimensional output space. Following this result, our method aligns supports by minimizing a symmetrized relaxed optimal transport cost in the discriminator 1D output space via a non-zero-sum game process. We show that our approach can be viewed as a limit of existing notions of alignment based on optimal transport by increasing transportation assignment tolerance. We quantitatively evaluate the method across domain adaptation tasks with shifts in label distributions. Our experiments show that the proposed method is more robust against these shifts compared to other alignment-based methods.

Research Question 3. Recently, significant advances in generative modeling have been achieved by models based on iterative processes. Models such as diffusion models [226, 231] and GFlowNets [16], learn a deep policy that guides a long chain of data refinement updates. High training costs of generative models and the need to fine-tune them for specific tasks have created a strong interest in model reuse and composition. A key challenge in composing iterative generative processes, such as GFlowNets and diffusion models, is that to realize the desired target distribution, all steps of the generative process need to be coordinated, and satisfy delicate balance conditions. **How can we guide multiple iterative generation processes towards realizing composite distributions? How can we control contributions of individual models?**

Contributions. In Chapter 5, we propose Compositional Sculpting: a general approach for defining compositions of iterative generative processes. We introduce a method for sampling from these compositions built on classifier guidance. We showcase ways to accomplish compositional sculpting in both GFlowNets and diffusion models. We highlight two binary operations — the *harmonic mean* ($p_1 \otimes p_2$) and the *contrast* ($p_1 \RIGHTcircle p_2$) between pairs, and the generalization of these operations to multiple component distributions. We offer empirical results on image and molecular generation tasks.

Chapter 2

Background: Deep Probabilistic Models

2.1 Probabilistic Machine Learning

The goal of machine learning as an area of computer science is to build algorithms that can learn, i.e., acquire capabilities to make inferences (predictions) about the world, from experience or observed data. The fundamental purpose of machine learning models is to enable informed decision-making, with their outputs serving as inputs for human-driven decisions or as direct decision-making mechanisms for artificial intelligence systems engaged in real-world tasks. Real-world data and predictions about the world are inherently noisy and involve irreducible uncertainty. Probability theory [128] offers a principled and rigorous framework for quantifying uncertainty.

Learning involves constructing models that build representations of objects in the world and capture the relations between different observable quantities related to those objects. Based on the probabilistic principles, probabilistic machine learning models [24, 174, 175] aim to learn statistical dependencies between observed variables from data. Statistical dependencies refer to the relationships or associations between variables where the distribution of one variable is related to the value(s) of other variable(s). For instance, one might want to estimate the conditional probability $p_{Y|X}(y|x)$ of Y given X, encoding the likelihood of observing different values y of the variable Y given that the variable X is observed to have the value x. Alternatively, instead of focusing on the one-way dependency of Y on X, one might want to characterize the mutual interdependency between X and Y by estimating the joint distribution $p_{X,Y}(x, y)$, which encodes the likelihood of observing X and Y taking the respective values x and y simultaneously. The joint distribution completely characterizes the statistical relation between variables X and Y. The conditional probabilities $p_{X|Y}$ and $p_{Y|X}$, which represent one-way dependencies, can be derived from the joint distribution:

$$p_{Y|X}(y|x) = \frac{p_{X,Y}(x,y)}{p_X(x)}, \quad p_{X|Y}(x|y) = \frac{p_{X,Y}(x,y)}{p_Y(y)}, \tag{2.1}$$

$$p_X(x) = \int p_{X,Y}(x,y)\,dy, \quad p_Y(y) = \int p_{X,Y}(x,y)\,dx. \tag{2.2}$$

Note that the marginal distributions $p_X(x)$ and $p_Y(y)$, that encode the likelihoods of observing X taking the value x in isolation and respectively Y taking the value of y in isolation, can also be derived from the joint distribution.

Continuing with the abstract example of two variable X and Y, we now describe the tasks of probabilistic machine learning: learning statistical dependencies from data and using learned models to produce inferences about the variables.

2.1.1 Training (Learning from Data)

Suppose that we can collect a number of observations $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$ of variables X and Y. We assume that these real-world observations constitute independent and identically distributed (i.i.d.) samples from some underlying data-generating distribution $p^D_{X,Y}(x, y)$. In general, this distribution might be unknown and we only observe a finite number n of samples from this distribution.

The goal of training in probabilistic machine learning is to learn a model distribution p^M so that the learned model

- explains the observed data or does not contradict the observed data;
- approximates the true data generating distribution $p^M \approx p^D$ and approaches p^D

as n grows ($n \to \infty$).

There are many ways to approach this task, and the kind of model to be learned must be specified. It is common to distinguish between "discriminative" and "generative" models [179]. In the example of two variables X and Y (which in this case abstracts away some nuance, but provides general intuition) the difference between discriminative and generative models can be broadly understood as follows

- **Discriminative Models** learn the conditional probability $p^{\text{model}}_{Y|X}(y|x)$, and thus can be used only to evaluate the one-directional effect of X on the distribution of Y;

- **Generative Models** learn the joint probability distribution $p^{\text{model}}_{X,Y}(x, y)$ and thus aim to completely characterize statistical dependencies between X and Y. As discussed above, the distributions $p^{\text{model}}_{Y|X}$, $p^{\text{model}}_{X|Y}$, p^{model}_{X}, p^{model}_{Y} can be derived from the model of the joint distribution.

2.1.2 Inference

Once a probabilistic model is trained, it can be used to make statistical predictions about the world. The goal of the inference task is to leverage the knowledge distilled in the model during training to address various types of questions about the variables of interest. Inference tasks include:

- **Prediction.** Prediction focuses on estimating the value of a variable. For example, if the model $p^M_{Y|X}(y|x)$ is learned, one might want to know the most likely value of Y given a new observation $X = x'$. This involves finding $\hat{y} = \text{argmax}_y \ p^M_{Y|X}(y|x')$. More generally, one might compute any statistics derived from $p^M_{Y|X}(y|x')$ (such as mean, median, modes, . . .).

- **Uncertainty Quantification and Decision Making.** Having access to the full probability distribution rather than focusing on a single prediction allows one to quantify the uncertainty associated with the prediction. This can involve computing confidence intervals for predictions or evaluating the entropy of the predictive distribution as a measure of the model's uncertainty. In some applications, the goal is to make decisions based on the inferred probabilities. This involves using the model's predictions and uncertainty estimates to choose actions that minimize a certain risk function.

- **Probaility (Density) Evaluation.** This task involves computing the probability or probability density of data points under the learned model: given a model $p^M_{X,Y}$ and a query point (x', y'), evaluating the likelihood $p^M_{X,Y}(x', y')$ of the observing (x', y') under the learned model. The probability (density) values can be used to rank data by likelihood, find anomalies, and estimate distribution properties. Moreover, other inference tasks often involve (unnormalized) likelihood evaluation as a subroutine.

- **Sample Generation (Sampling).** Sample generation involves creating new data points that are consistent with the learned model. For generative models that learn the joint distribution $p^M_{X,Y}$, this task involves sampling new pairs (x, y) from the model distribution. Samples might be the main focus of inference (e.g. in image generation) or might be used to achieve other goals, such as computing Monte Carlo estimates.

- **Conditional / Controllable Generation.** Conditional generation focuses on generating new data samples given information about observed quantities. An example is generation of Y given $X = x'$ using the learned conditional distribution $p^M_{Y|X}$. Controllable generation extends this task by allowing control over certain aspects of the generated data, based on user-defined conditions or constraints.

- **Use in Structured Probabilistic Models.** The learned distribution can be used as a component in more complex structured probabilistic models, such as graphical models [127] or compositional models (e.g, [60, 97], and models described in Chapter 5).

2.1.3 A Note on Probabilistic and Causal Models

So far, the discussion has focused on statistical dependencies. In order to make decisions and plan in the real world, it is often important to infer causal dependencies between variables. Causal dependencies refer to relationships where one variable directly affects the mechanism by which another variable is realized, and performing an intervention (manipulation) on a variable changes the data generation process (e.g., $p(Y = y|\operatorname{do}(X = x)) \neq p(Y = y|X = x)$). Statistical dependencies (correlations) only indicate that two variables are related, assuming a fixed data generation process; causal dependencies imply a cause-and-effect relationship and encode the variation in the data generation process under interventions on variables. Thus, causal models are more general than probabilistic models, as causal models enable reasoning about interventions and counterfactuals. Probabilistic models are estimated from observational data only (data collected under fixed data generation mechanisms) and cannot capture causal relationships; therefore, such probabilistic models cannot be used to reason about interventions and counterfactuals (without additional assumptions on the structure of the causal model). Still, probabilistic modeling is an essential component of causal modeling.

We make this note here because any discussion of artificial intelligence acting in the real world would not be sound without addressing causal models and their distinction from probabilistic models. However, further discussion of causal models is beyond the scope of this thesis, and the remainder of the thesis will focus solely on probabilistic models. We refer the reader to [190, 196] for a deeper review of modern causal modeling techniques.

2.2 Deep Probabilistic Models

Learning statistical dependencies between few variables is relatively straightforward and can often be achieved with classical learning methods [24]. For instance, estimation of conditional probabilities can be achieved with linear models or closed-form nonparametric models and requires a manageable amount of data and computational effort.

However, the situation becomes more challenging when dealing with a high number of variables in problems such as modeling statistical dependencies between pixels in a high-resolution image. In high-dimensional spaces, the complexity of relationships and interactions among variables increases significantly, making it difficult to capture these dependencies faithfuly.

The curse of dimensionality — the fact that the number of possible configurations of the data increases exponentially with the number of dimensions — makes estimating probability distributions from a finite number of data points in high-dimensional spaces a fundamentally ill-posed problem. In such vast spaces, even a large dataset represents only a sparse sampling of all possible data points. Consequently, many different probability distributions can agree with the observed data, making it difficult to uniquely determine the underlying distribution that generated the data. This ambiguity and underdetermination pose significant challenges for probabilistic modeling, as it becomes exceedingly difficult to infer accurate and reliable probability distributions without additional assumptions or constraints. One way to approach such challenging learning problems is to seek models with appropriate inductive biases. Inductive biases are assumptions built into a model family and a learning algorithm (often implicitly) that result in certain solutions being favored over others, thereby guiding the learning process. Without suitable inductive biases, models can overfit the limited data, capturing noise rather than meaningful patterns, or fail to generalize to unseen data points.

Deep neural networks (DNNs) [77, 272] have proven to provide inductive biases that are particularly suitable for modeling high-dimensional data such as images, texts, robotic control trajectories, and molecular structures. Deep probabilistic models (DPMs) are probabilistic machine learning models that parameterize probability distributions via DNNs and, thus, combine the representation learning capacity and generalization power of DNNs with probabilistic tools for representing and quantifying uncertainty.

A DPM defines a probability distribution $p_\theta(x)$ over data x parameterized by a DNN with parameters θ. In order to construct a DPM, one must specify parameterization (a way in which a DNN is used to define a probability distribution), training algorithm(s), and inference algorithm(s). Below, we provide an overview of several DPM families, parameterizations of high-dimensional probability distributions, and algorithms for training and inference.

2.2.1 Deep Probabilistic Models: Parameterization

Deep Discriminative Models (Classifiers / Regressors)

Deep discriminative models [130, 131, 139, 217] leverage neural networks to parameterize conditional distributions, enabling complex mappings from input data x to distributions over outputs y. For a classification problem with C classes, $p(y|x)$ is a categorical distribution over classes y. Typically, $p(y|x)$ is represented via the softmax function applied to the outputs of a neural network:

$$p_\theta(y|x) = [\text{softmax}(F_\theta(x))]_y, \qquad [\text{softmax}(z)]_i = \frac{\exp(z_i)}{\sum_{j=1}^{C}\exp(z_j)}, \tag{2.3}$$

where $F_\theta : \mathcal{X} \to \mathbb{R}^C$ denotes a mapping realized by a deep neural network with parameters θ.

Deep Autoregressive Models

Deep autoregressive models [1, 17, 18, 33, 39, 43, 54, 59, 67, 80–82, 137, 201, 202, 205, 237, 247, 248, 250] decompose the joint probability of a set of d random variables $x = \{x_i\}_{i=1}^d$ into a product of d conditional probabilities:

$$\begin{aligned} p_\theta(x_1, \ldots, x_d) &= p_\theta(x_1)p_\theta(x_2|x_1)p_\theta(x_3|x_1, x_2)\ldots p_\theta(x_d|x_1, \ldots, x_{d-1}) \\ &= p_\theta(x_1)\prod_{i=2}^{d} p_\theta(x_i|\{x_j\}_{j=1}^{i-1}), \end{aligned} \tag{2.4}$$

where each conditional probability $p_\theta(x_i|x_1, x_2, \ldots, x_{i-1})$ is typically realized by a deep discriminative model. Typical choices of network architectures include recurrent neural networks (RNNs), convolutional neural networks (CNNs) or transformers, all of which are designed to process long input sequences.

Variational Autoencoders (VAEs)

Variational Autoencoders (VAEs) [122, 249] are latent variables models. Models of this kind are based on the idea that a complex probability distribution $p_\theta(x)$ can be obtained by specifying a joint model $p_\theta(x, z) = p_\theta(x|z)p(z)$ with the auxiliary latent (unobserved) variable z. The modeling power of this approach is due to the fact that even if the factors $p(z)$, $p_\theta(x|z)$ are modeled in a relatively simple way, the implied marginal distribution

$$p_\theta(x) = \int p_\theta(x|z)p(z)\,dz, \tag{2.5}$$

can be very complex.

In a typical implementation of a VAE the prior over latent variable $p(z) = \mathcal{N}(z; 0, I)$ is a standard normal distribution and the likelihood $p_\theta(x|z)$ is defined by a "decoder" network F_θ, e.g.:

$$p_\theta(x|z) = \mathcal{N}(x; \mu(z, \theta), \mathrm{diag}(\sigma^2(z, \theta)), \tag{2.6}$$

where the likelihood of x given z is a normal distribution over x with mean $\mu(z, \theta)$ and the diagonal covariance matrix $\mathrm{diag}(\sigma^2(z, \theta))$ are derived from the outputs of the decoder network $[\mu(z, \theta), \sigma(z, \theta)] = F_\theta(z)$.

The marginal likelihood (2.5) requires the computation of a high-dimensional integral that can not be carried out exactly. To overcome this issue, VAEs introduce a variational posterior model $q_\phi(z|x)$ defined by an "encoder network" E_ϕ. Typical parameterization of the approximate posterior has the form

$$q_\phi(z|x) = \mathcal{N}(z; \mu(x, \phi), \mathrm{diag}(\sigma^2(x, \phi)), \tag{2.7}$$

where $[\mu(x, \phi), \sigma(x, \phi)] = E_\phi(x)$.

Together, the encoder and the decoder networks form a variational autoencoder. We provide an overview of VAE training procedure in Section 2.2.2.

Genrative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) [9, 78, 87, 116, 160, 183], use a parameterization that is similar to that of VAEs (2.5). However, the neural network underlying the model in GANs is called a "generator" $G_\theta : \mathcal{Z} \to \mathcal{X}$ and a typical parameterization of GAN, the generate directly outputs the sample $x = G_\theta(z)$, which corresponds to a Dirac-delta function likelihood $p_\theta(x|z) = \delta(x - G_\theta(z))$ and the marginal likelihood

$$p_\theta(x|z) = \int \delta(x - G_\theta(z))p(z)\, dz. \tag{2.8}$$

The main difference between GANs and VAEs is that GANs use a different kind of auxiliary model for training. To avoid the complexity of evaluating the integral (2.8), GANs employ an auxiliary "discriminator" model $\mathcal{D}_\phi : \mathcal{X} \to \mathbb{R}$ which is a discriminative model (e.g. a binary classifier) whose objective is to distinguish examples x produced by the generators from samples from the target distribution (e.g. dataset samples). We provide an overview of GAN training procedure in Section 2.2.2.

Both GANs and VAEs represent the probability distribution $p(x)$ implicitly, in the sense that the PDF $p(x)$ cannot be evaluated exactly. While likelihood evaluation in these models is hard, they provide a direct recipe for drawing samples from $p(x)$ by 1) drawing a latent variable sample $\hat{z} \sim p(z)$; 2) drawing a sample $\hat{x}$ from the conditional distribution $p(\hat{x}|z = \hat{z})$.

Energy-based Models (EBMs)

Energy-based Models (EBMs) [61, 95, 97, 141, 182, 230, 263] represent a complex probability distribution by using an energy-network $E_\theta(x) : \mathcal{X} \to \mathbb{R}$ to parameterize the "energy" function (unnormalized negative log-density):

$$-\log p_\theta(x) = E_\theta(x) + \log Z_\theta, \quad p_\theta(x) = \frac{1}{Z_\theta} \exp(-E_\theta(x)), \tag{2.9}$$

where the normalizing constant Z is

$$Z_\theta = \int \exp(-E_\theta(x))\, dx. \tag{2.10}$$

The energy function completely characterizes the probability distribution $p(x)$. By construction, EBMs can be readily used for all computations involving the un-normalized log-density. However, evaluating normalized (log-)density is challenging as the integral in the expression for the normalizing constant (2.10) is intractable. Training and inference algorithms for energy-based models (reviewed in Sections 2.2.2, 2.2.3 respectively) are based on techniques that avoid the direction evaluation of the normalizing constant [230].

Generative Flow Networks (GFlowNets)

Generative Flow Networks (GFlowNets) [16, 19, 158] represent a distribution $p(x)$ over discrete space $\mathcal{X}$ by parameterizing a stochastic policy that realizes a sample x through a sequence of discrete updates.

Formally, possible GFlowNet generation trajectories are represented by a direct acyclic graph (DAG) $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ with the a of states (vertices) $\mathcal{S}$ and a set of actions (directed edges) $\mathcal{A} \subset \{s \to s' \mid s, s' \in \mathcal{S}\}$.

The set of states $\mathcal{S}$ includes a special initial (source) state s_0, which has no incoming edges. The set $\mathcal{X}$ of discrete objects x realized by a GFlowNet is a subset of the set of states: $\mathcal{X} \subset \mathcal{S}$. The states $x \in \mathcal{X}$ are called terminal (sink) states. These states have no outgoing edges.

A GFlowNet generation trajectory $\tau = (s_0 \to s_1 \to \ldots \to s_T = x)$ starts at the initial state s_0, passes through a sequence of transitions $[(s_i \to s_{i+1})]_{i=0}^{T-1}, (s_i \to s_{i+1}) \in \mathcal{A}$ until a terminal state $x = s_T \in \mathcal{X}$ is reached. The sample realized by the trajectory $\tau = (s_0 \to s_1 \to \ldots \to s_T = x)$ is the terminal state of the trajectory $x = s_T$. We denote the number T of transitions in the trajectory τ by $|\tau|$.

A GFlowNet distribution $p_\theta(x)$ over $\mathcal{X}$ is derived from a distribution $p_\theta(\tau)$ of possible trajectories $\tau \in \mathcal{T}$

$$p_\theta(\tau) = \prod_{t=0}^{|\tau|-1} p_\theta(s_{t+1}|s_t). \tag{2.11}$$

The equation above implies that trajectories are realized by a stochastic Markov policy $p_\theta(s'|s)$ parameterized by a neural network $F_\theta(s)$. One possible parameterization of $p_\theta(s'|s)$ is the following

$$p_\theta(s'|s) = [\text{softmax}(F_\theta(s))]_{s'}, \tag{2.12}$$

where the network F_θ takes as input the current state s and outputs a vector of logits for all possible successor states s', such that $(s \to s') \in \mathcal{A}$.

Under trajectory distribution $p_\theta(\tau)$ (2.11), the probability of a trajectory passing through a given state $s \in \mathcal{S}$ is

$$p_\theta(s) := \mathbb{P}_\theta(s \in \tau) = \sum_{\tau \in \mathcal{T}_{s_0,s}} \prod_{i=0}^{|\tau|-1} p_\theta(s_{t+1}|s_t), \tag{2.13}$$

where $\mathcal{T}_{s_0,s}$ is the set of (partial) trajectories starting at s_0 and ending at s. Note that $p_\theta(s)$ above is just a shorthand notation for the probability of observing a state s on a trajectory τ under trajectory distribution (2.11). $p_\theta(s)$ is not a valid distribution over the states $s \in \mathcal{S}$, in particular $\sum_{s\in\mathcal{S}} p_\theta(s) > 1$.

The values of (2.13) evaluated at the terminal states $x \in \mathcal{X}$

$$p_\theta(x) = \sum_{\tau \in \mathcal{T}_{s_0,x}} \prod_{t=0}^{|\tau|-1} p_\theta(s_{t+1}|s_t), \quad x \in \mathcal{X}. \tag{2.14}$$

form a valid probability distribution over the terminal states: $\sum_{x\in\mathcal{X}} p_\theta(x) = \sum_{\tau\in\mathcal{T}} p_\theta(\tau) = 1$. $p_\theta(x)$ is the "model" distribution over the objects of interest realized by the GFlowNet policy $p_\theta(s'|s)$.

Markovian distributions over trajectories are connected to flows on graphs. To see that, one can rewrite the equation (2.13) as

$$p_\theta(s') = \sum_{s:(s\to s')\in\mathcal{A}} p_\theta(s)p_\theta(s'|s). \tag{2.15}$$

$p_\theta(s)$ can be interpreted as a "probability flow" through state s. Equation (2.15) describes the evolution of this flow under policy $p_\theta(s'|s)$. The goal of GFlowNet can be re-interpreted as learning the evolution of probability flow on the DAG $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ that leads to the target terminal distribution $p(x)$.

Note that GFlowNet creates a discrete-state probability flow on DAG $\mathcal{G}$ with a discrete set of transitions $\mathcal{A}$.

Normalizing Flows

Normalizing flows [57, 58, 123, 187, 208] parameterize a complex d-dimensional probability distribution $p_\theta(x), x \in \mathbb{R}^d$ through a sequence of transformations applied to a simple base distribution $p(z_0)$ (e.g. standard normal):

$$x = f_{\theta,t}(f_{\theta,T-1}(\ldots f_{\theta,1}(z_0)\ldots)) = (f_{\theta,T}\circ f_{\theta,T-1}\circ\ldots\circ f_{\theta,1})(z_0), \quad z_0 \sim p(z_0), \tag{2.16}$$

or, equivalently,

$$x = z_T, \qquad z_t = f_{\theta,t}(z_{t-1}),\ 1 \le t \le T, \qquad z_0 \sim p(z_0), \tag{2.17}$$

where each of the mappings $f_{t,\theta} : \mathbb{R}^d \to \mathbb{R}^d$ are learnable bijections.

The density of the distribution $p_\theta(x)$ induced by the normalizing flow (2.16)-(2.17) is given by

$$p_\theta(x) = p(z_0)\prod_{t=1}^{T}\Big|\det\frac{df_{t,\theta}}{dz_{t-1}}\Big|^{-1}, \quad z_T = x, \quad z_{t-1} = f^{-1}_{\theta,t}(z_t),\ 1 \le t \le T, \tag{2.18}$$

which follows from the change of variables rule

$$z_t = f_{\theta,t}(z_{t-1}), \qquad p(z_t) = p(z_{t-1})\Big|\det\frac{df_{t,\theta}}{dz_{t-1}}\Big|^{-1}, \tag{2.19}$$

applied repeatedly to sequence of variables $z_0, \dots, z_T = x$.

Re-written in terms of log-densities (2.18) gives

$$\log p_\theta(x) = \log p(z_0) - \sum_{t=1}^{T} \log \left| \det \frac{df_{t,\theta}}{dz_{t-1}} \right|. \tag{2.20}$$

Construction of normalizing flows requires designing neural blocks $f_{t,\theta}$ that implement bijection mappings and admit a simple way of computing the inverse $f_{t,\theta}^{-1}$ and the log-determinant of the Jacobian $\log \left| \det \frac{df_{t,\theta}}{dz_{t-1}} \right|$.

Normalizing flows support direct likelihood evaluation and provide simple recipe for sample generation, however the construction and learning of normalizing flows is complicated by the constraint of using only bijective transformations.

Note that normalizing flows define a continuous-state discrete-time probability flow $p(z_0), p(z_1), \dots, p(z_{T-1}), p(x)$ with the flow evolution (2.19).

Continuous Normalizing Flows (CNFs)

Continuous Normalizing Flows (CNFs) [5, 6, 38, 79, 148, 152, 231, 243, 265, 266] build a probabilistic distribution $p_\theta(x), x \in \mathbb{R}^d$ by transforming a prior probability density $p_0(z), z \in \mathbb{R}^d$ via a flow induced by an ordinary differential equation (ODE):

$$x = z(T), \quad \frac{d}{dt} z(t) = u_\theta(z(t), t),\ 0 < t < T, \quad z(0) \sim p_0, \tag{2.21}$$

where the vector-field $u_\theta(z, t)$ is realized by a neural network $u_\theta : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$.

Intuitively, (2.21) states that the CNF distribution $p_\theta(x)$ can be realized by drawing an initial position of a "particle" $z(0)$ from p_0 and then simulating the dynamics of a particle under the vector field $u_\theta(z, t)$. The particle evolution creates a probability flow $\{p_t\}_{t=0}^{T}$, that is the probability density $p_t(z)$ of $z(t)$ changes as $z(t)$ evolves. This PDF evolution is described by the well-known partial differential equation (PDE) called the Liouville equation:

$$\frac{\partial}{\partial t} p_t(z) = -\big(\nabla_z \cdot (u_\theta\, p_t)\big)\Big|_z = -\sum_{i=1}^{n} \frac{\partial}{\partial z_i}\big([u_\theta(z,t)]_i\, p_t(z)\big),\ 0 < t < T,\ z \in \mathbb{R}^d, \tag{2.22}$$

$$\begin{aligned}\frac{\partial}{\partial t} \log p_t(z) &= -\ \big(\nabla_z \cdot u_\theta + u_\theta \cdot \nabla_z \log p_t\big)\Big|_z \\ &= -\left(\sum_{i=1}^{d} \frac{\partial}{\partial z_i}[u_\theta(z,t)]_i + \sum_{i=1}^{d}[u_\theta(z,t)]_i \frac{\partial}{\partial z_i}\log p_t(z)\right).\end{aligned} \tag{2.23}$$

Evaluating the density $p_t(z)$ along the trajectory of ODE (2.21) gives

$$\frac{\partial}{\partial t} p_t(z(t)) = -p_t(z(t)) \sum_{i=1}^{d} \frac{\partial}{\partial z_i}[u_\theta(z(t),t)]_i, \tag{2.24}$$

$$\frac{\partial}{\partial t} \log p_t(z(t)) = -\sum_{i=1}^{d} \frac{\partial}{\partial z_i}[u_\theta(z(t),t)]_i = -\operatorname{Tr}\left(\tfrac{\partial u_\theta}{\partial z}(z(t),t)\right), \tag{2.25}$$

which is known as the instantaneous change of variables formula.

Given a point $x = z(T)$, the log-density $\log p_\theta(x)$ can be recovered by simulating ODE dynamics $z(t)$ backward in time

$$\begin{bmatrix} z_0 \\ \log p_\theta(x) - \log p_0(z_0) \end{bmatrix} = \begin{bmatrix} z(T) \\ 0 \end{bmatrix} + \int_T^0 \begin{bmatrix} u_\theta(z(t),t) \\ -\operatorname{Tr}\left(\frac{\partial u_\theta}{\partial z}(z(t),t)\right) \end{bmatrix} dt. \tag{2.26}$$

Once z_0 and $\log p_\theta(x) - \log p_0(z_0)$ are evaluated, $\log p_0(z_0)$ can be evaluated and added to the solution yielding $\log p_\theta(x)$.

Continuous normalizing flows support direct likelihood evaluation and provide a simple recipe for sample generation. Both of these tasks require numerically simulating the ODEs, and numerical simulation inevitably introduces approximation errors. Moreover, direct evaluation of the divergence $\operatorname{Tr}\left(\frac{\partial u_\theta}{\partial z}(z(t),t)\right)$ is computationally expensive and requires repeated calls to automatic differentiation. Instead, many practical realizations rely on stochastic trace estimators, which can speed up computation but create another source of approximation errors.

Note that continuous normalizing flows define a continuous-state continuous-time probability flow $\{p_t(z)\}_{t=0}^{T}$ with the flow evolution (2.22)-(2.25).

Diffusion Models

To describe the flow models above, we first introduced the parameterization of the dynamical systems (state update rules) and then wrote down the implied probability flows (probability update rules). We follow the opposite route to introduce diffusion models: start with a desired probability flow and then introduce state updates that produce the desired flow.

Diffusion Model Probability Flows. Diffusion models [99, 117, 226–229, 231] aim to reproduce the diffusion probability flow $\{q_t(z)\}_{t=0}^{T}$, which is constructed from the target data distribution $q_0 = p_{\text{data}}$ by applying Gaussian "noising" kernel (also called "forward process") $q(z_t|z_{\text{data}}) = \mathcal{N}(z_t; z_{\text{data}}, \sigma(t)^2 I)$:

$$q_t(z_t) = \int q(z_t|z_{\text{data}}) p_{\text{data}}(z_{\text{data}})\, dz_{\text{data}} = p_{\text{data}} \star \mathcal{N}(0, \sigma(t)^2 I),\ 0 \le t \le T, \tag{2.27}$$

where $\star$ denotes the convolution of probability density functions, and $\sigma(t) > 0$ is a smooth monotonously increasing function which controls the standard deviation of the Gaussian noise added at time t.

Crucially, for significantly large $\sigma(T)$ the end-time distribution $p_T(z)$ becomes very close to a pure Gaussian noise $q_T(z) \approx p_{\text{prior}}(z) = \mathcal{N}(z; 0, \sigma(T)^2)$. The idea of diffusion models is to learn a state update dynamics that initialized with a prior sample $z_T \sim p_{\text{prior}}(z)$ creates a flow p_t that runs backwards in time $t = T \to t = 0$ and reproduces the probability flow $p_t \approx q_t$ as closely as possible so that p_0 approaches q_0 which is minimally noised version of the target distribution p_{data}.

The diffusion probability flow (2.27) can be viewed as the solution of the heat equation PDE

$$\frac{\partial}{\partial t} q_t(z) = \dot{\sigma}(t)\sigma(t)\, \nabla_x \cdot (\nabla_x q_t(x)),\ 0 < t < T, z \in \mathbb{R}^d, \tag{2.28}$$

$$q_0 = p_{\text{data}} \star \mathcal{N}(0, \sigma(0)^2 I). \tag{2.29}$$

There are several ways of describing the probability flow (2.27)-(2.28) as the product of the evolution of a dynamical system. We review a few notable examples below.

In physics, diffusion processes arise as limits of random-walk processes. Consider a trajectory z_t of a particle that moves in random directions distributed by a Gaussian law

$$z_{t+\Delta t} - z_t = \Delta z_t \sim \mathcal{N}(\Delta z_t; 0, (\sigma(t+\Delta t)^2 - \sigma(t)^2)I). \tag{2.30}$$

Under the linearization $\sigma(t+\Delta t)^2 - \sigma(t)^2 \approx \frac{d}{dt}(\sigma(t)^2)\Delta t = 2\dot{\sigma}(t)\sigma(t)\Delta t$ and infinitesimal time increments Δt, the discrete-time process z_t converges to the Stochastic Differential Equation (SDE)

$$dz(t) = \sqrt{2\dot{\sigma}(t)\sigma(t)}dw_t,\ 0 < t < T, \qquad z(0) \sim q_0, \tag{2.31}$$

where w_t is the standard Wiener process (a.k.a., Brownian motion). The evolution of the probability density q_t of the SDE process is described by the Fokker-Planck PDE (a.k.a the Kolmogorov forward PDE):

$$\begin{aligned} \text{SDE: }\ & dz(t) = f(z(t),t)dt + g(t)dw_t, \\ & \Downarrow \\ \text{PDE: }\ & \frac{\partial}{\partial t}q_t(z) = -\nabla_z \cdot (f(z,t)q_t(z)) + \frac{1}{2}g(t)^2\nabla_z^2(q_t(z)), \end{aligned} \tag{2.32}$$

where $f(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is the drift term and $g(\cdot) : \mathbb{R} \to \mathbb{R}$ is the diffusion coefficient.

Note that the Fokker-Planck PDE (2.32) applied to the SDE (2.31) recovers the heat equation PDE (2.28).

The "forward" SDE (2.31) produces the probability flow $\{q_t(z)\}_{t=0}^T$ when ran forward in time. A stochastic process theory result of Anderson [7] states that for each "forward" SDE, there is a corresponding "backward" SDE that runs backwards in time and reproduces the same marginals $\{q_t\}_{t=0}^T$:

$$\begin{aligned} \text{Forward SDE: }\ & dz(t) = f(z(t),t)dt + g(t)dw_t, \\ & \Downarrow \\ \text{Backward SDE: }\ & dz(t) = \left[f(z(t),t) - g(t)^2\nabla_z \log q_t(z(t))\right]dt + g(t)d\overline{w}_t, \end{aligned} \tag{2.33}$$

where $\overline{w}_t$ is the standard Wiener process in reverse time. Applying the SDE reversal rule (2.33) to the diffusion SDE (2.36), we obtain the diffusion backward SDE

$$dz(t) = -2\dot{\sigma}(t)\sigma(t)\nabla_z \log q_t(z(t))dt + \sqrt{2\dot{\sigma}(t)\sigma(t)}d\overline{w}_t,\ 0 < t < T,\ z(0) \sim q_0. \tag{2.34}$$

The Fokker-Planck PDE (2.32) can be re-written (see e.g. [161, 231]) in the form of a Liouville PDE (2.22) for a determinstic ODE with the appropriately chosen vector field:

$$\begin{aligned} &\text{ODE:}\ \ dz(t) = \left[f(z(t),t)dt - \frac{1}{2}g(t)^2\nabla_z \log q_t(z(t))\right]dt, \\ &\qquad\qquad\qquad\qquad \Downarrow \\ &\text{PDE:}\ \ \frac{\partial}{\partial t}q_t(z) = -\nabla_z \cdot \left(f(z,t)q_t(z)\right) + \frac{1}{2}g(t)^2\nabla_z^2\left(q_t(z)\right). \end{aligned} \tag{2.35}$$

Applying the SDE (2.32) to ODE (2.35) conversion to the diffusion SDE (2.31), we obtain the diffusion ODE

$$dz(t) = -\dot{\sigma}(t)\sigma(t)\nabla_z \log q_t(z(t))\, dt,\ 0 < t < T, \quad z(0) \sim q_0, \tag{2.36}$$

which produces the probability flow $\{q_t(z)\}_{t=0}^T$ (2.27)-(2.28). As ODE dynamics is deterministic by construction the ODE (2.36) can be run both forwards and backwards in time.

Diffusion Model Parameterization. The analysis above shows that to achieve the goal of reversing the diffusion process (2.27)-(2.28), one can use either the backward SDE (2.34) or the backward ODE (2.36). Note that both (2.34) and (2.36) involve the score function $\nabla_z \log q_t(z)$ of the diffusion marginal $q_t(z)$. Dependence on the diffusion score prevents us from simulating backward SDE or ODE directly, as the score

$$\nabla_{z_t} \log q_t(z_t) = \nabla_{z_t} \log\left(\int q(z_t|z_0)q_0(z_0)\, dz_0\right), \tag{2.37}$$

is intractable. However, as we discuss in Section 2.2.2, one can train a learnable approximation $u_\theta(z,t) \approx \nabla_z \log q_t(z)$ using samples from p_{data} and the forward process

$q(z_t|z_{\text{data}})$ (2.27).

This observation inspires the construction of the diffusion model probability flows parameterized via backward SDE (2.34) or backward ODE (2.36) and the learnable score-function approximation $u_\theta(z,t)$ parameterized by a neural network (which is often called "score-network")

$$\begin{aligned} p^{\text{ODE}}_{\theta,t}(z)\colon & dz(t) = -\dot{\sigma}(t)\sigma(t)u_\theta(z(t),t)dt, && T > t > 0, \\ & z(T) \sim p_{\text{prior}}, \\ p^{\text{SDE}}_{\theta,t}(z)\colon & dz(t) = -2\dot{\sigma}(t)\sigma(t)u_\theta(z(t),t)dt + \sqrt{2\dot{\sigma}(t)\sigma(t)}d\overline{w}_t, && T > t > 0, \\ & z(T) \sim p_{\text{prior}}. \end{aligned} \tag{2.38}$$

These processes parameterize probability distributions $p^{\text{ODE}}_\theta(x)$, $p^{\text{SDE}}_\theta(x)$ which are obtained as the terminal distributions of respective probability flows $\{p^{\text{ODE}}_{\theta,t}\}^0_{t=T}$, $\{p^{\text{SDE}}_{\theta,t}\}^0_{t=T}$ at $t = 0$.

Note that diffusion models are realized by continuous-time, continuous-state stochastic, or deterministic processes. Our presentation and notation above follow those of Karras et al. [117]. There exist different descriptions of diffusion models [99, 117, 134, 226, 227, 231, 273]. Early formulations of diffusion models were described as discrete-time, continuous-state stochastic processes [99, 226, 227].

2.2.2 Deep Probabilistic Models: Training

In Section 2.2.1 we reviewed a set of deep probabilistic model families and techniques they employ for construction of high-dimensional probability distributions p_θ parameterized by deep neural networks. This section reviews techniques for formalization and construction of training algorithms. Training algorithms take a training data $\mathcal{D}_n = \{x_i\}^n_{i=1}$ as an input and produce an estimate of neural network parameters θ_* that agree with the data the most according to a specific notion of "agreement" or "optimality".

Throughout this section we assume that the dataset $\mathcal{D}_n$ consists of n i.i.d. samples from an unknown data generating distribution $p_*(x)$. We use $p_\theta(x)$ to denote the model distribution parameterized by θ and p_n to denote the empirical distribution constructed from the dataset $\mathcal{D}_n$:

$$p_n(x) = \frac{1}{n}\sum_{i=1}^{n} \delta(x - x_i), \quad \mathcal{D}_n = \{x_i\}_{i=1}^{n}, \quad x_i \overset{\text{i.i.d}}{\sim} p_*(x). \tag{2.39}$$

Large-Scale Training and Stochastic Gradient-Based Optimization

One of the key requirements for the design of training algorithms for deep probabilistic models is computational efficiency and the capacity to scale to large amounts of data. This requirement is motivated by the curse of dimensionality: faithful estimation of probability distributions in high dimensional data domains (where deep neural networks shine) requires a large number of samples. It is a common practice to train models on millions of examples. Some modern models are trained on datasets with sizes measured in billions and trillions of examples [1]. It has been observed that some phenomena and qualitative changes in model performance emerge only in models trained at scale (large models trained on large datasets) [see e.g. 15].

The core algorithmic technique behind large-scale training algorithms is stochastic gradient-based learning which combines two ideas: learning by gradient descent and utilization of stochastic gradient estimation.

Gradient descent has emerged as the go-to algorithm for training deep neural networks since their inception [217]. Suppose that model training can be cast as minimization of a function $\mathcal{L}(\theta, \mathcal{D}_n)$ that depends on the model parameters θ and the training dataset $\mathcal{D}_n$. We assume that $\mathcal{L}(\theta, \mathcal{D}_n)$ measure the agreement between the model and the data: the lower the value $\mathcal{L}(\theta, \mathcal{D}_n)$ the stronger the agreement. Moreover, we assume that $\mathcal{L}$ is smooth with respect to θ. We will refer to this function as loss function. The optimal parameters θ_* are those that minimize the loss:

$$\theta_* \in \operatorname*{Argmin}_{\theta} \mathcal{L}(\theta, \mathcal{D}_n). \tag{2.40}$$

Gradient descent (GD) progresses towards an optimal solution by repeatedly performing steps along the negative gradient of the loss function $\mathcal{L}$. Starting from an initialization θ_0 the gradient descent updates are given by

$$\theta_i = \theta_{i-1} - \eta_i \cdot \nabla_\theta \mathcal{L}(\theta_{i-1}, \mathcal{D}_n), \tag{2.41}$$

where $\{\eta_i\}$ is the sequence of step sizes $\eta_i > 0$. Since the negative gradient $-\nabla_\theta \mathcal{L}(\theta_i, \mathcal{D}_n)$ is the direction of the steepest decline of the loss in a vicinity of the current iterate θ_i, given appropriate step sizes gradient descent decreases the value of the loss at each step. Gradient descent is known to have linear convergence rate to the optimal solution for Lipschitz-smooth and strongly convex functions. We refer the reader to [30, 90, 135, 178, 232] for theoretical analysis of convergence of gradient descent for certain classes of loss functions.

Stochastic gradient descent (SGD) [210] extends gradient descent by minimizing the loss using the stochastic estimates of the gradient, rather than the exact gradient. Formally, an unbiased stochastic gradient for a loss function $\mathcal{L}(\theta, \mathcal{D}_n)$ is a random variable $g(\theta)$ whose expectation is the gradient of the loss: $\mathbb{E}[g(\theta)] = \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_n)$. It might seem that using stochastic gradients for gradient descent updates is suboptimal as stochastic gradient might include some noise and deviate from the exact gradient, and, therefore, might not provide the direction of steepest descent. However, in practice computing stochastic estimates of the gradient often requires less compute resources and can be done faster than computing the exact gradient. In many cases, performing just one exact gradient update takes the same time as performing thousands to millions stochastic gradient updates.

The most common scenario where stochastic gradient is used in machine learning is optimizing loss functions of the form

$$\mathcal{L}(\theta, \mathcal{D}_n) = \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, x_i), \tag{2.42}$$

where $\ell(\theta, x)$ is a per-example loss function. For such additive function the full gradient is given by

$$\nabla_\theta \mathcal{L}(\theta, \mathcal{D}_n) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \ell(\theta, x_i). \tag{2.43}$$

The computation time of the full gradient is proportional to the total dataset size n. A stochastic gradient estimate for an additive loss can be constructed as

$$g(\theta) = \frac{1}{m} \sum_{i \in \mathcal{B}_m} \nabla_\theta \ell(\theta, x_i), \qquad \mathcal{B}_m \subset \{1, \dots, n\},\ |B_m| = m, \tag{2.44}$$

where $\mathcal{B}_m$ is a randomly uniformly chosen subset of indices $\mathcal{B}_m \subset \{1, \dots, n\}$ of size m. The dataset subset $\{x_i\}_{i \in \mathcal{B}_m} \subset \mathcal{D}_n$ is called "mini-batch" and m is the mini-batch size. By construction, the mini-batch gradient is an unbiased estimate of the full gradient: $\mathbb{E}[g(\theta)] = \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_n)$. The time needed to compute the mini-batch gradient is proportional to the mini-batch size m, which can be chosen to be much smaller that the total dataset size ($m \ll n$). We refer the reader to [30, 90, 135] for discussion of convergence of stochastic gradient descent and its variants for certain classes of loss functions. The algorithms reviewed and developed in this thesis are based on SGD and its variants such as Adam [120].

Under the stochastic gradient-based learning paradigm the goal of learning algorithm design is to find 1) a loss function $\mathcal{L}(\theta)$ which measures agreement between model and data; and 2) a compute-efficient gradient estimator $g(\theta)$ for $\mathcal{L}$. Assuming that the implementation of the algorithm is based on stochastic optimization methods, in the remaining of this thesis we will often write down the loss functions in the more abstract way as expectations of the form

$$\mathcal{L}(\theta) = \mathbb{E}_{p_\theta(x)}[f(x, \theta)], \tag{2.45}$$

where both the distribution $p_\theta(x)$ and the function inside the expectation depend on parameters θ. Gradient estimation for such objectives can be formalized through the notion of stochastic computation graphs [173, 223]. Taking into the account the dependency of the probability distribution p_θ on the parameters often requires special care since it is a hard to build an efficient and accurate estimator for a general probability distribution which might depend on the parameters in a complex way. A set of techniques have been developed to address this problem [173].

For most of the objectives used in this thesis, the parameterization of the distribution p_θ is known, and the graident estimators can be obtained using the "re-parameterization trick" [122, 209]. Suppose that $x \sim p_\theta(x)$ can be obtained via transforming another random variable $\varepsilon \sim p(\varepsilon)$ by a parametric smooth function $x = T(\varepsilon, \theta)$, and $p(\varepsilon)$ admits efficient sampling. Then, objective (2.45) can be re-written as

$$\mathcal{L}(\theta) = \mathbb{E}_{p_\theta(x)}[f(x,\theta)], = \mathbb{E}_{p(\varepsilon)}[f(T(\varepsilon,\theta),\theta)], \tag{2.46}$$

where the distribution in the last expectation does not depend on θ. A Monte-Carlo estimate of the gradient $\nabla_\theta \mathcal{L}(\theta)$ can be obtained as

$$g(\theta) = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta f(T(\varepsilon_i,\theta),\theta), \quad \varepsilon_1, \ldots, \varepsilon_m \sim p(\varepsilon). \tag{2.47}$$

Maximum Likelihood Estimation

The learning problem can be viewed as the statistical estimation problem: finding the model $p_{\theta^*}(x)$ that provides the most likely explanation of the observed data $\mathcal{D}_n$. One way of formalizing this problem is through the maximum likelihood estimation. Given a dataset $\mathcal{D}_n$, the likelihood of this dataset to be drawn from the model p_θ is given by likelihood function

$$f(\theta, \mathcal{D}_n) = \prod_{i=1}^{n} p(x_i; \theta). \tag{2.48}$$

The maximum-likelihood estimate of the parameters θ are $\theta_{\text{MLE}} \in \text{Argmax}_\theta\, f(\theta, \mathcal{D}_n,)$ or, equivalently, the parameters that minimize the mean negative log-likelihood loss

$$\mathcal{L}_{\text{MNLL}}(\theta, \mathcal{D}_n) = \frac{1}{n}\sum_{i=1}^{n}[-\log p_\theta(x_i)]. \tag{2.49}$$

Note that one can re-write the mean negative log-likelihood loss as the expectation of the negative log-likelihood taken over the empirical data distribution

$$\mathcal{L}_{\text{MNLL}}(\theta, \mathcal{D}_n) = \mathbb{E}_{p_n(x)}[-\log p_\theta(x_i)]. \tag{2.50}$$

Divergence Minimization

This last expression suggests that we seek the model $p_\theta(x)$ that assigns high likelihood to points that are likely under the data distribution $p_n(x)$. In fact, one can connect the mean negative log-likelihood to a divergence (a kind of "distance" function) between probability distributions $p_n(x)$ and $p_\theta(x)$. A probability distribution divergence is a function $D(\cdot,\cdot)$ such that

$$1) D(p,q) \geq 0,\ \forall\, p,q \in \mathcal{P}(\mathcal{X}), \quad 2) D(p,q) = 0 \iff p = q, \tag{2.51}$$

where $\mathcal{P}(\mathcal{X})$ is the space of probability distributions over the space $\mathcal{X}$. As divergence functions are non-negative and take the value of zero if and only if the two distributions are equal, divergences can be used to measure discrepancy between two distributions and to guide parameterized distributions to alignment.

A well-known example of a divergence function is the Kullback-Leibler (KL) divergence $D_{\text{KL}}(\cdot,\cdot)$:

$$D_{\text{KL}}(p,q) = \mathbb{E}_{p(x)}\left[\log \frac{p(x)}{q(x)}\right] = \mathbb{E}_{p(x)}[\log p(x)] + \mathbb{E}_{p(x)}[-\log q(x)]. \tag{2.52}$$

The mean negative-likelihood loss is equivalent to the KL divergence between the data distribution $p_n(x)$ and the model distribution $p_\theta(x)$:

$$\mathcal{L}_{\text{MNLL}}(\theta, \mathcal{D}_n) = D_{\text{KL}}(p_n, p_\theta) + \text{Constant}. \tag{2.53}$$

This observation provides a more general view of model estimation, we seek the model distribution $p_{\theta^*}(x)$ that minimizes the discrepancy with the data distribution $p_n(x)$, where the discrepancy is measured by a divergence function $D(\cdot,\cdot)$ of our choice. In the remainder of this section, we show a few examples of learning algorithms derived from this general divergence minimization paradigm.

Exact Models: Direct Maximum-Likelihood Estimation

The minimization of the mean negative log-likelihood loss requires evaluating the log-likelihood $\log p_\theta(x)$ and computing its gradients w.r.t. parameters θ. In models that support direct log-likelihood evaluation, such as deep discriminative models, autoregressive models, normalizing flows, and CNFs, maximum likelihood training can be performed directly. The mean negative log-likelihood loss has the additive form and its stochastic gradient can be estimated with mini-batches.

VAEs: Variational Inference and Evidence Lower Bound

VAEs directly parameterize the conditional likelihood $p_\theta(x|z)$ and the prior $p(z)$, however the expression for the marginal likelihood $p_\theta(x)$ (2.5) involves an intractable integral over the latent variable z. While the marginal likelihood can not be evaluated directly, it can be approximated with the tecnhique known as "Variational Inference" (VI).

Variational Inference extends the model by introducing an auxiliary condtional distribution $q(z|x)$. Observe, that for any choice of the distribution $q(z|x)$ the following representation of the log of marginal likelihood $p_\theta(x)$ holds:

$$\begin{aligned}\log p_\theta(x) &= \mathbb{E}_{q(z|x)}[\log p_\theta(x)] &(2.54)\\ &= \mathbb{E}_{q(z|x)}[\log p_\theta(x|z) + \log p_\theta(z)] - \mathbb{E}_{q(z|x)}[\log p_\theta(z|x)] &(2.55)\\ &= \mathbb{E}_{q(z|x)}[\log p_\theta(x|z) + \log p(z) - \log q(z|x)] + D_{\mathrm{KL}}(q(z|x), p_\theta(z|x)). &(2.56)\end{aligned}$$

The expectation in the last expression is known as the "Evidence Lower Bound" (ELBO) or "Variational Lower Bound":

$$\begin{aligned}\mathrm{ELBO}(p_\theta(x|z), p(z), q(z|x), x) &= \mathbb{E}_{q(z|x)}[\log p_\theta(x|z) + \log p(z) - \log q(z|x)] &(2.57)\\ &= \log p_\theta(x) - D_{\mathrm{KL}}(q(z|x), p_\theta(z|x)) \le \log p_\theta(x), &(2.58)\end{aligned}$$

which indeed provides a lower bound on the log of marginal likelihood since $D_{\mathrm{KL}}(q(z|x), p_\theta(z|x))$ is non-negative for any $q(z|x)$.

Note that the expression for the ELBO includes logarithms of model distributions $\log p_\theta(x|z)$, $\log p(z)$, which can be evaluated directly. The only remaining unknown distribution is $q(z|x)$. The VAE model uses a parametric $q_\phi(z|x)$ parameterized by a neural network called "encoder" with parameters ϕ. With this model, a Monte-Carlo estimate of ELBO can be obtained by drawing samples from $q_\phi(z|x)$ and evaluating the logarithms of model distributions $\log p_\theta(x|z)$, $\log p(z)$, $\log q_\phi(z|x)$.

Thus, we derived a practical algorithm for the estimation and maximization of the ELBO. Maximization of ELBO does not correspond to direct maximization of log-likelihood $\log p_\theta(x)$. However, ELBO maximization is justified by the following observations

1. $\mathrm{ELBO}(p_\theta(x|z), p(z), q_\phi(z|x), x)$ is a valid lower bound on the log-likelihood $\log p_\theta(x)$. Thus, achieving a certain value of ELBO during ELBO maximization guarantees that the log-likelihood $\log p_\theta(x)$ is at least as large as the achieved value of ELBO.

2. The decomposition $\mathrm{ELBO}(p_\theta(x|z), p(z), q_\phi(z|x), x) = \log p_\theta(x) - D_{\mathrm{KL}}(q(z|x), p_\theta(z|x))$ implies that maximization of ELBO w.r.t. $q_\phi(z|x)$ doesn't change the value of $\log p_\theta(x)$ but makes the lower bound tighter. In particular, if $q_\phi(z|x)$ reaches the optimal solution $q_*(z|x) = p_\theta(z|x)$, the lower bound becomes tight, and ELBO is exactly equal to the log-likelihood.

Therefore, maximization of the ELBO w.r.t. both parameters θ of the decoder and parameters ϕ of the encoder approaches log-likelihood maximization (assuming that the encoder and decoder are flexible enough to represent the respective target distributions and the optimization is performed to optimality).

Negating the expression for ELBO (2.57) and averaging of examples x_i in the dataset $\mathcal{D}_n$ we obtain the ELBO loss function

$$\mathcal{L}_{\text{ELBO}}(\theta, \phi, \mathcal{D}_n) = \mathbb{E}_{p_n(x)}\mathbb{E}_{q_\phi(z|x)}[-\log p_\theta(x|z) - \log p(z) + \log q_\phi(z|x)] \tag{2.59}$$

$$= \mathbb{E}_{p_n(x)}\left[\mathbb{E}_{q_\phi(z|x)}[-\log p_\theta(x|z)] + D_{\text{KL}}(q_\phi(z|x), p(z))\right] \tag{2.60}$$

$$\geq \mathcal{L}_{\text{MNLL}}(\theta, \mathcal{D}_n). \tag{2.61}$$

This loss can be minimized by approximating the loss and its gradient with Monte-Carlo estimates (sampling $\hat{x} \sim p_n(x)$ and $\hat{z} \sim q_\phi(z|x = \hat{x})$) and evaluating all the log-likelihood values in the expression (2.59). For some choices of the encoder distribution $q_\phi(z|x)$ and the prior distribution $p(z)$, the KL divergence $D_{\text{KL}}(q_\phi(z|x), p(z))$ can be computed analytically instead of being approximated with MC estimates, and, then, the expression (2.60) can be used.

The first term in the expression (2.60) is known as the "log-likelihood loss" or the "reconstruction loss". This term encourages the encoder $q_\phi(z|x)$ and the decoder $p_\theta(x|z)$ to work together to reconstruct the sample x from its embedding z as accurately as possible. The second term is the "KL regularization term", which penalizes the encoder distribution $q_\phi(z|x)$ for deviation from the prior over latent codes $p(z)$.

GANs: Variational Form of Divergences and Game-Theoretic Learning Algorithms

Similar to VAEs considered above, GANs also do not provide a direct way of evaluating the likelihood $p_\theta(x)$ as this evaluation involves an intractable integral (2.8). While the two models parameterize the model distribution in a similar way, GANs work around the issue of direct likelihood evaluation in a different way.

Specifically, GAN training is built on the observation that a divergence between two distributions can be estimated by training an auxiliary discriminative model that scores samples from two distributions and aims to tell the two sets of samples apart.

Let us consider a concrete formalization of this idea. Given two probability density functions $p(x)$ and $q(x)$, we construct a probabilistic classifier that we will call a "discriminator" $r_\phi(y|x)$, where $y \in \{-1, +1\}$ is a binary label, and we parameterize the classifier with a neural network $u_\phi(\cdot) : \mathcal{X} \to \mathbb{R}$ with parameters ϕ:

$$r_\phi(y = \hat{y}|x) = \frac{1}{1 + \exp(-\hat{y} \cdot u_\phi(x))}, \quad \hat{y} \in \{-1, +1\}. \tag{2.62}$$

Suppose that we train the classifier to distinguish between samples $\hat{x} \sim p(x)$ drawn from $p(x)$ (that we will label with $y = +1$) and samples $\hat{x} \sim q(x)$ drawn from $q(x)$ (that we will label with $y = -1$) by minimizing binary classification mean negative log-likelihood

$$\begin{aligned}\mathcal{L}_D(\phi, p, q) &= \frac{1}{2}\mathbb{E}_{p(x)}[-\log r_\phi(y = +1|x)] + \frac{1}{2}\mathbb{E}_{q(x)}[-\log r_\phi(y = -1|x)] && (2.63)\\ &= \frac{1}{2}\mathbb{E}_{p(x)}[\log(1 + \exp(-u_\phi(x)))] + \frac{1}{2}\mathbb{E}_{q(x)}[\log(1 + \exp(u_\phi(x)))]. && (2.64)\end{aligned}$$

The expressions for the optimal function u^* that minimizes the classification negative log-likelihood loss and the associated optimal classifier r^* can be derived analytically and are given by

$$u^*(x) = \log\frac{p(x)}{q(x)}, \quad r^*(y = \hat{y}|x) = \frac{1}{1 + \exp\left(-\hat{y} \cdot \log\frac{p(x)}{q(x)}\right)}, \tag{2.65}$$

$$r^*(y = +1|x) = \frac{p(x)}{p(x) + q(x)}, \quad r^*(y = -1|x) = \frac{q(x)}{p(x) + q(x)}. \tag{2.66}$$

The optimal discriminator u^* is exactly the log of ratio of densities $\log\frac{p(x)}{q(x)}$ that appears in the expression for the KL divergence (2.52). Moreover, substituting the optimal solution u^* in (2.63) gives

$$\begin{aligned}\mathcal{L}_D^*(p, q) &= -\frac{1}{2}\mathbb{E}_{p(x)}\left[\log\frac{p(x)}{p(x) + q(x)}\right] - \frac{1}{2}\mathbb{E}_{q(x)}\left[\log\frac{q(x)}{p(x) + q(x)}\right] && (2.67)\\ &= -\frac{1}{2}\mathbb{E}_{p(x)}\left[-\log(2) + \log\frac{p(x)}{\frac{1}{2}(p(x) + q(x))}\right]\\ &\qquad - \frac{1}{2}\mathbb{E}_{q(x)}\left[-\log(2) + \log\frac{q(x)}{\frac{1}{2}(p(x) + q(x))}\right] && (2.68)\\ &= \log(2) - \frac{1}{2}D_{\text{KL}}\left(p(x), \tfrac{1}{2}p(x) + \tfrac{1}{2}q(x)\right)\\ &\qquad - \frac{1}{2}D_{\text{KL}}\left(q(x), \tfrac{1}{2}p(x) + \tfrac{1}{2}q(x)\right) && (2.69)\end{aligned}$$

$$= \log(2) - D_{\text{JS}}(p,q), \tag{2.70}$$

where $D_{\text{JS}}(\cdot,\cdot)$ is a symmetric divergence function known as Jensen-Shannon divergence. Using the last equation above, we can re-write the Jensen-Shannon divergence in a variational form, i.e. as a result of optimization w.r.t. an auxiliary function $u(\cdot)$:

$$\begin{aligned} D_{\text{JS}}(p,q) =& \log(2) \\ &+ \sup_{u(\cdot):\mathcal{X}\to\mathbb{R}} \frac{1}{2}\mathbb{E}_{p(x)}[-\log(1+e^{-u_\phi(x)})] + \frac{1}{2}\mathbb{E}_{q(x)}[-\log(1+e^{u_\phi(x)})]. \end{aligned} \tag{2.71}$$

While the direct expression for D_{JS} can not be evaluated since GANs do not provide access to the log-likelihood, the variational form can be estimated (at least approximately) by training the discriminator network u_ϕ.

During training we are interested in minizining a divergence between the model distribution $p_\theta(x)$ and the data distribution $p_n(x)$ [1]. Using the variational form of JS divergence, we can write down the divergence minimization as

$$\min_\theta \max_\phi \frac{1}{2}\mathbb{E}_{p_\theta(x)}[-\log(1+e^{-u_\phi(x)})] + \frac{1}{2}\mathbb{E}_{p_n(x)}[-\log(1+e^{u_\phi(x)})], \tag{2.72}$$

which is a nested optimization problem.

The nature of this problem is different from the optimization problems we considered so far. In both direct mean negative log-likelihood minimization and the ELBO loss minimization we considered a single objective that was to be minimized w.r.t. all trainable parameters (only θ in case of exact models, and θ and ϕ in case of VAEs). Problem (2.72) can be considered as a nested optimization problem, where for each value of θ, one first has to solve the inner maximization problem w.r.t. ϕ, and then find the value of the that minimizes $\widetilde{\mathcal{L}}(\theta) = \max_\phi \frac{1}{2}\mathbb{E}_{p_\theta(x)}[-\log(1+e^{-u_\phi(x)})] + \frac{1}{2}\mathbb{E}_{p_n(x)}[-\log(1+e^{u_\phi(x)})]$.

Another way to view the problem (2.72) is to think about it as a game — a problem

[1] Technically the empirical distribution does not have an ordinary density function. For the purpose of theoretical analysis, the data distribution $p_n(x)$ can be replaced with a smoothed distribution $\tilde{p}_n(x) = \frac{1}{n}\sum_{i=1}^n \mathcal{N}(x; x_i, \varepsilon)$ obtained by applying the Gaussian noise with very small variance ε to data points.

that involves two different sets of parameters (players) optimizing different objective functions (utilities). In the case of (2.72), the two players are the generator parameters θ and the discriminator parameters ϕ, the loss (negative utility) functions of the players (to be minimized) are

$$\mathcal{L}_G(\theta, \phi) = \mathbb{E}_{p_\theta(x)}[-\log(1 + e^{-u_\phi(x)})] + \frac{1}{2}\mathbb{E}_{p_n(x)}[-\log(1 + e^{u_\phi(x)})], \tag{2.73}$$

$$\mathcal{L}_D(\theta, \phi) = \mathbb{E}_{p_\theta(x)}[\log(1 + e^{-u_\phi(x)})] + \frac{1}{2}\mathbb{E}_{p_n(x)}[\log(1 + e^{u_\phi(x)})]. \tag{2.74}$$

Game (2.73)-(2.74) is a zero-sum game, meaning that loss of one player is exactly the gain of the other player: $\mathcal{L}_G(\theta, \phi) = -\mathcal{L}_D(\theta, \phi)$. If one player decreases their loss function by δ, the other player suffers an increase of their loss by δ.

We can connect the optimal solution of divergence minimization, i.e. generator parameters θ^* such that $p_{\theta^*} = p_n$, to Nash equilibria (a kind of solution) of game (2.73)-(2.74). Formally, if there exists θ^* such that $p_{\theta^*}(x) = p_n(x),\ \forall\, x$ and there exists ϕ^* such that $u_{\phi^*}(x) = 0,\ \forall\, x$ is a constant discriminator function, then (θ^*, ϕ^*) together result in a Nash equilibrium for (2.73)-(2.74), meaning that

$$\mathcal{L}_G(\theta^*, \phi^*) \leq \mathcal{L}_G(\theta, \phi^*) \ \ \forall\, \theta, \tag{2.75}$$

$$\mathcal{L}_D(\theta^*, \phi^*) \leq \mathcal{L}_D(\theta^*, \phi) \ \ \forall\, \phi. \tag{2.76}$$

In other words, the point (θ^*, ϕ^*) is such that none of the players θ, ϕ can decrease their respective loss function below $\mathcal{L}_G(\theta^*, \phi^*)$, $\mathcal{L}_D(\theta^*, \phi^*)$ by making unilateral changes to their states. See [70, 78] for proof of the result and its generalizations.

While GAN formulation in the form (2.72) involves nested optimization, i.e. the inner maximization w.r.t. ϕ needs to be solved to (near) optimality before θ can be updated, the game formulation (2.73)-(2.74) suggests another algorithm where both players simultaneously make progress by locally decreasing their utilities:

$$\begin{bmatrix}\theta_{i+1}\\ \phi_{i+1}\end{bmatrix} = \begin{bmatrix}\theta_i\\ \phi_i\end{bmatrix} + \eta_i \begin{bmatrix}v_\theta(\theta, \phi)\\ v_\phi(\theta, \phi)\end{bmatrix}, \quad v(\theta, \phi) = \begin{bmatrix}v_\theta(\theta, \phi)\\ v_\phi(\theta, \phi)\end{bmatrix} = \begin{bmatrix}-\nabla_\theta \mathcal{L}_G(\theta, \phi)\\ -\nabla_\phi \mathcal{L}_D(\theta, \phi)\end{bmatrix}, \tag{2.77}$$

where $v(\theta, \phi)$ is the negative gradient vector field associated with the game defined by $\mathcal{L}_G(\theta, \phi)$, $\mathcal{L}_D(\theta, \phi)$. The algorithm (2.77) is known as simultaneous gradient descent (SimGD). Given sufficiently small stepsizes SimGD either enters a limit or converges to a stationary point, i.e. a point $(\overline{\theta}, \overline{\phi})$ such that $v(\overline{\theta}, \overline{\phi}) = 0$. In certain cases the stationary point of the vector field recovers a local Nash equilibrium, which is one way of formalizing the notion of local solution of the game, a state where neither player can decrease their loss through a unilateral local change of their state. However, not all stationary points are local Nash equilibria. We refer the reader to [165, 167, 183] for analysis of critical points in continuous games and convergence of SimGD.

In practice GANs are trained by procedures similar to SimGD, except that exact gradient are computationally expensive and stochastic gradients are used instead. Note that stochastic gradients of the loss functions (2.73)-(2.74) can be estimated by drawing samples from the model $p_\theta(x)$ and from the data distribution $p_n(x)$. Often GAN training algorithms alternate between individual player updates (Alternating Gradient Descent) instead of updating players' states simultaneously. It is also a common practice to introduce individual learning rates for each player and perform multiple updates for one player before switching to updating the other player.

Above we introduced GANs training as minimization of Jensen-Shannon divergence in the variational form. Variational estimators or bounds have been derived for other divergences such as f-divergences [181, 183] and Wasserstein distance [8, 9]. These results inspired variants of GANs, optimizing various probability divergences [8, 9, 160, 183]. A class of divergences, which are expressed as

$$D_{\text{MDD-}\mathcal{F}}(p, q) = \sup_{f \in \mathcal{F}} \big(\mathbb{E}_{p(x)}[f(x)] - \mathbb{E}_{q(x)}[f(x)]\big), \tag{2.78}$$

where $\mathcal{F}$ is a class of functions, are known as maximum mean discrepancy or integral probability metrics [85].

We conclude this section by introducing Wasserstein distance. The Wasserstein-1 distance or the Earth-Mover distance is defined as

$$D_W(p, q) = \inf_{\gamma \in \Pi(p,q)} \mathbb{E}_{\gamma(x,y)} \big[\|x - y\| \big], \tag{2.79}$$

where $\Pi(p, q)$ denotes the set of all joint distributions $\gamma(x, y)$ on $\mathcal{X} \times \mathcal{X}$ whose marginals are p and q respectively. The joint distribution γ can be interpreted as the transportation plan, i.e. one can think about the value of $\gamma(x, y)$ as the amount of probability mass that must be transported from x to y in order to transform distribution p into distribution q. The total mass transported from a given point x is $\int \gamma(x, y)\, dy = p(x)$ and the total mass transporter to a given point y is $\int \gamma(x, y)\, dx = q(y)$. In this interpretation, $D_W(p, q)$ is minimal possible total probability mass transportation cost assuming that the cost of transporting one unit of mass from x to y is $c(x, y) = \|x - y\|$. The Wasserstein distance is a valid metric on the space of probability distributions. Note that the Wasserstein distance explicitly takes the distance between points in the space $\mathcal{X}$ into account, while KL and JS divergences are expressed in terms of the density ratios $\frac{p(x)}{q(x)}$, $\frac{p(x)}{\frac{1}{2}(p(x)+q(x))}$.

Practical approximation of the Wasserstein distance in the form (2.79) involves solving a complex minimization problem over joint probability distributions $\gamma \in \Pi(p, q)$. However, using the Kantorovich-Rubinstein duality [252], the Wasserstein distance can be re-written as

$$D_W(p, q) = \sup_{f : \|f\|_L \le 1} \mathbb{E}_{f(x)}[f(x)] - \mathbb{E}_{q(x)}[f(x)], \tag{2.80}$$

where the supremum is taken over all the 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}, |f(x) - f(y)| \le \|x - y\|,\ \forall\, x, y \in \mathcal{X}$. This representation inspires the min-max Wasserstein GAN (WGAN) game [9]

$$\min_{\theta} \max_{\phi} \mathbb{E}_{p_\theta(x)}[f_\phi(x)] - \mathbb{E}_{p_n(x)}[f_\phi(x)], \tag{2.81}$$

where the Wasserstein discriminator f_ϕ need to be parameterized (or regularized) to ensure that f_ϕ has a bounded Lipschitz norm. In practice this can be achieved with weight clipping [9], gradient penalties [8], or spectral normalization layers [171].

EBMs: Maximum Likelihood

The training methods we considered so far relied on the direct access to the log-likelihood $\log p_\theta(x)$ or direct mechanisms of drawing samples from the model $\hat{x} \sim p_\theta(x)$. Energy-based models parameterize both the log-likelihood function and the generative process implicitly through the energy function $E_\theta(x)$ (2.9)-(2.10).

Given a dataset $\mathcal{D}_n$ the mean negative log-likelihood (2.49) for an energy-based model (2.9)-(2.10) is

$$\mathcal{L}_{\text{MNLL}}(\theta) = \mathbb{E}_{p_n(x)}[E_\theta(x)] + \log Z_\theta = \mathbb{E}_{p_n(x)}[E_\theta(x)] + \log \left(\int \exp(-E_\theta(x))\, dx \right). \quad (2.82)$$

While the first term and its gradient can be estimated by drawing samples from the dataset and evaluating the energy function, the second term involves an intractable integral.

One way of estimating the gradient $\nabla_\theta \mathcal{L}_{\text{MNLL}}(\theta)$ is based on the fact [see e.g., 230] that the gradient $\nabla_\theta \log Z_\theta$ of the logarithm of the normalizing constant can be expressed as

$$\nabla_\theta \log Z_\theta = \mathbb{E}_{p_\theta(x)}[-\nabla_\theta E_\theta(x)], \quad (2.83)$$

where the expectation is computed over the model distribution $p_\theta(x)$.

Substituting the last expression in the mean negative log-likelihood loss and differentiating w.r.t. θ gives

$$\nabla_\theta \mathcal{L}_{\text{MNLL}}(\theta) = \mathbb{E}_{p_n(x)}[\nabla E_\theta(x)] - \mathbb{E}_{p_\theta(x)}[\nabla_\theta E_\theta(x)]. \quad (2.84)$$

This expression implies that maximum likelihood training in energy-based models can be performed by estimating the gradient (2.84) via drawing samples from the data distribution $p_n(x)$ and the model distribution $p_\theta(x)$, and evaluating the gradient of the energy function. However, drawing samples from the model distribution $p_\theta(x)$ is not straightforward. There is a family of training methods for EBMs that are based on approximating the model distribution with Markov Chain Monte Carlo (MCMC) procedures. One early algorithm of this kind is Contrastive Divergence proposed by Hinton [97]. More recent techniques have been developed in [61, 182, 263]. We refer the reader to [230] for a more detailed review of EBM training techniques.

EMBs and Diffusion Models: Denoising Score Matching

One way to avoid the need to estimate the normalizing constant in the energy-based models is to focus on the score function — the gradient of the log-density w.r.t. input x:

$$\nabla_x \log p_\theta(x) = \nabla_x(-E_\theta(x) - \log Z_\theta) = -\nabla_x E_\theta(x), \tag{2.85}$$

which does not depend on Z_θ (constant as a function of x). Instead of measuring the discrepancy between distributions in terms of likelihood functions, one can use score functions as a basis for comparison. Fisher divergence is a divergence function based on this principle:

$$D_F(p, q) = \mathbb{E}_{p(x)}\left[\frac{1}{2}\left\|\nabla_x \log p(x) - \nabla_x \log q(x)\right\|^2\right]. \tag{2.86}$$

The methods based on score function discrepancy are known as score matching [106].

Naturally, one might want to train the model $p_\theta(x)$ by minimizing the Fisher divergence $D_F(p_n, p_\theta)$ as the estimation of the expectation in (2.86) requires only the samples from the data distribution $p_n(x)$. However, for p_n, an empirical distribution, the log of the density (and its gradient) is poorly behaved $\log p_n(x)$ is discontinuous and is negative-infinity for $x \notin \mathcal{D}_n$. One way to alleviate this problem is to consider a smoothed version of the data distribution $q(x) = \int q(x|x_0)p_n(x_0)dx_0$, where $q(x|x_0)$ is a smooth noising kernel (e.g. Gaussian noise $q(x|x_0) = \mathcal{N}(x; x_0, \sigma^2 I)$).

Vincent [253] has shown that estimation of $D_F(q(x), p_\theta(x))$ is tractable since the divergence can be represented as

$$D_F(q(x), p_\theta(x)) = \mathbb{E}_{q(x)}\left[\frac{1}{2}\left\|\nabla_x \log q(x) - \nabla_x \log p_\theta(x)\right\|^2\right] \tag{2.87}$$

$$= \mathbb{E}_{p_n(x_0)}\mathbb{E}_{q(x|x_0)}\left[\frac{1}{2}\left\|\nabla_x \log q(x|x_0) - \nabla_x \log p_\theta(x)\right\|^2\right] + C, \quad (2.88)$$

where C is a constant which does not depend on θ and does not affect the optimization. Estimation of (2.88) requires drawing samples $\hat{x}_0 \sim p_n(x_0)$, $\hat{x} \sim q(x|x_0 = \hat{x}_0)$, and evaluating the gradient of the log of the noising kernel $\nabla_x \log q(x|x_0)$ and the score of the model $\nabla_x \log p_\theta(x)$. Therefore, the noising kernel $q(x|x_0)$ is required to provide a simple way of evaluating the gradient $\nabla_x \log q(x|x_0)$. Note that minimization of $D_F(q(x), p_\theta(x))$ will lead to $p_\theta(x)$ matching $q(x)$ not the original data distribution $p_n(x)$, so when choosing the noising kernel $q(x|x_0)$ one must ensure that the noise introduced by $q(x|x_0)$ to $p_n(x)$ does not distort the original distribution excessively.

Substituting the Gaussian noising kernel $q(x|x_0) = \mathcal{N}(x; x_0, \sigma^2 I)$ or equivalently $x = x_0 + \sigma\varepsilon$, $\varepsilon \sim \mathcal{N}(\varepsilon; 0, I)$ and the EBM $p_\theta(x)$ in (2.88) and omitting multiplicative and additive constants gives

$$\mathcal{L}_{\text{DSM}}(\theta) = \mathbb{E}_{p_n(x)}\mathbb{E}_{\mathcal{N}(\varepsilon;0,I)}\left[\left\|\nabla_x E_\theta(x + \sigma\varepsilon) - \frac{\varepsilon}{\sigma^2}\right\|^2\right]. \quad (2.89)$$

Assuming that there exists θ^* such that $p_\theta^*(x) = q(x)$, $\forall\, x$, the parameters θ^* constitute a minimizer of the denoising score matching loss (2.89) and recover the score function

$$\nabla_x \log p_{\theta^*}(x) = -\nabla_x E_{\theta^*}(x) = \nabla_x \log\left(\int \mathcal{N}(x|x_0; \sigma^2 I) p_n(x_0)\, dx_0\right). \quad (2.90)$$

In Section 2.2.1, we introduced diffusion model parameterization based on a learnable approximation $u_\theta(z, t)$ to the time-dependent score function $\nabla_{z_t} \log q(z_t)$ (2.37). Note that this diffusion score (2.37) is exactly the solution (2.90) of the minimization of the denoising score matching loss (2.89) for the time-dependent noising kernel $q(z_t|z_0) = \mathcal{N}(z_t; z_0, \sigma^2(t)I)$. Thus, training in diffusion models can be performed via time-dependent denoising score matching. Karras et al. [117] propose to re-parameterize the score function approximation $u_\theta(z, t) = (D_\theta(z, \sigma(t)) - x)/\sigma^2(t)$, where $D_\theta(z, \sigma(t))$ is learnable "denoiser" network with input and output pre-conditioning (see [117] for details), and train the denoiser by minimizing the denoising loss

$$\mathcal{L}_{\text{denoising}}(\theta) = \mathbb{E}_{p_{\text{train}}(\sigma)}\mathbb{E}_{p_n(x_0)}\mathbb{E}_{\mathcal{N}(\varepsilon;0,I)}\left[\lambda(\sigma)\|D_\theta(x_0+\sigma\varepsilon,\sigma)-x_0\|^2\right], \tag{2.91}$$

where $p_{\text{train}}(\sigma)$ is a distribution over noise levels $\sigma \in [\sigma_{\min}, \sigma_{\max}]$, and $\lambda(\cdot) : [\sigma_{\min}, \sigma_{\max}] \to (0,\infty)$ is a positive weighting function. Given the optimal denoiser D_{θ^*} the score can be approximated as

$$\nabla_{z_t} \log q_t(z_t) \approx u_{\theta^*}(z,t) = \frac{D_{\theta^*}(z_t,\sigma(t)) - z_t}{\sigma^2(t)}. \tag{2.92}$$

CNFs: Flow Matching

Continuous Normalizing Flows provide tractable mechanisms for estimating the log-likelihood via simulation of the ODE (2.26). It is possible to train CNFs via maximum likelihood estimation. However, log-likelihood estimation involves numerical integration of the ODE. Differentiation of the log-likelihood can be done either by differentiating from the ODE solver or estimating the gradient as a numerical solution of the adjoint ODE [38, 198]. Both the numerical integration of ODEs and estimation of the trace of the Jacobian $\operatorname{Tr}\left(\frac{\partial u_\theta}{\partial z}(z(t),t)\right)$ either incur high computational costs or introduce approximation errors.

A recent line of work [5, 6, 148, 152, 243] proposed an alternative way of training CNFs. The goal of CNF training is to find probability flow $\{p_{\theta,t}(z)\}_{t=0}^T$ induced by the vector field $u_\theta(z,t)$ such that the terminal time distribution $p_{\theta,T}(z)$ approximates the data distribution $p_n(x)$ as closely as possible. Suppose we construct a probability flow $\{q_t(z)\}_{t=0}^T$ which has fixed end-point distributions $q_0 = p_0$ and $q_T = p_n$ and satisfies the continuity equation for a known vector field $v(z,t)$

$$q_0(z) = p_0(z),\ q_T(z) = p_n(z),\ \frac{\partial}{\partial t}q_t(z) = -(\nabla_z \cdot (v \cdot q_t))|_z,\ 0 < t < T, z \in \mathbb{R}^n. \tag{2.93}$$

Then, we can formalize training of the CNF probability flow as matching of the prescribed probability flow $\{q_t(z)\}_{t=0}^T$. In particular, if one can draw samples from $q_t(z)$ and evaluate the vector field $v(z,t)$, then CNF can be trained with the flow matching objective

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{\mathcal{U}(t;[0,T])}\mathbb{E}_{q_t(z)}\left[\|v(z,t) - u_\theta(z,t)\|^2\right], \tag{2.94}$$

where $\mathcal{U}(t;[0,T])$ is the uniform distribution over time $t \in [0,T]$. The main challenge in applying this training procedure lies in the construction of a probability flow $q_t(z)$ and the corresponding vector field $v(z,t)$ which would admit tractable sampling and vector field evaluation. Below we consider two ways of constructing such probability flows and describe the corresponding training objectives.

Lipman et al. [148], Tong et al. [243] proposed a method for constructing probability flows connecting given pair of distribution using a mixture of simpler probability paths. The construction starts from a conditional probability flow $q_t(z|\alpha)$ that varies with some conditioning variable α and that is generated by a conditional vector field $v(z,\alpha,t)$. If the conditional variable α has the distribution $q(\alpha)$, then by marginalizing over α gives a mixture probability flow (or marginal probability flow):

$$q_t(z) = \int q_t(z|\alpha)q(\alpha)\,d\alpha. \tag{2.95}$$

One can show (see [148, 243]) that marginal probability flow is generated by the vector field

$$v(z,t) = \int v(z,\alpha,t)\frac{q_t(z|\alpha)q(\alpha)}{q_t(z)}\,d\alpha = \mathbb{E}_{q_t(\alpha|z)}[v(z,\alpha,t)]. \tag{2.96}$$

By substituting this expression for the marginal vector filed in the expression for the flow matching loss (2.94), simplifying the expression, and omitting the constants that do not affect the optimization, one can arrive at the conditional flow matching loss [148, 243]:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{\mathcal{U}(t;[0,T])}\mathbb{E}_{q(\alpha)}\mathbb{E}_{q_t(z|\alpha)}\left[\|v(z,\alpha,t) - u_\theta(z,t)\|^2\right]. \tag{2.97}$$

This objective can be optimized and its minimization leads to our goal of CNF training given that

- the conditional probability flow $q_t(z|\alpha)$ is generated by the conditional vector field $v(z, \alpha, t)$ from initial condition $q_0(z|\alpha)$;
- $q_0(x) = \int q_0(x|\alpha)q(\alpha)$ is equal to desired prior distribution $p_0(x)$;
- $q_T(x) = \int q_0(x|\alpha)q(\alpha)$ is equal to the target data distribution $p_n(x)$;
- drawing samples from $q_t(z|\alpha)$ is tractable;
- evaluation of $v(z, \alpha, t)$ is tractable.

Lipman et al. [148], Tong et al. [243] constructed several conditional probability flows and corresponding vector fields. Moreover, these works have shown that certain choices recover diffusion probability flows as well as dynamic optimal transport probability flows, and Schrödinger bridge probability flows. Note that the relationship between the marginal flow matching objective (2.94) and the conditional flow matching objective (2.97) is similar to the relationships between the denoising score matching objective with the score of the marginal distribution (2.87) and the denoising score matching objective with the conditional score (2.89): learning by regression to the conditional vector field is simpler that regressing to the marginal vector field.

Albergo et al. [5], Albergo and Vanden-Eijnden [6] propose another way of constructing probability paths and learning the associated vector fields. Their approach is based on a "stochastic interpolants". Given two artibtraty probability distributions $q_0(z)$ and $q_T(z)$, one can construct a probability flow $\{q_t(z)\}_{t=0}^{T}$ that interpolates between $q_0(z)$ and $q_T(z)$ by drawing samples z_t at time $t \in [0, T]$ as

$$z_t = I(t, z_0, z_T) + \gamma(t)\varepsilon,\ z_0 \sim q_0(z),\ z_T \sim q_T(z),\ \varepsilon \sim \mathcal{N}(\varepsilon; 0, I),\ t \in [0, T], \quad (2.98)$$

where

- $I : [0, T] \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ is a smooth interpolant function such that $I(0, z_0, z_T) = z_0$ and $I(T, z_0, z_T) = z_T$,
- $\gamma(t)$ is a standard deviation function such that $\gamma(0) = \gamma(T) = 0$, $\gamma(t) > 0, t \in (0, T)$ and $\gamma^t(\cdot) \in C^2([0, T])$,

- z_0, z_T, ε are independently drawn from respective distributions [2].

Albergo et al. [5] show that the stochastic interpolant probability flow is generated by the vector field $u^*(z,t)$

$$u^*(z,t) = \mathbb{E}_{z_0,z_T,\varepsilon}\left[\frac{\partial}{\partial t}I(t,z_0,z_T) + \frac{\partial}{\partial t}\gamma(t)\varepsilon \,\middle|\, z_t = z\right], \tag{2.99}$$

which can be recovered by minimization of the quadratic objective

$$\begin{aligned}\mathcal{L}_{\mathrm{SI}}(\theta) = \mathbb{E}_{\mathcal{U}(t;[0,T])}\mathbb{E}_{q_0(z_0)}\mathbb{E}_{q_T(z_T)}\mathbb{E}_{\mathcal{N}(\varepsilon;0,I)}\Bigg[\\ \frac{1}{2}\|u_\theta(I(t,z_0,z_T),t)\|^2 - \left(\frac{\partial}{\partial t}I(t,z_0,z_T) + \frac{\partial}{\partial t}\gamma(t)\varepsilon\right)\cdot u_\theta(I(t,z_0,z_T),t)\Bigg].\end{aligned} \tag{2.100}$$

By construction, sampling from all distributions and evaluation of all terms appearing in (2.100) can be carried out exactly. Albergo et al. [5] provide details of the mathematical analysis and constructive examples of stochastic interpolatns, discuss connections to flow matching methods and diffusion models, and describe ODE and SDE sampling procedures for the stochastic interpolant probability flows.

GFlowNets: Trajectory Balance

The conceptual approach to GFlowNet training is similar to the approaches developed for conditional normalizing flows. In fact, the two frameworks have deep connections [134, 159, 273]: GFlowNets can be interpreted as discrete analogues of CNFs or diffusion models; CNFs and diffusion can be derived from an extension of GFlowNet to continuous-state-action spaces [134]. The GFlowNet probability flow recursion (2.15) and its variants play the same role in the analysis of discrete GFlowNet probability flows as the Liouville equation / continuity equation (2.22) plays in the analysis of deterministic continuous probability flow and the Fokker-Planck / Kolmogorov Forward equation (2.32) plays in the analysis of stochastic continuous probability flows.

[2] For the sake of brevity we are omitting some technical conditions on the I and γ. We refer the reader to Albergo et al. [5] for details.

In fact, all of these equations can be interpreted through the lens of the principle of (probability) mass conservation.

Before we describe training objectives for GFlowNet training, let us introduce the notion of target distribution for GFlowNets. In GFlowNet literature, it is common to specify the target distribution in terms of the reward function $R : \mathcal{X} \to [0, \infty)$, which maps the terminal states x to non-negative reward functions $R(x) \geq 0$. It is assumed that the reward function is available and can be evaluated efficiently for any terminal state. The target distribution $p^*(x)$ over terminal states $x \in \mathcal{X}$ is defined as

$$p^*(x) = \frac{1}{Z} R(x), \quad Z = \sum_{x' \in \mathcal{X}} R(x'), \tag{2.101}$$

where Z is a normalizing constant. The reward function defines the unnormalized probability mass function of $p^*(x)$. While it is standard to describe training in terms of the rewards, this description does not prevent one from training a GFlowNet given a dataset of examples $\mathcal{D}_n = \{x_i\}_{i=1}^n$. Given a dataset, one can define an empirical reward function and the empirical distribution

$$R_n(x) = \sum_{i=1}^{n} I[x = x_i], \quad p_n(x) = \frac{1}{n} R_n(x), \tag{2.102}$$

where $I[\cdot]$ is the indicator function. Below, we describe GFlowNet training algorithms using the reward function $R(x)$.

Given a parameterized GFlowNet policy $p_\theta(s'|s)$ the likelihood of a given terminal x is given by (2.14). Most interesting applications of GFlowNets involve DAG $(\mathcal{S}, \mathcal{A})$ with a combinatorial number of states and/or actions (usually, the DAG is specified implicitly). In this case, the summation over all trajectories $\Sigma_{\tau \in \mathcal{T}_{s_0,x}}$ terminating at x in (2.14) can not be carried out in a reasonable time (the number of paths in DAGs with big branching factor is combinatorial in the number of states). Using the recursion (2.15), one could evaluate $p(x)$ via dynamic programming which would take time and space proportional to $|\mathcal{S}| + |\mathcal{X}|$ which is also prohibitively expensive in GFlowNet problems such as generation of molecular graphs from fragments [16].

Instead of pursuing maximum likelihood training, we could specify a target probability flow on a DAG and train the policy by matching this flow. Given access to the rewards $R(x)$ of terminal states, we can construct a target probability flow $q(s)$ by introducing a backward policy $q_B(s_*|s)$ which for a given state s gives a probability distribution over all predecessor states $s_* : (s_* \to s) \in \mathcal{A}$. A simple example of a backward policy is the uniform policy that equally distributes the mass between all predecessor states s_* of a given state. Similar to the model probability flow (2.13), (2.15) induced by the forward policy $p_\theta(s'|s)$, we can write the backward probability flow $q(s)$ as

$$\begin{aligned} q(s) &= \sum_{x\in\mathcal{X}} \left[p^*(x) \sum_{\tau\in\mathcal{T}_{s,x}} \prod_{t=0}^{|\tau|-1} q_B(s_t|s_{t+1}) \right] \\ &= \frac{1}{Z} \sum_{x\in\mathcal{X}} \left[R(x) \sum_{\tau\in\mathcal{T}_{s,x}} \prod_{t=0}^{|\tau|-1} q_B(s_t|s_{t+1}) \right], \end{aligned} \tag{2.103}$$

and in the recurrent form as

$$q(s) = \sum_{s':(s\to s')\in\mathcal{A}} q(s')q_B(s|s'),\ \forall s \in \mathcal{S}\setminus\mathcal{X},\ \ q(x) = p^*(x) = \frac{1}{Z}R(x),\ \forall x\in\mathcal{X}. \tag{2.104}$$

Starting from the definition of GFlowNet probability flows, one can show [19] that if the forward $p_\theta(s)$ and the backward probability flows $q(s)$ are equal, then for any edge $(s \to s') \in \mathcal{A}$ the probability of observing $(s \to s')$ on a trajectory drawn from the probability flow is the same under both forward and backward flows:

$$q(s \to s') = q(s)p_\theta(s'|s) = q_B(s|s')q(s'). \tag{2.105}$$

The above equation is known as the ”detailed balance” equation, and it inspired several training objectives for GFlowNets [16, 158]. In this review, we will focus only on the trajectory balance loss, which we present below.

Given a trajectory $\tau = (s_0 \to \ldots \to s_n = x)$ on a GFlowNet DAG, we can write down the detailed balance equation (2.105) for all edges and, after simple algebraic manipulations, arrive at a trajectory balance condition

$$\prod_{t=0}^{|\tau|-1} p_\theta(s_{t+1}|s_t) = \frac{1}{Z} R(x) \prod_{t=0}^{|\tau|-1} q_B(s_t|s_{t+1}), \tag{2.106}$$

where we assumed that both probability flows generate the terminal distribution $p^*(x) = \frac{1}{Z}R(x)$. This constraint is satisfied for all trajectories if and only if the policies $p_\theta(s'|s)$ and $q_B(s|s')$ generate the same probability flows [19, 158]. This observation inspires a training method based on the minimization of the trajectory balance loss

$$\mathcal{L}_{\text{TB}}(\theta) = \mathbb{E}_{\tau \sim \widetilde{p}(\tau)} \left[\left(\log \frac{Z_\theta \prod_{t=0}^{|\tau|-1} p_\theta(s_{t+1}|s_t)}{R(x) \prod_{t=0}^{|\tau|-1} q_B(s_t|s_{t+1})} \right)^2 \right], \tag{2.107}$$

where the expectation can be taken over any probability distribution $\widetilde{p}(\tau)$ over trajectories, which assign positive probabilities to all trajectories, and Z_θ is a scalar parameter that is learned alongside the parameters of the policy network p_θ. Minimization of $\mathcal{L}_{\text{TB}}(\tau)$ to 0 leads to learning the probability flow that generates the desired terminal distribution $p^*(x) \propto R(x)$ [19, 158]. Malkin et al. [158] propose to use the model trajectory distribution $p_{\widetilde{\theta}}(\tau)$ as $\widetilde{p}(\tau)$, where $\widetilde{\theta}$ is a variable of the current parameters θ or a moving average of recent values of θ [3]. Drawing trajectory samples from $p_{\widetilde{\theta}}(\tau)$ can be accomplished simply by unrolling the policy $p_{\widetilde{\theta}}(s'|s)$. In theory, any strictly positive distribution over trajectories is a valid choice; however, in practice, the training dynamics of the model strongly depend on the trajectories sampled during training. If a dataset $\mathcal{D}_n$ of terminal samples is available, one can generate trajectories by following the backward policy $q_B(s|s')$ starting from a terminal state $\hat{x} \sim p_n(x)$ drawn from the dataset. We note that [158] propose to parameterize both the forward and the backward policies $p_\theta(s'|s)$ and $p_{\theta,B}(s|s')$ to have more flexibility in the choice of the target probability flow (generated by the backward policy). In this case, the parameters of both policies can still be learned with the trajectory balance loss (2.107).

[3]We distinguish between $p_{\widetilde{\theta}}$ used in the base of the expectation in (2.107) and the policy p_θ appearing inside the expectation, as Malkin et al. [158] point out that is not necessary to account for the influence of trajectory generation distribution on θ when computing gradients of $\mathcal{L}_{\text{TB}}$.

2.2.3 Deep Probabilistic Models: Inference

After a deep probabilistic model has been specified and trained, the final model $p_\theta(x), x \in \mathbb{R}^d$ (here θ denotes the final value of the model parameters) can be used to ask questions related to staistical dependencies between random variables $\{[x]_i\}_{i=1}^d$. This section provides a brief overview of some inference techniques for deep probabilistic models.

Likelihood evaluation. Given a model $p_\theta(x)$, one might want to evaluate the likelihood $p_\theta(x')$ or log-likelihood $\log p_\theta(x')$ at various points x'.

When discussing training in Section 2.2.2, we mentioned that in deep discriminative models, autoregressive models, and normalizing flows model parameterization enables exact evaluation of likelihood / log-likelihood. CNFs and diffusion models (in ODE form) enable approximate evaluation of log-likelihood via numerical simulation of ODEs of type (2.26).

GFlowNets do not provide direct mechanisms for likelihood evaluation. In principle likelihood can be evaluated by direct summation over trajectories (2.14), or via dynamic programming (2.15) or sampling.

Energy-based models by construct enable direct evaluation of unnormalized likelihood and unnormalized log-likelihood (negative enegergy). Estimation of exact likelihood, requires approximation of the normalizing constant $Z_\theta = \int \exp(-E_\theta(x))\, dx$.

In VAEs, exact likelihood evaluation involves an intractable integral (2.5) over latent variable z, which can be estimated with samples. As we discussed in Secion 2.2.2, the ELBO (2.57) provides a lower bound on the log-likelihood and can be estimated with sampling. There is a line of work on developing upper bounds (e.g., [233]) and tighter lower bounds (e.g.,[35]) on log-likelihood for VAEs.

GANs do not provide exact mechanisms for likelihood evaluation and compared to VAEs do not provide an encoder model that can be used to estimate a lower bound on log-likelihood. One can use the relation between the optimal discriminator and the density ratio (2.65), to estimate the ratio of densities between the model distribution $p_\theta(x)$ and a given background distribution $q(x)$.

Generation (Sampling). Drawing samples $\hat{x} \sim p_\theta(x)$ allows one to generate examples that "resemble" examples from the training data $\mathcal{D}_n$ in some aspects (at least in statistical dependencies between coordinates $\{[x]_i\}_{i=1}^d$ as represented by the model $p_\theta(x)$). Many approximate inference algorithms (and training procedures) rely on sampling from the model as a subroutine.

GANs, VAEs, Normalizing Flows, CNFs, diffusion models, GFlowNets provide direct mechanisms for drawing samples. In the case of CNFs and diffusion models numerical integration of ODEs / SDEs is required. Sampling from deep discriminative models, and deep autoregressive Models typically involves sampling from low-dimensional distributions $p(y|x)$ or $p(x_i|x_1, \dots x_{i-1})$, which are amenable to sampling techniques such as inverse CDF sampling, importance sampling, rejection sampling, Gibbs sampling and others (see e.g. Section 29 in [157]).

Among the models we discussed, energy-based models are the only model family where the parameterization is not directly connected to sampling. Since EBMs provide access to the unnormalized log-likelihood $-E_\theta(x) = \log p_\theta(x) + \log Z_\theta$ and the score function $-\nabla_x E_\theta(x) = \nabla_x \log p_\theta(x)$, the samples can be generated with sampling techniques which only require access to score and/or unnormalized log-likelihood. Gradient-based Markov Chain Monte Carlo algorithms [65, 83, 177, 188] is a family of efficient algorithms of this kind. For example, Langevin MCMCM constructs a iterative process

$$x_{t+1} = x_t + \frac{1}{2}\varepsilon^2 \nabla_x \log p_\theta(x_t) + \varepsilon z_i, \qquad i = 0, 1, \dots, T, \tag{2.108}$$

where x_0 is an initial sample drawn from a simple prior distribution, $\varepsilon > 0$, and $\{z_i\}$ are i.i.d. samples from the standard Gaussian distribution. When $\varepsilon \to 0$ the stationary distribution of this process is guaranteed to recover $p_\theta(x)$ (under some regularity conditions). For large enough T, x_T can be used as sample from the data distribution. In practice one have to use finite step sizes $\varepsilon > 0$ which introduces a discretization error, which could be corrected with a Metropolis-Hastings rejection procedure [89]. Langevin MCMC with Metropolis-Hastings correction is known as

Metropolis-Adjusted Langevin Algorithm. We refer the reader to [230] for a review of MCMC-based sampling techniques for EBMs in the context of maximum-likelihood training, and to Sections 29, 30, and 41 in [157] for introduction to gradient-based Sequential Monte Carlo methods.

Conditional (Controllable) Generation. After the model $p_\theta(x)$ was trained, one might want to generate samples $\hat{x}$ that satisfy certain conditions or draw samples that simultaneously have high likelihood under $p_\theta(x)$ and high value of some external scoring function. We refer to such tasks as controllable generation. This task can be formalized as drawing samples from a modified version of the learned distribution.

One way to derive new distributions from $p_\theta(x)$ is via conditioning. Starting from $p_\theta(x)$, one can consider associated conditional distributions

$$p(x_{\overline{I}}|x_I) = \frac{p_\theta(x_{\overline{I}}, x_I)}{p_\theta(x_I)}, \qquad p(x|y) = \frac{p_\theta(x)p(y|x)}{\int p_\theta(x')p(y|x')dx'}, \tag{2.109}$$

where $I \subset \{1, \ldots, d\}$, $x_I = [x_i]_{i \in I}$ denotes a subset of dimensions of $x \in \mathbb{R}^d$, $\overline{I} = \{1, \ldots, d\} \setminus I$, and y denotes an external random variable with a specified conditional likelihood $p(y|x)$. Sampling from the conditional distribution $p(x_{\overline{I}}|x_I)$ can be interpreted as filling the values of the missing components $x_{\overline{I}}$ of x given the subset of observed components x_I (data imputation, completion, inpating). Conditional generation from $p(x|y)$ can be interpreted as generation of samples that simultaneously have high likelihood under model and agree with observation y. Here we interpret y as a stochastic observation (function) of x (think of $\hat{y} \sim p(y|x)$ as a generalization of $y = f(x)$). We can also re-interpret sampling from $p(x|y)$ by re-writing

$$p(x|y) \propto \exp\left(\log p_\theta(x) + f_y(x)\right), \quad f_y(x) = \log p(y|x), \tag{2.110}$$

and observing that up-to a normalizing constant (independent of x) the conditional log-likelihood $\log p(x|y)$ is equal to the sum of the model likelihood $\log p_\theta(x)$ and a "scoring function" $f_y(x) = \log p(y|x)$. This function $f_y(x) = \log p(y|x)$ can encode relative preference between different points x or measure the extent to which x agrees with certain constraints.

One way to build a model capable of controllable generation is to prescribe the desired controls at the training time and train the model to perform controllable generation, e.g. train a conditional generative model $p_\theta(x|y)$ on a dataset of pairs (x_i, y_i) and introduce y as an additional input to neural networks in the model. In many applications one might want to re-use and control a pre-trained unconditional model $p_\theta(x)$.

Autoregressive models support conditional generation of sequence suffix $x_{i+1}, \dots, x_d$ given observed prefix $x_1, \dots, x_i$ by design. For example a language model trained to generate text documents, can generate a plausible continuation of a text given a few starting words (sentences, paragraphs, ...).

Energy-based models support conditional generation via MCMC sampling with modified energy functions $-\log p(x_{\overline{I}}|x_I = \hat{x}_I) = E_\theta(x)|_{x_I=\hat{x}_I} + C$ or $-\log p(x|y) = E_\theta(x) - \log p(y|x) + C$.

Research on design choices and training objectives for training conditional GANs and VAEs supporting various control mechanisms (specified at training time) received significant attention (e.g. [169], [116], [170], [107]). There are also works addressing contional generation with pre-training GANs [163] and VAEs [88] by learning a condtional encoder $q(z|y)$ which maps y to a distribution of latent codes z that produce outputs x agreeing with y.

Conditional sampling with pre-trained diffusion models can be realized via the technique known as classifier guidance [226, 231]. A likelihood function $p_0(y|x_0)$ specified for the terminal (clean) object x_0 at $t = 0$ induces a conditional distribution $p_0(x_0|y) \propto p_{\theta,0}(x_0)p_0(y|x_0)$ the conditional diffusion flow initialized with $p_0(x_0|y)$ is given by

$$p_t(x_t|y) = \int q(x_t|x_0)p_0(x_0|y)\, dx_0. \tag{2.111}$$

Since the conditional probability flow $p_t(x_t|y)$ is generated by the same noising process $q(x_t|x_0)$ as the pre-trained diffusion probability flow $p_{t,\theta}(x_t)$, the conditional probability flow can be reversed and realized by the backward SDE (2.34) or backward ODE

(2.36) provided a conditional score function $\nabla_x \log p_t(x_t|y)$. The conditional score can be decomposed as

$$\nabla_x \log p_t(x_t|y) = \nabla_x \log p_{\theta,t}(x_t) + \nabla_x \log p_t(y|x_t), \tag{2.112}$$

where the first term is approximated by the pre-trained (unconditional) score network $u_\theta(x_t, t) \approx \nabla_x \log p_{\theta,t}(x_t)$, and the second term is the gradient of the log-probability of the time-dependent classifier $p_t(y|x_t)$. The probability of conditioning variable y given a noised datapoint x_t is given by

$$\begin{aligned} p(y|x_t) =& \frac{p_t(x_t|y)p(y)}{p_t(x_t)} = \frac{p(y)}{p_t(x_t)} \int q(x_t|x_0) \frac{p(y|x_0)}{p(y)} p(x_0)\, dx_0 \\ =& \mathbb{E}_{p(x_0|x_t)}[p_0(y|x_0)]. \end{aligned} \tag{2.113}$$

Given access to the terminal time classifier $p_0(y|x_0)$, the time-dependent classifier $p_{\phi,t}(y|x_t) \propto f_\phi(y, x_t, t)$, parameterized by a nerual network $f_\phi(y, x_t, t)$ with parameters ϕ, can be learned by minimizing the negative log-likelihood loss

$$\mathcal{L}(\phi) = \mathbb{E}_{\mathcal{U}(t;[0,T])} \mathbb{E}_{p(x_t, x_0)} \mathbb{E}_{p_0(y|x_0)}[-\log p_{\phi,t}(y|x_t)]. \tag{2.114}$$

The pairs (x_t, x_0) can be either generated from the model $p_{\theta,t}(x_t)$ (by simulating backward ODE/SDE) or by applying the noising process $q(x_t|x_0)$ to the clean data $x_0 \sim p(x_0)$ (either sampled from the model or from the dataset).

Section 5.3 provides a review of recent work on controllable generation with diffusion models. Methods for conditional generation with GFlowNets are discussed in Chapter 5.

Model Composition. In practical applications, one might want to combine several deep probabilistic models trained for different tasks or on different datasets. With the growing availability of generative models pre-trained at scale, it becomes increasingly important to re-use and integrate previously trained models for new tasks. Compositional methods provide a powerful framework for this, allowing the capacities of multiple models to be combined in a way that enables solving tasks different from those specified at training time. Moreover, compositional approaches offer mechanisms for controlling the generation results with respect to multiple criteria, enhancing the flexibility and applicability of deep probabilistic models.

In terms of probabilistic models, composition can be understood as the construction of a complex probability distribution from available pre-trained probabilistic models. This can be described at the level of probability distributions, where a complex distribution is the result of applying a composition operation to several input probability distributions. Alternatively, compositions can be constructed by coupling and coordinating the generative processes that realize these distributions. We refer the reader to [60] for review of compositional generative modeling techniques. Methods for composition of energy-based and diffusion models are also reviewed in Section 5.3. Chapter 5 addresses composition operations for GFlowNets and diffusion models.

Chapter 3

Pairwise-Discriminator Objectives for Generative Adversarial Networks

3.1 Introduction

In Generative Adversarial Networks (GANs) [78] we seek to align the generated samples with the real ones (e.g., images). The generative model in GANs is trained by minimizing a discrepancy or divergence measure between the two distributions. This divergence measure is realized by a discriminator trained to separate real examples from those sampled from the model [183].

Despite their appeal, GANs are known to be hard to train due to stability issues. Since the estimation is typically setup as two objectives, one for the generator, the other for the discriminator, the desired solution is analogous to a Nash equilibrium of the associated game. Without additional regularization, the dynamics between the two can become unstable [168], and lead to a never-ending game. While there are multiple reasons for instability, in this Chapter, we focus in particular on what we define as alignment instability: the instability of alignment around the optimal solution(s).

The generator sees the training signal, the divergence measure, only through the discriminator. As a result, a generator that aligns perfectly with the target distribution can be thrown off the alignment by a sub-optimal discriminator. In other words,

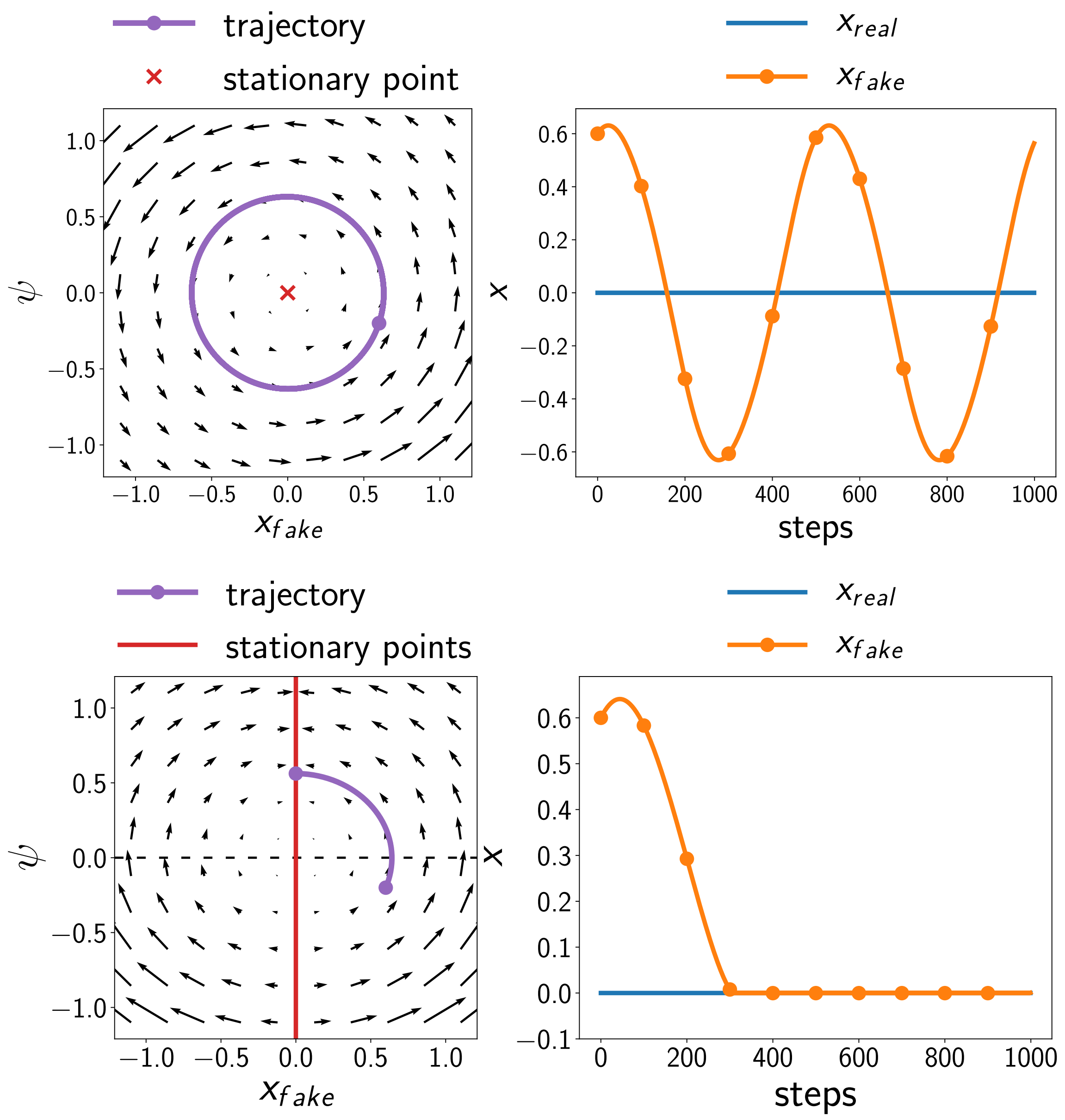


Figure 3-1: Demonstration of adversarial training dynamics on toy examples for: **(top)** unary discriminator with standard objective; **(bottom)** pairwise discriminator with our objective. **Left:** vector field and training trajectories for a toy generator $q(x) = \delta(x - x_{\text{fake}})$ and a discriminator D_ψ parameterized by a single real number ψ. **Right:** trajectory of fake and real samples: x_{fake} and $x_{\text{real}} = 0$.

generator can achieve alignment if and only if the discriminator reaches its optimum at the same time.

We illustrate the instability problem and our approach to resolving it with a toy example [168] in Figure 3-1. In this example, the generative model is simply

$q(x) = \delta(x - x_{\text{fake}})$, i.e., concentrated on a single point x_{fake}, which is the parameter to optimize. The goal is to align it with a real point fixed at $x_{\text{real}} = 0$. The discriminator is a simple classifier $D_\psi(x) = \psi \cdot x$ parameterized by slope ψ. The top left panel in Figure 3-1 gives the vector field for an alternating gradient descent training as well as an example trajectory. The training objective is given by the zero-sum game:

$$\min_{x_{\text{fake}}} \max_{\psi} [f(\psi \cdot x_{\text{fake}}) + f(0)], \quad f(t) = -\log(1 + e^{-t}).$$

The top right panel in Figure 3-1 shows the time evolution of x_{fake} in relation to x_{real}. Note, in particular, that even when x_{fake} reaches the target position $x_{\text{real}} = 0$ (perfect alignment) the sub-optimal discriminator drives the points apart.

In this Chapter, we focus on a different class of discriminators that operate on pairs of samples, trained to identify whether the samples come from the same distribution or not. Utilizing such pairwise discriminators, we identify a family of training objectives which ensures alignment stability even if the discriminator is sub-optimal.

The bottom panels in Figure 3-1 illustrate the same example, now with a pairwise discriminator: $D_\psi(x, y) = \psi \cdot |x - y|^\gamma$, where γ is a constant: $\gamma \geq 1$. The left panel again shows the vector field for alternating gradient updates between x_{fake} and ψ, resulting from our objective function described in Section 3.5. In this case, the alignment $x_{\text{fake}} = 0$ is a stationary point for *any* discriminator. The time evolution now shows that the alignment is preserved. In the experiment described in Section 3.7.1, we observe alignment instability occurring in a controlled setting with deep GAN architectures (Figure 3-2). Each panel in the figure shows time evolution of the generator over the course of training. In this experiment a deep generator is re-parameterized with a single scalar α so that the generated distribution is aligned with the target at $\alpha^* = 0$. The pattern observed in the toy example in Figure 3-1 holds in Figure 3-2 where a unary objective of SGAN results in oscillating training curve and pairwise objective of PairSGAN derived from our approach stabilizes training dynamics. We refer to Appendix A.7.1 for a detailed description of the toy example in Figure 3-1 and to Section 3.7.1 and Appendix A.8.1 for details on Figure 3-2.

We make following contributions:

1. In Section 3.4, we introduce a family of training objectives with pairwise discriminators, that preserve the distribution alignment, if achieved, regardless of the discriminator status.

2. In Section 3.5.2, we show that in our setup, only the generator needs to converge, and we provide conditions for local convergence.

3. In Section 3.5.3, we introduce the notion of a sufficient discriminator that formalizes the relationship between the capacities of the discriminator and generator.

4. In Section 3.7, we design and carry out experiments which evaluate alignment instability in deep models and we empirically confirm that PairGAN ensures stability of alignment. In experiments on real image datasets we find that PairGAN provides training stability.

All proofs can be found in Appendix A.

3.2 Related Work

Goodfellow et al. [78] proposed Generative Adversarial Networks and showed that the associated min-max game can be viewed as minimization of Jensen-Shannon divergence. It was pointed out by Nowozin et al. [183] that the standard GAN objective is a special case of a broader family of min-max objectives corresponding to f-divergences. Arjovsky et al. [9] showed that game-theoretic setup of GANs can be extended to approximately optimize the Wasserstein distance. Mao et al. [160] propose LSGAN which uses a least squares objective related to Pearson χ^2 divergence.

Mescheder et al. [167] and Nagarajan and Kolter [176] proved that GAN training convergences locally for absolutely continuous distributions. However, GANs are commonly used to approximate distributions that lie on low-dimensional manifolds [8]. Mescheder et al. [168] showed that in this case, many training methods do

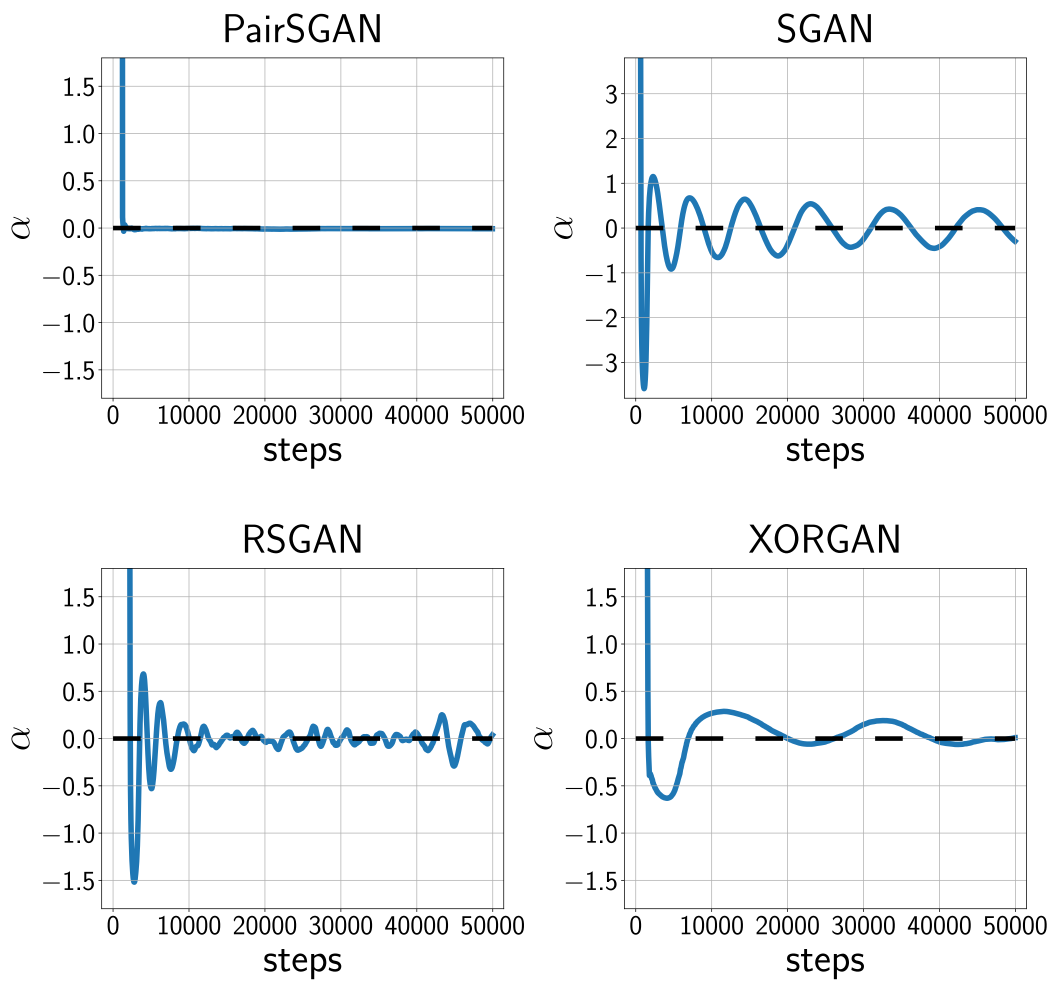


Figure 3-2: Training curves for DCGAN generator re-parameterized with a single scalar parameter α for fixed generator matching experiment (Section 3.7.1) with different GAN models. $\alpha^* = 0$ correspond to the generator perfectly aligned with the target distribution.

not guarantee local convergence without additional regularizations, such as gradient penalties [214], which enjoy both theoretic guarantees and empirical improvements.

There is a body of work on GANs which utilize pairwise discriminators for improving training dynamics. To combat mode collapse, Lin et al. [147] proposed to feed a pack of multiple samples from the same distribution to the discriminator rather than a pair of samples from different distributions. In Section 3.6 we discuss the relation between our methods and closely related Relativistic GAN [114], XORGAN [245], and MMD-GAN [144, 256].

3.3 Background

Let $\mathcal{X}$ denote the space of objects (e.g. images). We consider functional spaces $\mathcal{F}(\mathcal{X})$ of real-valued functions $f : \mathcal{X} \to \mathbb{R}$ operating on $\mathcal{X}$. In this Chapter we consider two particular settings: **(i)** $\mathcal{X} = \{x_1, \dots, x_k\}$ is a finite set, $\mathcal{F}(\mathcal{X}) = \mathbb{R}^k$; **(ii)** $\mathcal{X} \subset \mathbb{R}^k$ is a compact set, $\mathcal{F}(\mathcal{X}) = L_2(\mathcal{X})$. In both cases, $\mathcal{F}(\mathcal{X})$ is a vector space with inner product. In our analysis, we build intuition about linear functionals and linear operators on $\mathcal{F}(\mathcal{X})$ treating them as finite-dimensional vectors and matrices. While the space $\mathbb{R}^k$ provides useful intuition, our results naturally extend to $L_2(\mathcal{X})$.

Objectives for GANs. Consider a generative modelling setup where we want to approximate a distribution of "real" objects $p(x)$ with a distribution of generated ("fake") objects $q(x)$. The training in GANs is performed by solving a game between the generator $q(\cdot)$ and a unary discriminator $D(\cdot) : \mathcal{X} \to \mathbb{R}$ which operates on single samples, with the loss functions[1] for the two given by

$$\mathcal{L}_D = \mathbb{E}_p\Big[f_1(D(x))\Big] + \mathbb{E}_q\Big[f_2(D(x))\Big], \tag{3.1a}$$

$$\mathcal{L}_G = \mathbb{E}_p\Big[g_1(D(x))\Big] + \mathbb{E}_q\Big[g_2(D(x))\Big]. \tag{3.1b}$$

where $f_1, f_2, g_1, g_2 : \mathbb{R} \to \mathbb{R}$ are activation functions applied to the discriminator. The original GAN by Goodfellow et al. [78], which we refer to as the standard GAN (SGAN for short), has $f_1(t) = -g_1(t) = -\log(t)$, $f_2(t) = -g_2(t) = -\log(1-t)$ for the saturating version and $f_1(t) = g_2(t) = -\log(t)$, $f_2(t) = g_1(t) = -\log(1-t)$ for the non-saturating one.

Unary objectives as linear forms. An expectation $\mathbb{E}_p\big[f(D(x))\big] = \int f(D(x))p(x)\,dx$ can be viewed as a linear form in the function space: $\big\langle\, a_D^f\,,\, p \,\big\rangle$, where a_D^f and p are the function space vectors corresponding to functions $f(D(\cdot))$ and $p(\cdot)$ respectively. Using the function space notation we can rewrite the losses

$$\mathcal{L}_D = \big\langle\, a_D^{f_1}\,,\, p \,\big\rangle + \big\langle\, a_D^{f_2}\,,\, q \,\big\rangle, \tag{3.2a}$$

[1]Throughout this Chapter, we assume that all loss functions are to be minimized.

$$\mathcal{L}_G = \left\langle\, a_D^{g_1}\,,\, p \,\right\rangle + \left\langle\, a_D^{g_2}\,,\, q \,\right\rangle. \tag{3.2b}$$

Set of probability density functions. Note that p and q must define valid density functions. We define $\mathcal{P}(\mathcal{X})$ as the set of probability density functions which belong to $\mathcal{F}(\mathcal{X})$. Formally, we define $\mathcal{P}(\mathcal{X})$ as $\mathcal{P}(\mathcal{X}) = \left\{p \in \mathcal{F}(\mathcal{X}) \middle| \left\langle\, p\,,\, e \,\right\rangle = 1;\ p(x) \geq 0\ \forall\, x \in \mathcal{X}\right\}$, where $e \in \mathcal{F}(\mathcal{X})$ is a function space vector $e(x) \equiv 1$ having the constant value of 1 on all of its "positions".

Pairwise discriminators. The standard setup of GANs can be extended by replacing the unary discriminator $D(\cdot)$ with a pairwise discriminator $D(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which operates on pairs of samples [114, 144, 245]. In this Chapter we define a pairwise discriminator as which classifies the pairs of samples into two classes: same distribution pairs and different distribution pairs:

$$M^{+}_{p,q}(x,y) = \tfrac{1}{2}(p(x)\cdot p(y) + q(x)\cdot q(y)); \tag{3.3a}$$

$$M^{-}_{p,q}(x,y) = \tfrac{1}{2}(p(x)\cdot q(y) + q(x)\cdot p(y)). \tag{3.3b}$$

With a pairwise discriminator, we define a modified game for GANs:

$$\begin{aligned}\mathcal{L}_D =& \mathbb{E}_{p\times p}\Big[f_1(D(x,y))\Big] + \mathbb{E}_{q\times q}\Big[f_1(D(x,y))\Big] \\ &+ \mathbb{E}_{p\times q}\Big[f_2(D(x,y))\Big] + \mathbb{E}_{q\times p}\Big[f_2(D(x,y))\Big],\end{aligned} \tag{3.4a}$$

$$\begin{aligned}\mathcal{L}_G =& \mathbb{E}_{p\times p}\Big[g_1(D(x,y))\Big] + \mathbb{E}_{q\times q}\Big[g_1(D(x,y))\Big] \\ &+ \mathbb{E}_{p\times q}\Big[g_2(D(x,y))\Big] + \mathbb{E}_{q\times p}\Big[g_2(D(x,y))\Big].\end{aligned} \tag{3.4b}$$

Pairwise objectives as bilinear forms. An expectation $\mathbb{E}_{p,q}\big[f(D(x,y))\big] = \iint f(D(x,y))p(x)q(y)dxdy$ can be viewed as a bi-linear form in the function space: $\left\langle\, p\,,\, A_D^f q \,\right\rangle$, where A_D^f denotes a function-space linear operator corresponding to the function $f(D(\cdot,\cdot))$: $[A_D^f q](x) = \int f(D(x,y))q(y)\,dy$. Using the bi-linear forms we re-write the losses (3.4) in the form:

$$\mathcal{L}_D = \left\langle\, p\,,\, A_D^{f_1} p \,\right\rangle + \left\langle\, q\,,\, A_D^{f_1} q \,\right\rangle + \left\langle\, p\,,\, A_D^{f_2} q \,\right\rangle + \left\langle\, q\,,\, A_D^{f_2} p \,\right\rangle, \tag{3.5a}$$

$$\mathcal{L}_G = \left\langle\, p\,,\, A_D^{g_1} p \,\right\rangle + \left\langle\, q\,,\, A_D^{g_1} q \,\right\rangle + \left\langle\, p\,,\, A_D^{g_2} q \,\right\rangle + \left\langle\, q\,,\, A_D^{g_2} p \,\right\rangle. \tag{3.5b}$$

In order to simplify analysis in the following sections, we assume that $D(x, y)$ is a symmetric function ($D(x, y) = D(y, x)$) corresponding to a self-adjoint operator: $(A_D^f)^T = A_D^f$. Note that all results below hold for non-symmetric discriminators, since objectives (3.5) are invariant to symmetrization of operators $A \to \frac{1}{2}(A + A^T)$.

3.4 How To Preserve The Alignment?

Unary discriminators destroy the alignment. Consider the generator loss for a unary GAN (3.2). Suppose that at some moment the generator has been aligned with the target distribution: $q^* = p$. With the subsequent update, q receives the gradient signal $\nabla_q \mathcal{L}_G = a_D^{g_2}$. Below we show that unless $D(x)$ is constant in the support of p, the discriminator will drive q away from p and destroy the alignment.

We consider an infinitesimal perturbation $q' = q^* + \varepsilon$. Since q' must be a valid density function, ε must satisfy: **(i)** $\left\langle\, \varepsilon\,,\, e \,\right\rangle = 0$; **(ii)** $p(x) + \varepsilon(x) \geq 0 \;\forall\, x$. The first-order change of the generator loss (3.2) corresponding to the perturbation ε is given by: $\mathcal{L}_G(q^* + \varepsilon, D) - \mathcal{L}_G(q^*, D) \approx \left\langle\, a_D^{g_2}\,,\, \varepsilon \,\right\rangle$. The generator is stationary at q^* iff $\left\langle\, a_D^{g_2}\,,\, \varepsilon \,\right\rangle = 0,\; \forall\, \varepsilon : \left\langle\, \varepsilon\,,\, e \,\right\rangle = 0,\; p(x) + \varepsilon(x) \geq 0 \;\forall\, x$. This is only possible when $g_2(D(x))$ is constant in the support of p.

This observation implies that the generator can not converge unless the discriminator $D(\cdot)$ converges to the equilibrium position.

Pairwise discriminators preserve the alignment. We find that there is a family of generator objectives (3.5) with pairwise discriminators that prevents the discriminator from destroying the alignment, meaning: $\nabla_q \mathcal{L}_G(q, D)\big|_{q=p} = 0,\; \forall D$. Indeed, in order to satisfy $\nabla_q \mathcal{L}_G(q, D)\big|_{q=p} = 2(A_D^{g_1} + A_D^{g_2})p = 0$ it is sufficient to choose $g_2(x) = -g_1(x) \Rightarrow A_D^{g_2} = -A_D^{g_1}$. We define a function $g(x) = g_1(x) = -g_2(x)$

and consider the following instance of the generator loss (3.5b):

$$\mathcal{L}_G = \big\langle\, p - q\,,\; A_D^g(p - q) \,\big\rangle \tag{3.6}$$

3.5 PiarGAN

In this section, we first propose PairGAN, a formulation of GANs with the generator loss of the form (3.6). Then, in Section 3.5.1, for specific choices of f_1, f_2 and g, we provide a theoretical insight similar to that in Goodfellow et al. [78] to show that our approach in a specific form also minimizes a meaningful divergence metric. In Section 3.5.2, through evaluating the sufficient condition for local convergence, we introduce the notion of sufficient discriminators, which we analyze in details in Sections 3.5.3 and 3.5.4.

PairGAN. General formulation of PairGAN loss functions are described by a two player game:

$$\begin{aligned} L_{\mathcal{D}}(D, q) =& \mathbb{E}_{p\times p}\Big[f_1(D(x,y))\Big] + \mathbb{E}_{q\times q}\Big[f_1(D(x,y))\Big] \\ &+ \mathbb{E}_{p\times q}\Big[f_2(D(x,y))\Big] + \mathbb{E}_{q\times p}\Big[f_2(D(x,y))\Big], \end{aligned} \tag{3.7a}$$

$$\begin{aligned} L_{\mathcal{G}}(D, q) =& \mathbb{E}_{p\times p}\Big[g(D(x,y))\Big] + \mathbb{E}_{q\times q}\Big[g(D(x,y))\Big] \\ &- \mathbb{E}_{p\times q}\Big[g(D(x,y))\Big] - \mathbb{E}_{q\times p}\Big[g(D(x,y))\Big]. \end{aligned} \tag{3.7b}$$

PairGAN-Z. We also consider a zero-sum game for loss (3.6). We call the corresponding formulation PairGAN-Z:

$$L_{\mathcal{D}}(D, q) = - L_{\mathcal{G}}(D, q) \tag{3.8a}$$

$$\begin{aligned} L_{\mathcal{G}}(D, q) =& \mathbb{E}_{p\times p}\Big[g(D(x,y))\Big] + \mathbb{E}_{q\times q}\Big[g(D(x,y))\Big] \\ &- \mathbb{E}_{p\times q}\Big[g(D(x,y))\Big] - \mathbb{E}_{q\times p}\Big[g(D(x,y))\Big]. \end{aligned} \tag{3.8b}$$

3.5.1 Divergence Minimization

Consider the following choice of functions f_1, f_2 and g for PairGAN: $f_1(t) = -\log(t)$, $f_2(t) = -\log(1-t)$, $g(t) = \log(t)$. These loss functions are natural choice for a probabilistic discriminator. In this setup, we interpret the output of a pairwise discriminator $D(\cdot, \cdot)$ as the estimated probability of a pair being sampled from the same distributions. We refer to this particular PairGAN as PairSGAN (pair standard GAN). We show that PairSGAN setup corresponds to meaningful divergence metric.

Proposition 3.5.1. *Let $\widehat{L}(q)$, denote the value of the generator loss for the optimal discriminator $\widehat{L}_{\mathcal{G}}(q) = L_{\mathcal{G}}(D^*(q), q),\ D^*(q) = \operatorname{argmin}_{D \in \mathcal{D}} L_{\mathcal{D}}(D, q)$. $\widehat{L}_{\mathcal{G}}$ is equivalent to a divergence between the distributions $M^+_{p,q}$ and $M_{p,q} = \frac{1}{2}(M^+_{p,q} + M^-_{p,q})$:*

$$\widehat{L}_{\mathcal{G}}(q) = 4 \cdot \big(\mathrm{KL}(M^+_{p,q} \| M_{p,q}) + \mathrm{KL}(M_{p,q} \| M^+_{p,q})\big),$$

*Consequently: **(i)** $\widehat{L}_{\mathcal{G}}(q) \geq 0$; **(ii)** $\widehat{L}_{\mathcal{G}}(q) = 0$ iff $q = p$.*

Consider a particular instance of the PairGAN-Z game (3.8) corresponding to: $g(t) = \log(t)$.

We define a family of probabilistic discriminators whose values are separated from zero:

$$\mathcal{D}_{[\varepsilon,1]} = \{D(\cdot, \cdot) \mid D(x, y) \in [\varepsilon, 1]\ \forall\, (x, y)\},$$

where $\varepsilon \in (0, 1)$.

Then, the generator loss evaluated at the optimal PairGAN-Z discriminator is

$$\widehat{L}^Z_{\mathcal{G}}(q) = L_{\mathcal{G}}(D^*_\varepsilon(q), q),\ D^*_\varepsilon(q) = \operatorname*{argmax}_{D \in \mathcal{D}_{[\varepsilon,1]}} L_{\mathcal{G}}(D, q).$$

Proposition 3.5.2. *$\widehat{L}^Z_{\mathcal{G}}$ is equivalent to the total variation distance between the mixture distributions $M^+_{p,q}$ and $M^-_{p,q}$:*

$$\widehat{L}^Z_{\mathcal{G}}(q) = -\log(\varepsilon) \cdot \delta_{TV}(M^+_{p,q} \| M^-_{p,q}).$$

Consequently: ***(i)*** $\widehat{L}^{Z}_{\mathcal{G}}(q) \geq 0 \; \forall\, q$; ***(ii)*** $\widehat{L}^{Z}_{\mathcal{G}}(q) = 0 \iff q = p$.

3.5.2 Local Convergence Of Generator

We note that in game (3.7), since the generator loss is designed to preserve alignment once achieved, we only require the generator to reach alignment but do not require the discriminator to converge to a specific position. Thus, the goal of our convergence analysis is to identify the set of discriminators which allow the generator to converge.

Let $D_\psi(\cdot,\cdot)$ and $q(\cdot;\theta)$ be parametric discriminator and generator parameterized by vectors $\psi \in \mathbb{R}^m$ and $\theta \in \mathbb{R}^n$ respectively. We consider the realizable setup, that is we assume that there exists θ^* such that: $q(\cdot;\theta^*) = p(\cdot)$. Generally, a parametrization may permit different instances of parameters to define the same distribution. Hence, we consider a reparametrization manifold [168]: $\mathcal{M}_G = \{\theta | q(\cdot;\theta) = p(\cdot)\}$. In our analysis below, we assume that there is an ϵ-ball $B_\epsilon(\theta^*)$ around $\theta^* \in \mathcal{M}_G$ such that $\mathcal{M}_G \cap B_\epsilon(\theta^*)$ defines a $\mathcal{C}^1$-manifold. We denote the tangent space of the manifold $\mathcal{M}_G$ at θ^* by $\mathcal{T}_{\theta^*}\mathcal{M}_G$.

Recall from Section 3.4 that θ^* is a stationary generator for any discriminator ψ. Similar to Mescheder et al. [168] we analyze the local convergence by examining the eigenvalues of the Hessian of the loss (3.7b) w.r.t θ at θ^*. We denote this Hessian by $H(\theta^*;\psi) = \nabla^2_{\theta\theta} L_{\mathcal{G}}(D_\psi, q(\cdot;\theta))|_{\theta=\theta^*}$. In Appendix A.3 we show that the Hessian is given by

$$H(\theta^*;\psi) = 2 \iint \Big[g(D_\psi(x,y)) \cdot \big([\nabla_\theta q(x;\theta^*)][\nabla_\theta q(y;\theta^*)]^T \big) \Big] dx\, dy. \tag{3.9}$$

The following proposition provides a sufficient condition for local convergence of the generator.

Proposition 3.5.3. *Suppose that $\theta^* \in \mathcal{M}_G$ and a pair (ψ_0, θ^*) satisfies:*

$$u^T [H(\theta^*;\psi_0)] u > 0 \quad \forall u \notin \mathcal{T}_{\theta^*}\mathcal{M}_G. \tag{3.10}$$

Then, with fixed $\psi = \psi_0$, gradient descent w.r.t. θ for (3.7b) *converges to $\mathcal{M}_G$ in*

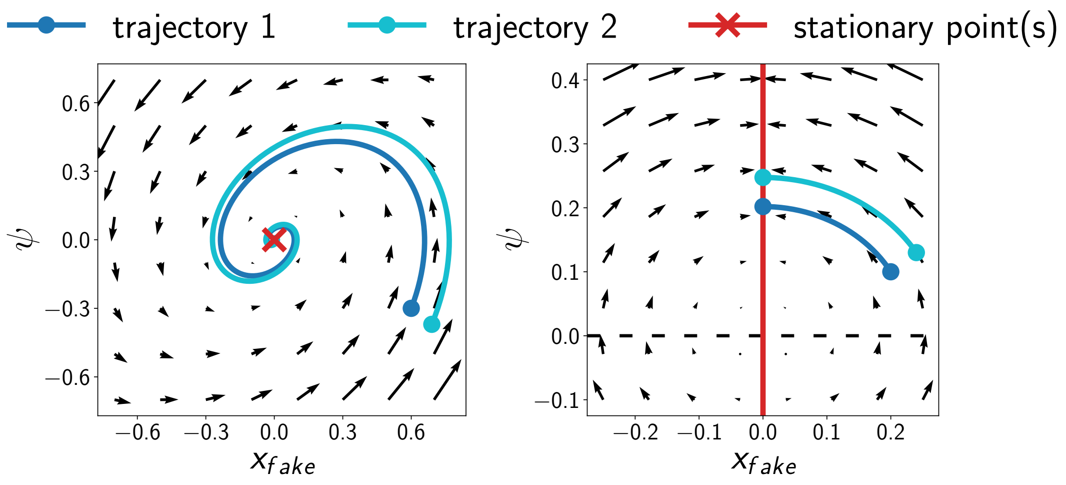


Figure 3-3: Visualization of convergence points. **Left:** SGAN with gradient penalty. **Right:** PairGAN

a neighborhood of θ^* provided a small enough learning rate. Moreover, the rate of convergence is at least linear.

Proposition 3.5.3 states that a discriminator ψ_0 satisfying condition (3.10) allows the generator to converge. While, the convergence guarantee is only established for training the generator with a fixed discriminator, this result still holds if we allow ψ to vary within a set. Indeed, from Proposition 3.5.3 it follows that θ converges to $\mathcal{M}_G$, given that ψ remains in the set of the discriminators satisfying (3.10). Note that this set includes all discriminator ψ in a neighborhood of ψ_0, since $u^T[H(\theta^*, \psi)]u$ is continuous at ψ_0 for any u.

Figure 3-3 contrasts the convergence properties for GANs with unary discriminators and PairGAN on a toy example identical to that described in Section 3.1. Left panel of Figure 3-3 shows two trajectories for SGAN with gradient penalties [168]. Both trajectories converge to the only stationary point. On the contrary, for PairGAN (Figure 3-3, right), two trajectories initialized at different points both achieve the alignment $x_{\text{fake}} = 0$ but converge to different positions of discriminator ψ. In this example, the discriminators corresponding to $\psi > 0$ satisfy (3.10) and define the gradient vector field pointing towards the line $x_{\text{fake}} = 0$. We note that the discriminator updates tend to keep ψ positive.

3.5.3 Sufficient Discriminators

To characterize the set of discriminators satisfying condition (3.10), we build intuition from the function space perspective.

We consider a perturbed value of the parameters of the generator $\theta' = \theta^* + u$, where $u \in \mathbb{R}^n$ is an infinitesimal perturbation vector. The corresponding first-order perturbation of the generated distribution q can be expressed via Taylor expansion: $\varepsilon_u(x) = u^T[\nabla_\theta q(x;\theta^*)]$. Note that $\varepsilon_u(.)$ is a linear combination of the derivatives w.r.t. to individual parameters $\theta_1, \dots, \theta_n$: $\varepsilon_u(x) = \sum_{i=1}^n u_i \alpha_i(x)$, $\alpha_i(x) = \frac{\partial}{\partial \theta_i} q(x;\theta^*)$. Thus, the set of all $\varepsilon_u(x)$ defines a finite-dimensional subspace of the function space. We denote this subspace by $W_q(\theta^*)$: $W_q(\theta^*) := \left\{\varepsilon(\cdot) | \varepsilon(x) = \sum_{i=1}^n u_i \alpha_i(x),\ u \in \mathbb{R}^n \right\}$. Note that $\dim(W_q(\theta^*)) = n - \dim(\mathcal{M}_G)$, since $\varepsilon_u(x) \equiv 0 \Leftrightarrow u^T[\nabla_\theta q(x;\theta^*)] \equiv 0 \Leftrightarrow u \in \mathcal{T}_{\theta^*}\mathcal{M}_G$.

The expression in equation (3.10) can be rewritten in terms of the perturbation ε_u: $u^T[H(\theta^*;\psi)]u = \left\langle \varepsilon_u\,,\, A^g_{D_\psi}\varepsilon_u \right\rangle$. The following definition gives a function space reformulation of the condition (3.10).

Definition 3.5.4. We say that a self-adjoint operator A is <u>sufficient</u> for a parametric generator $q(\cdot;\theta)$ at θ^* if

$$\left\langle \varepsilon_u\,,\, A\varepsilon_u \right\rangle > 0, \quad \forall\, \varepsilon_u \in W_q(\theta^*),\ \varepsilon_u \neq 0. \tag{3.11}$$

We say that a discriminator D is <u>sufficient</u> for $q(\cdot;\theta)$ at θ^* if the corresponding operator A^g_D is sufficient for $q(\cdot;\theta)$ at θ^*.

This definition essentially means that a discriminator is sufficient for a particular aligned generator if every possible change that this generator can make only results in increasing the generator loss.

3.5.4 Minimally Sufficient Discriminators

In this section, we define a notion of minimally sufficient discriminators and provide constructive examples of such discriminators. Note that for the condition (3.11) to be satisfied, it is required that $W_q(\theta^*) \subseteq \mathrm{Im}(A)$.

Definition 3.5.5. We say that an operator A is minimally sufficient for $q(\cdot;\theta)$ at θ^* if

(i) A is sufficient for $q(\cdot;\theta)$ at θ^*;

(ii) for any sufficient operator $B : \mathrm{Im}(A) \subseteq \mathrm{Im}(B)$.

The following proposition provides constructive examples of minimally sufficient discriminators for any given parametric generator.

Proposition 3.5.6. *Let $g_1(x;\theta)$ and $g_2(x;\theta)$ denote the functions:*

$$g_1(x;\theta) = \nabla_\theta q(x;\theta) \quad g_2(x;\theta) = \nabla_\theta \log q(x;\theta).$$

The operators A_1^* and A_2^*:

$$A_i^*(x, x'; \theta^*) = [g_i(x;\theta^*)]^T [g_i(x;\theta^*)];$$

are minimally sufficient operators for $q(\cdot;\theta)$ at θ^*.

These operators define the following generator objectives (3.7b)*:*

$$L_i^*(\theta) = \left\| \mathbb{E}_{p(x)} \Big[g_i(x;\theta) \Big] - \mathbb{E}_{q(x;\theta)} \Big[g_i(x;\theta) \Big] \right\|^2. \tag{3.12}$$

Discussion of Proposition 3.5.6. Operators A_i^* correspond to discriminators defined through the gradients of the density/log-density of a parametric generator $q(x;\theta)$. In other words, these examples show that given a parametric generator one can construct a minimally sufficient discriminator using the gradients $\nabla_\theta q(x;\theta)/\nabla_\theta \log q(x;\theta)$.

Consider the minimization problem for L_i^*

$$\min_\theta L_i^*(\theta),$$

which can be written as

$$\min_\theta \big\langle\, p(\cdot) - q(\,\cdot\,;\theta)\,,\, A_i^*(\,\cdot\,,\,\cdot\,;\theta)[p(\cdot) - q(\,\cdot\,;\theta)] \,\big\rangle.$$

This optimization problem defines a training procedure for the generator $q(x;\theta)$, where instead of training a discriminator, we utilize the operator $A_i^*(\,\cdot\,,\,\cdot\,;\theta)$ which depends on the generator $q(x;\theta)$ itself.

Below, we consider each of the losses $L_1^*(\theta)$ and $L_2^*(\theta)$ and show that they are connected to particular divergence metrics between the distributions $p(x)$ and $q(x;\theta)$.

Interpretation of $L_1^*(\theta)$

We re-write $L_1^*(\theta)$ as

$$L_1^*(\theta) = \Big\| \underbrace{\mathbb{E}_{q(x;\theta)}\Big[\nabla_\theta q(x;\theta)\Big] - \mathbb{E}_{p(x)}\Big[\nabla_\theta q(x;\theta)\Big]}_{I(\theta)} \Big\|^2,$$

where the function $I(\theta)$ can be expressed as

$$I(\theta) = \int \Big[q(x;\theta) - p(x)\Big] \cdot \nabla_\theta q(x;\theta)\, dx = \nabla_\theta \Bigg(\underbrace{\frac{1}{2} \int \Big[q(x;\theta) - p(x)\Big]^2 dx}_{L_{\mathrm{SQ}}(\theta)} \Bigg).$$

In the above, expression $L_{\mathrm{SQ}}(\theta)$ is a divergence defined by the square of the function-space distance $\|p - q\|$ between p and q.

The loss function $L_1^*(\theta)$ is connected to $L_{\mathrm{SQ}}(\theta)$:

$$L_1^*(\theta) = \Big\|\nabla_\theta L_{\mathrm{SQ}}(\theta)\Big\|^2.$$

Interpretation of $L_2^*(\theta)$

Using the fact that

$$\mathbb{E}_{q(x;\theta)}\Big[\nabla_\theta \log q(x;\theta)\Big] = 0,$$

we re-write the loss $L_2^*(\theta)$ as:

$$L_2^*(\theta) = \Big\|\mathbb{E}_{p(x)}\Big[\nabla_\theta \log q(x;\theta)\Big]\Big\|^2.$$

Next, we consider the KL-divergence

$$L_{\mathrm{KL}}(\theta) = \mathrm{KL}(p(x)\|q(x;\theta)) = \mathbb{E}_{p(x)}\Big[\log\frac{p(x)}{q(x;\theta)}\Big].$$

The gradient of $L_{\mathrm{KL}}(\theta)$ is given by

$$\nabla_\theta L_{\mathrm{KL}}(\theta) = -\mathbb{E}_{p(x)}\Big[\nabla_\theta \log q(x;\theta)\Big].$$

Similarly to L_1^*, L_2^* is connected to the KL-divergence:

$$L_2^*(\theta) = \big\|\nabla_\theta L_{\mathrm{KL}}(\theta)\big\|^2.$$

Relation to divergence minimization

Above, we have show that losses L_1^* and L_2^* are connected to the divergences L_{SQ} and L_{KL} respectively:

$$L_1^*(\theta) = \big\|\nabla_\theta L_{\mathrm{SQ}}(\theta)\big\|^2, \quad L_2^*(\theta) = \big\|\nabla_\theta L_{\mathrm{KL}}(\theta)\big\|^2.$$

Every divergence is non-negative and evaluates to zero iff $\theta = \theta^*$: $q(\,\cdot\,;\theta^*) = p(\cdot)$. Thus, θ^* is the unique global minimum of both L_{SQ} and L_{KL}[2].

We view the minimization of the losses L_1^* and L_2^* as a relaxation of the divergence minimization problem. Each of L_i^* reaches its minimal value $L_i^*(\hat{\theta}) = 0$ iff $\hat{\theta}$ is a stationary point of the corresponding divergence. In general, a stationary point of $L_{\mathrm{SQ}}/L_{\mathrm{KL}}$ is not global optimum ($\hat{\theta} \neq \theta^*$) since both divergences can be non-convex functions of θ. However, near θ^*, minimization of L_i^* converges to θ^*. Indeed, we are interested in analyzing sufficient operators as they provide guarantees for local convergence for the generator (see Section 3.5.2). Propositions 3.5.3 and 3.5.6 imply that gradient descent for $L_i^*(\theta)$ is locally convergent to θ^*.

[2]We note that minimization of the KL-divergence corresponds to maximum likelihood training of the generative model $q(x;\theta)$.

3.5.5 Towards Global Convergence of PairGAN

Now, we note an interesting property of PairGAN-Z (3.8).

For simplicity, we consider the case of finite $\mathcal{X} = \{x_1, \ldots, x_k\}$. Let Δ^k denote the probability simplex in $\mathbb{R}^k$. Consider game (3.8) between a generator $q \in \Delta^k$ and a discriminator-operator $A \in \mathbb{R}^{k \times k}$ given a target distribution $p \in \Delta^k$. Suppose we initialize q and A with $q_{(0)}$ and $A_{(0)}$ respectively. An iteration of alternating gradient descent is given by:

$$q_{(i+1)} = \operatorname{proj}_{\Delta^k}\Big(q_{(i)} - \alpha \cdot 2A_{(i)}[q_{(i)} - p]\Big), \tag{3.13a}$$

$$A_{(i+1)} = A_{(i)} + \beta \cdot [p - q_{(i+1)}][p - q_{(i+1)}]^T, \tag{3.13b}$$

where α and β are positive learning rates.

Suppose that at some iteration A is positive definite. Then, each step of the generator decreases the metric $\big\langle\, p - q\,,\, A(p-q)\,\big\rangle$ and drives q towards p. Furthermore, once A has become positive definite it is guaranteed to remain positive definite after the symmetric rank-1 update (3.13b). Thus once A becomes positive definite, q is guaranteed to converge.

We hypothesize that the observed effect opens the possibility to establish global convergence guarantees for PairGAN-Z (3.8). Informally, with each gradient update (3.13b), A becomes “more” positive definite. Then it remains to prove formally that with updates (3.13b) A reaches positive definite state from any starting point $A_{(0)}$. We leave further analysis of this problem for future work.

3.5.6 Aligning Multiple Distributions

In GANs the goal is to align the generated distribution q with a fixed real distribution p. In this section, we consider an extended setup for adversarial training, where our goal is to align multiple distributions $p_1, \ldots, p_N$ together. This setup is a simplified version of the distribution alignment problem arising in domain-invariant training [73, 146], where adversarial training is used to make the distributions of representations in multiple domains indistinguishable from one another.

We consider the following loss function for $p_1, \dots, p_N$:

$$\begin{aligned}\mathcal{L}(p_1, \dots, p_N | D) &= \sum_{i<j} \left\langle\, p_i - p_j \,,\, A_D^g(p_i - p_j) \,\right\rangle \\ &= (N-1)\sum_{i=1}^{N} \left\langle\, p_i \,,\, A_D^g p_i \,\right\rangle - \sum_{i \neq j} \left\langle\, p_i \,,\, A_D^g p_j \,\right\rangle. \end{aligned} \tag{3.14}$$

Proposition 3.5.7. *Suppose that* $p_1 = p_2 = \ldots = p_k$ *for some* $2 \leq k \leq N$*. Then* $\forall\, i, j \in \{1, \dots, k\}$*:*

$$\nabla_{p_i} \mathcal{L}(p_1, \dots, p_N | D) = \nabla_{p_j} \mathcal{L}(p_1, \dots, p_N | D) \quad \forall D.$$

Proposition 3.5.7 states that whenever all distribution in any given subset of $p_1, \dots p_N$ become mutually aligned they will receive the same gradient and the alignment within this subset will be preserved. The proof of the proposition is provided in Appendix A.6.

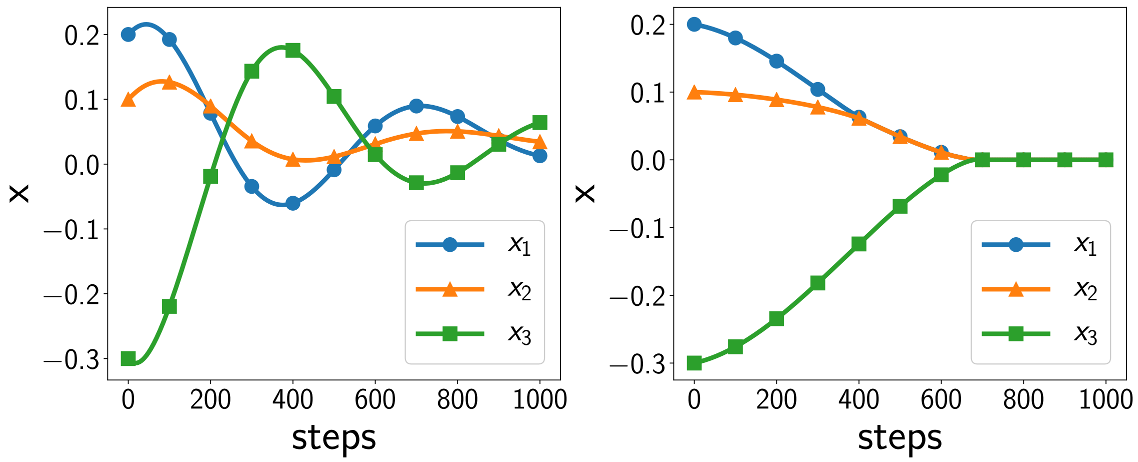


Figure 3-4: Demonstration of dynamics of multiple distribution alignment on a toy example.

Figure 3-4 provides a toy example demonstration for Proposition 3.5.7. In this example, the goal is to align three distributions $p_i(x) = \delta(x - x_i)$, $i \in \{1, 2, 3\}$. Figure 3-4 shows the trajectories of individual points x_i obtained as result of their interaction with a discriminator (domain-classifier). The left panel corresponds to a game with a unary discriminator $D(x)$. We observe that, when any pair of points becomes aligned the discriminator can still drive them apart. The right panel of Figure 3-4 shows the trajectories obtained by using objective (3.14) with a pairwise discriminator $D(x, y) = \psi \cdot |x - y|^{\gamma}$. We observe that with objective (3.14) the alignment is preserved for any pair of distributions. We provide the detailed specification of the toy example in Appendix A.7.

3.6 Connections To Other Pairwise Objectives

In this section we discuss the connections and differences between PairGAN and other GANs based on pairwise objectives. MMD-GAN [25, 144, 256] is built on Maximum Mean Discrepancy [MMD, 84] and shares the same function space view as PairGAN generator objective. From the point of view of (3.6), MMD-GAN operator A specifies a positive definite kernel: $A(x, y) = k(x, y)$. In practice, this kernel is parameterized by a neural network $A(x, y) = k(f(x), f(y))$, where a mapping f is defined by a deep network and a fixed kernel k is applied to embeddings produced by f. Clearly, alignment stability identified in Section 3.4 holds for both PairGAN and MMD-GAN.

PairGAN does not, however, require the discriminator to define a positive definite kernel, only a sufficient discriminator (see (3.11)). The corresponding operator needs to be positive definite in the subspace $W_q(\theta^*)$ defined by the generator. In this sense, PairGAN can be viewed as a parametric version of MMD-GAN. In contrast, MMD-GAN requires the operator (kernel) to be positive definite in the entire function space $\mathcal{F}(\mathcal{X})$. The use of kernels in PairGAN is straightforward but not necessary.

Suppose that in the minimization problem for loss (3.6) the generator q is not restricted to a parametric family and can define any density function in $\mathcal{P}(\mathcal{X})$. If we require for any infinitesimal perturbation of the aligned distribution $q' = p + \varepsilon$ to be "detectable" by operator A, we obtain a non-parametric version of condition (3.11):

$$\big\langle \varepsilon \, , \, A\varepsilon \big\rangle > 0, \quad \forall \, \varepsilon : \varepsilon \neq 0, \; \big\langle \varepsilon \, , \, e \big\rangle = 0, \; p(x) + \varepsilon(x) \geq 0 \; \forall \, x, \tag{3.15}$$

where the last two conditions ensure that $q' = p + \varepsilon$ is a valid density.

Condition (3.15) is a relaxed version of the condition (3.11), since, for a valid parameterization of q, any $\varepsilon \in W_q(\theta^*)$ defines a valid density q'. Put another way, we can contrast the difference between non-parametric (MMD-GAN) and parametric (PairGAN) cases.

- **Non-parametric:** the perturbation of q^* is not restricted by a parameterization; a sufficient operator A is required to be positive definite in the infinite-dimensional space (i.e. A is a kernel).

- **Parametric:** the perturbation of q^* is restricted by a parameterization; a sufficient operator A is required to be positive definite only in finite-dimensional subspace $W_q(\theta^*)$.

Thus, the notion of sufficient PairGAN discriminator naturally extends the notion of positive definite kernel for a parametric case. Proposition 3.5.6 shows that the set of sufficient discriminators is wider than the set of positive definite kernels.

In the Relativistic GAN [114], the discriminator is defined as a real-valued function $D_{\mathrm{R}}(x, y) = C(x) - C(y)$ which outputs relative "realness" score between samples x and y. The game between the generator and discriminator is defined by the objectives of the following form:

$$\begin{aligned}\mathcal{L}_D &= \mathbb{E}_{p\times q}\big[f_1(D_{\mathrm{R}}(x, y))\big] + \mathbb{E}_{q\times p}\big[f_2(D_{\mathrm{R}}(x, y))\big],\\ \mathcal{L}_G &= \mathbb{E}_{p\times q}\big[g_1(D_{\mathrm{R}}(x, y))\big] + \mathbb{E}_{q\times p}\big[g_2(D_{\mathrm{R}}(x, y))\big].\end{aligned}$$

A major difference between the above objectives and PairGAN's is the absence of sampled pairs from the same distribution. PairGAN makes use of all combinations of random pairs of samples from both generated and real distributions.

XORGAN [245] uses a probabilistic pair discriminator $\overline{D}(x, y)$ to quantify whether samples x, y come from different distributions. The pair discriminator is defined based on the optimal unary discriminator: $\overline{D}^*(x, y) = D^*(x)\cdot(1-D^*(y)) + (1-D^*(x))\cdot D^*(y)$. Tsirigotis et al. [245] maintain this functional form for the pair discriminator and

Table 3.1: Comparison of GAN FID curve statistics on CIFAR-10. We report min, max, mean, std over checkpoints at steps 20k, 30k, ...

Loss	Min	Max	Mean	SD
SGAN	**29.49**	41.20	**31.56**	2.24
RSGAN	31.27	**40.26**	34.12	**1.68**
RaSGAN	30.80	41.27	34.12	2.28
LSGAN	31.56	44.67	33.74	2.48
RaLSGAN	32.53	44.63	35.52	2.42
WGAN-GP	33.43	48.78	35.00	2.71
MMD-GAN	44.84	64.54	51.19	4.92
XORGAN	32.62	52.17	37.40	4.24
PairSGAN	30.42	42.85	33.81	2.00

directly estimate the unary part using $\mathcal{L}_D = \mathbb{E}_p[-\log(D(x))] + \mathbb{E}_q[-\log(1-D(x))]$. Compared to PairGAN, both relativistic GAN and XORGAN loss functions differ from (3.6) and do not ensure alignment stability. Moreover, both formulations use specific restricted families of pairwise discriminators.

3.7 Experiments

In this section we report the experimental results which evaluate the practical effectiveness of PairGAN. Our experiments are aimed at answering the following questions about the proposed model.

1. Do popular GAN models suffer from the alignment instability and can PairGAN address the issue in a practical setting? (Section 3.7.1)
2. Does PairGAN deliver better stability along the full training trajectory (rather than in closer to convergence situations only)? (Section 3.7.2)
3. Do the stability benefits of PairGAN translate into improved image generation quality? (Section 3.7.2)

Baseline GAN models Here we provide a list of abbreviations of GAN model names to use in the following sections. Unary GANs: non-saturating standard GAN [SGAN, 78], LSGAN [160], HingeGAN [171], SpectralGAN [171], WGAN-GP

[87]. Pairwise GANs: Relativistic GAN [114] and its variants (RSGAN, RaSGAN, RaLSGAN, RSGAN-GP, RaSGAN-GP), MMD-GAN [144], XORGAN [245].

3.7.1 Fixed Generator Matching

Our goal is to check whether the alignment instability indeed occurs in GANs. However, to assess this phenomenon in a realistic practical setting comes with definite challenges:

- Strictly speaking alignment instability would occur around where the generator aligns well with the real data. For a generator, parameterized by a deep neural network, one cannot easily verify that a given parameterization allows $q_\theta \approx p$.
- There are no clearly established alignment/misalignment measures in generative modeling problems such as image generation.

Below we describe a controlled experiment on a deep architecture and a complex target distribution which avoids these complications.

We consider a DCGAN generator [200]. The generator defines a distribution q_θ is parameterized by a vector $\theta \in \mathbb{R}^N$ whose elements are all trainable weights of the generator network listed in some fixed order. We pre-train DCGAN on CIFAR-10 [130] and extract the weights θ^* of the pre-trained generator. Now, we define the target distribution p in our controlled problem as $p \triangleq q_{\theta^*}$ and train another instance of the generator q_θ on the samples from q_{θ^*}. This problem setting allows us to test GAN models on close-to-real complex target distribution while ensuring the existence of the perfectly aligned solution q_{θ^*} in the chosen parametric family $\mathcal{Q} = \{q_\theta \mid \theta \in \mathbb{R}^N\}$. We propose to use the parameter space distance $\|\theta - \theta^*\|_2$ as a measure of misalignment between the distributions q_θ and q_{θ^*}. However, we need to take into account the flexibility of the generator and avoid situations where the target distribution can be represented by two distinct settings of weights. In order to overcome this problem, after pre-training we restrict the parameters of the trainable network θ to one-dimensional line $\theta(\alpha) = \theta^* + \alpha \cdot v$, where $\alpha \in \mathbb{R}$ is a single real valued parameter and $v \in \mathbb{R}^N$ is a random unit vector with $\|v\|_2 = 1$. In this parameterization, $\alpha = 0$ corresponds

Table 3.2: Comparison of FID at step 100k on CIFAR-10 without batch normalization in corresponding network. D: distriminator, G: generator.

Loss	D	G & D
SGAN	181.47	227.17
RSGAN	44.14	51.00
RaSGAN	47.63	54.28
PairSGAN	**40.13**	**43.24**

to the optimal solution $\theta(0) = \theta^*$ and $|\alpha| = \|\theta(\alpha) - \theta^*\|_2$ is a natural measure of misalignment.

Using the problem setting and generator parameterization described above, we evaluate different GAN formulations with unary and pairwise discriminators. Our considered formulations are PairSGAN, SGAN, RSGAN, and XORGAN. For each model we use mini-batches sampled from the pre-trained generator to compute the discriminator and generator losses. We train parameters in all layers of the discriminator and the single parameter α of the generator with SGD. We provide further details and used hyper-parameters for this experiment in Appendix A.8.1.

Figure 3-2 shows time-evolution of α along the course of training for all models. The obtained results agree with our analysis of alignment instability and follow the pattern observed in toy example in Section 3.1, Figure 3-1. The alignment stability guarantees of PairSGAN hold in the experiment and result in the trajectories of α that are stable in the vicinity of $\alpha^* = 0$. SGAN with unary discriminator and pairwise models without alignment stability guarantees (RGAN, XORGAN) repeatedly throw the generator off the aligned position and deliver oscillating trajectories.

3.7.2 Real World Datasets

Apart from ensuring alignment stability, we aim to find out that if PairGAN is actually a practical model for image generation. Here, we discuss our experiments conducted on CIFAR-10 [130] and CAT dataset [276]. For both datasets, we utilize DCGAN [200] architecture with specific implementation details discussed in Appendix A.8.2. For specific instances of PairGAN, we make the same choice as in previous sections, namely

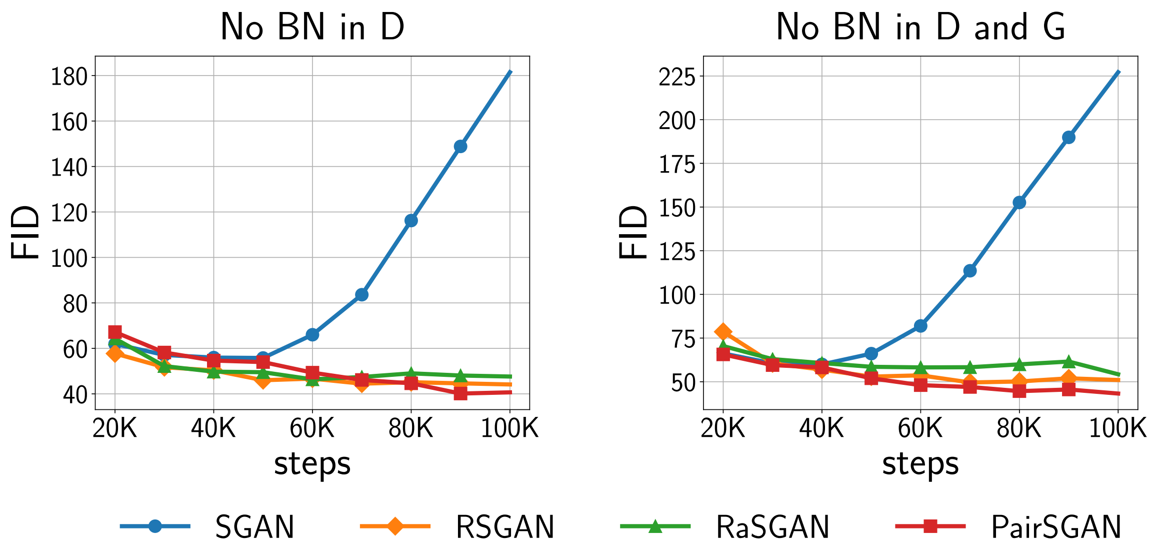


Figure 3-5: FID training curves on CIFAR-10 for DCGAN models without batch normalization (BN) layers in discriminator or both discriminator (D) and generator (G) networks.

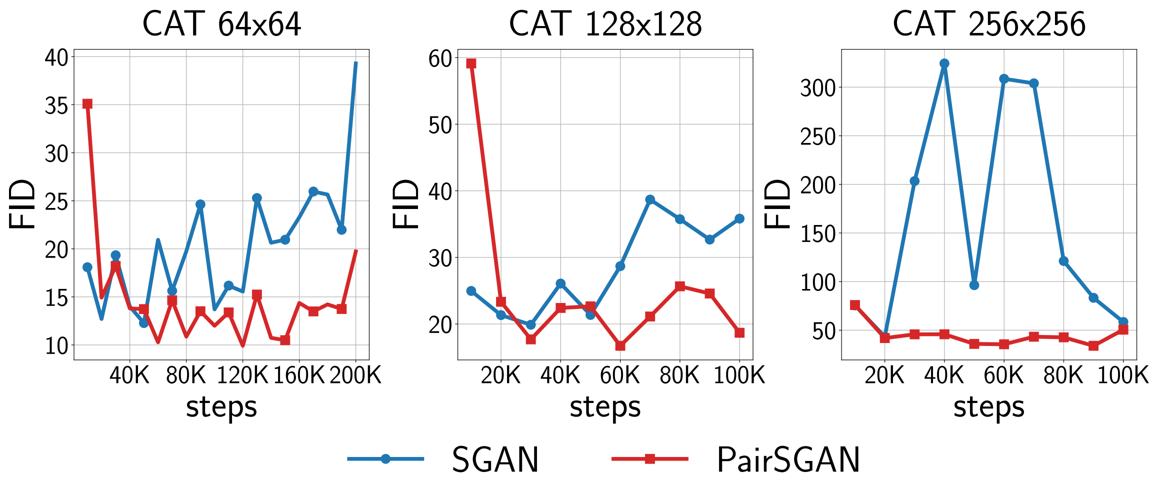


Figure 3-6: FID training curves for SGAN (blue) and PairSGAN (red) on CAT dataset for different resolutions.

PairSGAN. On CIFAR-10, We compare PairSGAN with some popular unary objectives (SGAN, LSGAN, WGAN-GP) and pairwise objectives of interests (RSGAN, RaSGAN, RaLSGAN, MMD-GAN and XORGAN). On CAT, since Jolicoeur-Martineau [114] utilizes the exact same architecture, we directly take their reported results as baselines (except for SGAN, which we replace with our fixed stable implementation[3]). Similar

[3]The baseline for SGAN provided in Jolicoeur-Martineau [114] uses a numerically unstable implementation of the cross-entropy loss

Table 3.3: Comparison of minimum, maximum, mean, and standard deviation of FID at steps 20k, 30k, ... on CAT dataset. '—' means model becomes stuck in the first few iterations. Baseline results denoted with (*) were extracted from Jolicoeur-Martineau [114], not independently run in our experiments.

64 × 64 images				
Loss	Min	Max	Mean	SD
SGAN	12.27	24.62	16.99	4.07
RSGAN*	19.03	42.05	32.16	7.01
RaSGAN*	15.38	33.11	20.53	5.68
LSGAN*	20.27	224.97	73.62	61.02
RaLSGAN*	11.97	19.29	15.61	2.55
HingeGAN*	17.60	50.94	32.23	14.44
RaHingeGAN*	14.62	27.31	20.29	3.96
RSGAN-GP*	16.41	22.34	18.20	1.82
RaSGAN-GP*	17.32	22	19.58	**1.81**
PairSGAN	**10.28**	**18.21**	**13.55**	2.24
128 × 128 images				
Loss	Min	Max	Mean	SD
SGAN	19.88	38.68	28.91	6.73
RaSGAN*	21.05	39.65	28.53	6.52
LSGAN*	19.03	51.36	30.28	10.16
RaLSGAN*	**15.85**	40.26	22.36	7.53
PairSGAN	16.72	**25.66**	**21.43**	**2.94**
256 × 256 images				
SGAN	43.30	324.38	171.42	108.47
RaSGAN*	**32.11**	102.76	56.64	21.03
SpectralSGAN*	54.08	90.43	64.92	12.00
LSGAN*	—	—	—	—
RaLSGAN*	35.21	299.52	70.44	86.01
WGAN-GP*	155.46	437.48	341.91	101.11
PairSGAN	33.94	**50.52**	**41.70**	**5.23**

to the quantitative evaluations presented in Jolicoeur-Martineau [114], we report the minimum, maximum, mean and standard deviation of FID [94] calculated across generator steps 20k, 30k, . . . for one run with a predefined seed. We run 500k steps for CIFAR-10 and 100k steps for CAT. Respective results can be found in Table 3.1 and Table 3.3.

On CIFAR-10, we notice that almost all methods perform similarly under the standard experiment with controls (e.g. batch normalization) that are designed for training unary GANs. To investigate the effects provided by PairGAN, we conduct more experiments in cleaner setting (less controls). In particular, we drop the batch normalization in discriminator or both generator and discriminator and we compare the performance of SGAN, RSGAN, RaSGAN and PairSGAN (all are variations of SGAN). Results of this experiments are summarized in Table 3.2 and Figure 3-5. We observe that although SGAN's performance in the standard setting is one of the best among the baselines, it does so by relying on aforementioned controls more, whereas PairSGAN can still generate good images in this harder setting. We think this shows the potential of PairGAN that once the controls specifically designed for it are in place, its theoretical benefits can translate into practical gains.

On CAT dataset, we change the parameterization of the pairwise discriminator used in the CIFAR-10 experiment (details in Appendix A.8.2) as we find this modification benefits training. As results shown in Table 3.3 and FID trajectory with SGAN and PairSGAN in Figure 3-6, we find that PairSGAN improves both performance and stability.

Chapter 4

Adversarial Support Alignment

4.1 Introduction

Learning tasks often involve estimating properties of distributions from samples or aligning such characteristics across domains. We can align full distributions (adversarial domain alignment), certain statistics (canonical correlation analysis), or the support of distributions (this paper). Much of the recent work has focused on full distributional alignment, for good reasons. In domain adaptation, motivated by theoretical results [12, 13], a series of papers [3, 72, 73, 132, 145, 191, 224, 246, 255, 277] seek to align distributions of representations between domains, and utilize a shared classifier on the aligned representation space.

Alignment in distributions implies alignment in supports. However, when there are additional objectives/constraints to satisfy, the minimizer for a distribution alignment objective does not necessarily minimize a support alignment objective. Example in Figure 4-1 demonstrates the qualitative distinction between two minimizers when distribution alignment is not achievable. The distribution alignment objective prefers to keep supports unaligned even if support alignment is achievable. Recent works [143, 238, 239, 262, 278] have demonstrated that a shift in label distributions between source and target leads to a characterizable performance drop when the representations are forced into a distribution alignment. The error bound in Johansson et al. [113] suggests aligning the supports of representations instead.

In this chapter, we focus on distribution support as the key characteristic to align. We introduce a support divergence to measure the support mismatch and algorithms to optimize such alignment. We also position our approach in the spectrum of other alignment methods. Our contributions are as follows (all proofs can be found in Appendix B.1):

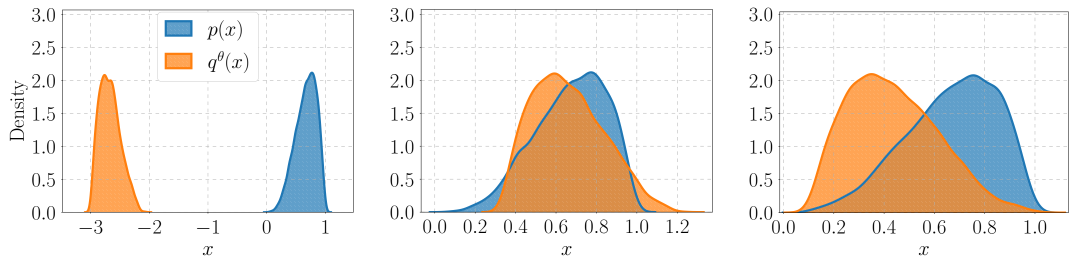
1. In Section 4.2.1, we measure the differences between supports of distributions. Building on the Hausdorff distance, we introduce a novel support divergence better suited for optimization, which we refer to as *symmetric support difference* (SSD) divergence.

2. In Section 4.2.2, we identify an important property of the discriminator trained for Jensen–Shannon divergence: support differences in the original space of interest are "preserved" as support differences in the one-dimensional discriminator output space.

3. In Section 4.3, we present our practical algorithm for support alignment, *Adversarial Support Alignment* (ASA). Essentially, based on the analysis presented in Section 4.2.2, our solution is to align supports in the discriminator 1D space, which is computationally efficient.

4. In Section 4.4, we place different notions of alignment – distribution alignment, relaxed distribution alignment and support alignment – within a coherent spectrum from the point of view of optimal transport, characterizing their relationships, both theoretically in terms of their objectives and practically in terms of their algorithms.

5. In Section 4.6, we demonstrate the effectiveness of support alignment in practice for domain adaptation setting. Compared to other alignment-based baselines, our proposed method is more robust against shifts in label distributions.



(a) Initialization
$\mathcal{D}_W(p, q^\theta) = 11.12$
$\mathcal{D}_\triangle(p, q^\theta) = 14.9$

(b) Distribution alignment
$\mathcal{D}_W(p, q^\theta) = 2 \cdot 10^{-3}$
$\mathcal{D}_\triangle(p, q^\theta) = 6 \cdot 10^{-4}$

(c) Support alignment
$\mathcal{D}_W(p, q^\theta) = 5 \cdot 10^{-2}$
$\mathcal{D}_\triangle(p, q^\theta) < 1 \cdot 10^{-6}$

Figure 4-1: Illustration of differences between the final configurations of distribution alignment and support alignment procedures. $p(x)$ is a fixed Beta distribution $p(x) = \mathrm{Beta}(x \,|\, 4, 2)$ with support $[0, 1]$; $q^\theta(x)$ is a "shifted" Beta distribution $q^\theta(x) = \mathrm{Beta}(x - \theta \,|\, 2, 4)$ parameterized by θ with support $[\theta, \theta + 1]$. Panel (a) shows the initial configuration with $\theta_{\text{init}} = -3$. Panel (b) shows the result by distribution alignment. Panel (c) shows the result by support alignment. We report Wasserstein distance $\mathcal{D}_W(p, q^\theta)$ (4.8) and SSD divergence $\mathcal{D}_\triangle(p, q^\theta)$ (4.1).

4.2 SSD divergence and support alignment

Notation. We consider an Euclidean space $\mathcal{X} = \mathbb{R}^n$ equipped with Borel sigma algebra $\mathcal{B}$ and a metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (e.g. Euclidean distance). Let $\mathcal{P}$ be the set of probability measures on $(\mathcal{X}, \mathcal{B})$. For $p \in \mathcal{P}$, the support of p is denoted by $\mathrm{supp}(p)$ and is defined as the smallest closed set $X \subseteq \mathcal{X}$ such that $p(X) = 1$. $f_\sharp p$ denotes the pushforward measure of p induced by a measurable mapping f. With a slight abuse of notation, we use $p(x)$ and $[f_\sharp p](t)$ to denote the densities of measures p and $f_\sharp p$ evaluated at x and t respectively, implicitly assuming that the measures are absolutely continuous. The distance between a point $x \in \mathcal{X}$ and a subset $Y \subseteq \mathcal{X}$ is defined as $d(x, Y) = \inf_{y \in Y} d(x, y)$. The symmetric difference of two sets A and B is defined as $A \triangle B = (A \setminus B) \cup (B \setminus A)$.

4.2.1 Difference between supports

To align the supports of distributions, we first need to evaluate how different they are. Similar to distribution divergences like Jensen–Shannon divergence, we introduce a notion of support divergence. A *support divergence*[1] between two distributions in $\mathcal{P}$ is a function $\mathcal{D}_S(\cdot,\cdot) : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$ satisfying: 1) $\mathcal{D}_S(p,q) \geq 0$ for all $p, q \in \mathcal{P}$; 2) $\mathcal{D}_S(p,q) = 0$ iff $\text{supp}(p) = \text{supp}(q)$.

While a distribution divergence is sensitive to both density and support differences, a support divergence only needs to detect mismatches in supports, which are subsets of the metric space $\mathcal{X}$. An example of a distance between subsets of a metric space is the Hausdorff distance: $d_H(X,Y) = \max\{\sup_{x\in X} d(x,Y), \sup_{y\in Y} d(y,X)\}$. Since it depends only on the greatest distance between a point and a set, minimizing this objective for alignment only provides signal to a single point. To make the optimization less sparse, we consider all points that violate the support alignment criterion and introduce *symmetric support difference* (SSD) divergence:

$$\mathcal{D}_\triangle(p,q) = \mathbb{E}_{x\sim p}\left[d(x, \text{supp}(q))\right] + \mathbb{E}_{x\sim q}\left[d(x, \text{supp}(p))\right]. \tag{4.1}$$

Proposition 4.2.1. *SSD divergence $\mathcal{D}_\triangle(p,q)$ is a support divergence.*

We note that our proposed SSD divergence is closely related to Chamfer distance/divergence (CD) [69, 180] and Relaxed Word Mover's Distance (RWMD) [133]. While both CD and RWMD are stated for discrete points (see Section 4.5 for further comments), SSD divergence is a general difference measure between arbitrary (discrete or continuous) distributions. This distinction, albeit small, is important in our theoretical analysis (Sections 4.2.2, 4.4.1).

4.2.2 Support Alignment in One-Dimensional Space

Goodfellow et al. [78] showed that the log-loss discriminator $f : \mathcal{X} \to [0,1]$, trained to distinguish samples from distributions p and q ($\sup_f \; \mathbb{E}_{x\sim p}\left[\log f(x)\right] + \mathbb{E}_{x\sim q}\left[\log(1 - f(x))\right]$) can be used to estimate the Jensen–Shannon divergence between

[1]It is not technically a divergence on the space of distributions, since $\mathcal{D}_S(p,q) = 0$ does not imply $p = q$.

p and q. The closed form maximizer f^* is

$$f^*(x) = \frac{p(x)}{p(x) + q(x)}, \qquad \forall x \in \operatorname{supp}(p) \cup \operatorname{supp}(q). \tag{4.2}$$

Note that for a point $x \notin \operatorname{supp}(p) \cup \operatorname{supp}(q)$ the value of $f^*(x)$ can be set to an arbitrary value in $[0, 1]$, since the log-loss does not depend on $f(x)$ for such x. The form of the optimal discriminator (4.2) gives rise to our main theorem below, which characterizes the ability of the log-loss discriminator to identify support misalignment.

Theorem 4.2.2. *Suppose distributions p and q have densities satisfying*

$$\frac{1}{C} < p(x) < C, \quad \forall\, x \in \operatorname{supp}(p); \qquad \frac{1}{C} < q(x) < C, \quad \forall\, x \in \operatorname{supp}(q). \tag{4.3}$$

Let f^* be the optimal discriminator (4.2). Then, $\mathcal{D}_\triangle(p, q) = 0$ if and only if $\mathcal{D}_\triangle({f^*}_\sharp p, {f^*}_\sharp q) = 0$.

The idea of the proof is to show that the extreme values (0 and 1) of $f^*(x)$ can only be attained in $x \in \operatorname{supp}(p) \triangle \operatorname{supp}(q)$. Assumption (4.3) guarantees that $f^*(x)$ cannot approach neither 0 nor 1 in the intersection of the supports $\operatorname{supp}(p) \cap \operatorname{supp}(q)$, i.e. the values $\{f^*(x) \mid x \in \operatorname{supp}(p) \cap \operatorname{supp}(q)\}$ are separated from the extreme values 0 and 1.

We conclude this section with two technical remarks on Theorem 4.2.2.

Remark 4.2.3. The result of Theorem 4.2.2 does not necessarily hold for other types of discriminators. For instance, the dual Wasserstein discriminator [9, 87] does not always highlight the support difference in the original space as a support difference in the discriminator output space. This observation is formaly stated in the following proposition.

Proposition 4.2.4. *Let $f^\star_W$ be the maximizer of $\sup_{f : \mathrm{L}(f) \le 1} \mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{x \sim q}[f(x)]$, where $\mathrm{L}(\cdot)$ is the Lipschitz constant. There exist p and q with $\operatorname{supp}(p) \neq \operatorname{supp}(q)$ but $\operatorname{supp}({f^\star_W}_\sharp p) = \operatorname{supp}({f^\star_W}_\sharp q)$.*

Recent works [52, 53] have proposed to perform optimal transport (OT)-based distribution alignment by reducing the OT problem (4.8) in the original, potentially high-dimensional space, to that between 1D distributions. Specifically, Deshpande et al. [52] consider the sliced Wasserstein distance [199]:

$$\mathcal{D}_{SW}(p, q) = \int_{\mathbb{S}^{n-1}} \mathcal{D}_W({f^\theta}_\sharp p, {f^\theta}_\sharp q)\, d\theta, \tag{4.4}$$

where $\mathbb{S}^{n-1} = \{\theta \in \mathbb{R}^n \mid \|\theta\| = 1\}$ is a unit sphere in $\mathbb{R}^n$, and f^θ is a 1D linear projection $f^\theta(x) = \langle\theta, x\rangle$. It is known that $\mathcal{D}_{SW}$ is a valid distribution divergence: for any $p \neq q$ there exists a linear slicing function f^θ , $\theta \in \mathbb{S}^{n-1}$ which identifies difference in the distributions, i.e. ${f^\theta}_\sharp p \neq {f^\theta}_\sharp q$ (Cramér–Wold theorem). By reducing the original OT problem to that in a 1D space, Wu et al. [260] and Deshpande et al. [53] develop efficient practical methods for distribution alignment based on fast algorithms for the 1D OT problem.

Unfortunately, the straight-forward extension of SSD divergence (4.1) to a 1D linearly sliced version does not provide a valid support divergence.

Proposition 4.2.5. *There exist two distributions p and q in $\mathcal{P}$, such that* $\operatorname{supp}(p) \neq \operatorname{supp}(q)$ *but* $\operatorname{supp}({f^\theta}_\sharp p) = \operatorname{supp}({f^\theta}_\sharp q)$, $\forall f^\theta(x) = \langle\theta, x\rangle$ *with* $\theta \in \mathbb{S}^{n-1}$.

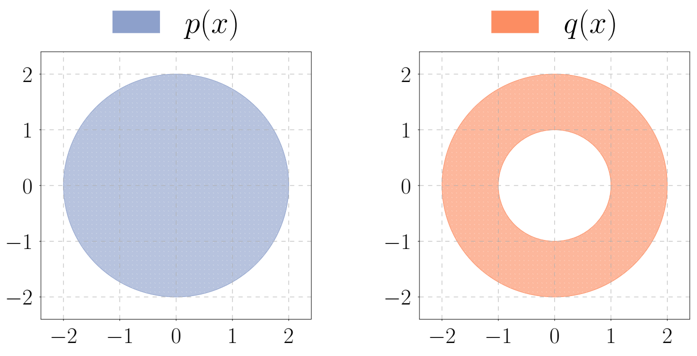


Figure 4-2: Visualization of example distributions for Proposition 4.2.5

Proof. Consider a 2-dimensional Euclidean space $\mathbb{R}^2$ and let $\operatorname{supp}(p) = \{(x, y) | x^2 + y^2 \leq 2\}$ and $\operatorname{supp}(q) = \{(x, y) | 1 \leq x^2 + y^2 \leq 2\}$. Then, $\forall f^\theta(x) = \langle\theta, x\rangle$ with $\theta \in \mathbb{S}^1$,

$$\operatorname{supp}({f^\theta}_\sharp p) = \operatorname{supp}({f^\theta}_\sharp q) = [-2, 2].$$

This counterexample is shown in Figure 4-2. □

Remark 4.2.6. In practice the discriminator is typically parameterized as $f(x) = \sigma(g(x))$, where $g : \mathcal{X} \to \mathbb{R}$ is realized by a deep neural network and $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid function. The optimization problem for g is

$$\inf_{g} \; \mathbb{E}_{x\sim p}\left[\log(1 + e^{-g(x)})\right] + \mathbb{E}_{x\sim q}\left[\log(1 + e^{g(x)})\right], \tag{4.5}$$

and the optimal solution is $g^*(x) = \log p(x) - \log q(x)$. Naturally the result of Theorem 4.2.2 holds for g^*, since $g^*(x) = \sigma^{-1}(f^*(x))$ and σ is a bijective mapping from $\mathbb{R} \cup \{-\infty, \infty\}$ to $[0, 1]$.

4.3 Adversarial Support Alignment

We consider distributions p and q parameterized by θ: p^θ, q^θ. The log-loss discriminator g optimized for (4.5) is parameterized by ψ: g^ψ. Our analysis in Section 4.2.2 already suggests an algorithm. Namely, we can optimize θ by minimizing $\mathcal{D}_\triangle({g^\psi}_\sharp p^\theta, {g^\psi}_\sharp q^\theta)$ while optimizing ψ by (4.5). This adversarial game is analogous to the setup of the existing distribution alignment algorithms[2].

In practice, rather than having direct access to p^θ, q^θ, which is unavailable, we are often given i.i.d. samples $\{x_i^p\}_{i=1}^N, \{x_i^q\}_{i=1}^M$. They form discrete distributions $\hat{p}^\theta(x) = \frac{1}{N}\sum_{i=1}^N \delta(x - x_i^p), \hat{q}^\theta(x) = \frac{1}{M}\sum_{i=1}^M \delta(x - x_i^q)$, and $[{g^\psi}_\sharp \hat{p}^\theta](t) = \frac{1}{N}\sum_{i=1}^N \delta(t - g^\psi(x_i^p)), [{g^\psi}_\sharp \hat{q}^\theta](t) = \frac{1}{M}\sum_{i=1}^M \delta(t - g^\psi(x_i^q))$. Since ${g^\psi}_\sharp \hat{p}^\theta$ and ${g^\psi}_\sharp \hat{q}^\theta$ are discrete distributions, they have supports $\{g^\psi(x_i^p)\}_{i=1}^N$ and $\{g^\psi(x_i^q)\}_{i=1}^M$ respectively. SSD divergence between discrete distributions ${g^\psi}_\sharp \hat{p}^\theta$ and ${g^\psi}_\sharp \hat{q}^\theta$ is

$$\begin{aligned}\mathcal{D}_\triangle({g^\psi}_\sharp \hat{p}^\theta, {g^\psi}_\sharp \hat{q}^\theta) =& \frac{1}{N}\sum_{i=1}^N d\big(g^\psi(x_i^p), \{g^\psi(x_j^q)\}_{j=1}^M\big) \\ &+ \frac{1}{M}\sum_{i=1}^M d\big(g^\psi(x_i^q), \{g^\psi(x_j^p)\}_{j=1}^N\big)\,. \end{aligned}\tag{4.6}$$

[2]Following existing adversarial distribution alignment methods, e.g. [78], we use single update of ψ per 1 update of θ. While theoretical analysis for both distribution alignment and support alignment (ours) assume optimal discriminators, training with single update of ψ is computationally cheap and effective.

Effect of mini-batch training. When training on large datasets, we need to rely on stochastic optimization with mini-batches. We denote the mini-batches (of same size, as in common practice) from p^θ and q^θ as $x^p = \{x_i^p\}_{i=1}^m$ and $x^q = \{x_i^q\}_{i=1}^m$ respectively. By minimizing $\mathcal{D}_\triangle(g^\psi(x^p), g^\psi(x^q))$, we only consider the mini-batch support distance rather than the population support distance (4.6). We observe that in practice the described algorithm brings the distributions to a state closer to distribution alignment rather than support alignment (see Appendix 4.6 for details). The problem is in the typically small batch size. The algorithm actually aims to enforce support alignment for all possible pairs of mini-batches, which is a much stricter constraint than population support alignment.

To address the issue mentioned above, without working with a much larger batch size, we create two "history buffers": h^p, storing the previous 1D discriminator outputs of (at most) n samples from p^θ, and a similar buffer h^q for q^θ. Specifically, $h = \{g^{\psi_{\text{old},i}}(x_{\text{old},i})\}_{i=1}^n$ stores the values of the previous n samples $x_{\text{old},i}$ mapped by their corresponding past "versions" of the discriminator $g^{\psi_{\text{old},i}}$. We minimize $\mathcal{D}_\triangle(v^p, v^q)$, where $v^p = \text{concat}(h^p, g^\psi(x^p))$, $v^q = \text{concat}(h^q, g^\psi(x^q))$:

$$\mathcal{D}_\triangle(v^p, v^q) = \frac{1}{n+m}\left(\sum_{i=1}^{n+m} d(v_i^p, v^q) + \sum_{j=1}^{n+m} d(v_j^q, v^p)\right). \tag{4.7}$$

Note that $\mathcal{D}_\triangle(\cdot,\cdot)$ between two sets of 1D samples can be efficiently calculated since $d(v_i^p, v^q)$ and $d(v_j^q, v^p)$ are simply 1-nearest neighbor distances in 1D. Moreover the history buffers store only the scalar values from the previous batches. These values are only considered in nearest neighbor assignment but do not directly provide gradient signal for optimization. Thus, the computation overhead of including a long history buffer is very light. We present our full algorithm, *Adversarial Support Alignment* (ASA), in Algorithm 1.

Algorithm 1 Adversarial Support Alignment (ASA). n (maximum history buffer size), we use $n = 1000$.

```
1: for number of training steps do
2:     Sample mini-batches $\{x_i^p\}_{i=1}^m \sim p^\theta$,  $\{x_i^q\}_{i=1}^m \sim q^\theta$.
3:     Perform optimization step on $\psi$ using stochastic gradient
           $\nabla_\psi \left( \frac{1}{m} \sum_{i=1}^{m} \Big[ \log(1 + \exp(-g^\psi(x_i^p))) + \log(1 + \exp(g^\psi(x_i^q))) \Big] \right)$.
4:     $v^p \leftarrow \text{concat}(h^p, \{g^\psi(x_i^p)\}_{i=1}^m)$,   $v^q \leftarrow \text{concat}(h^q, \{g^\psi(x_i^q)\}_{i=1}^m)$.
5:     $\pi^i_{p\to q} \leftarrow \text{argmin}_j\, d(v_i^p, v_j^q)$,   $\pi^j_{q\to p} \leftarrow \text{argmin}_i\, d(v_i^p, v_j^q)$.
6:     Perform optimization step on $\theta$ using stochastic gradient
           $\nabla_\theta \left( \frac{1}{n+m} \sum_{i=1}^{n+m} \Big[ d(v_i^p, v^q_{\pi^i_{p\to q}}) + d(v_i^q, v^p_{\pi^i_{q\to p}}) \Big] \right)$.
7:     UpdateHistory$(h^p, \{g^\psi(x_i^p)\}_{i=1}^m)$,   UpdateHistory$(h^q, \{g^\psi(x_i^q)\}_{i=1}^m)$.
8: end for
```

4.4 Spectrum of Notions of Alignment

In this section, we take a closer look into our work and different existing notions of alignment that have been proposed in the literature, especially their formulations from the optimal transport perspective. We show that our proposed support alignment framework is a limit of existing notions of alignment, both in terms of theory and algorithm, by increasing transportation assignment tolerance.

4.4.1 Theoretical connections

Distribution alignment. Wasserstein distance is a commonly used objective for distribution alignment. In our analysis, we focus on the Wasserstein-1 distance:

$$\mathcal{D}_W(p, q) = \inf_{\gamma \in \Gamma(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[d(x, y)], \tag{4.8}$$

where $\Gamma(p, q)$ is the set of all measures on $\mathcal{X} \times \mathcal{X}$ with marginals of p and q, respectively. The value of $\mathcal{D}_W(p, q)$ is the minimal transportation cost for transporting probability mass from p to q. The transportation cost is zero if and only if $p = q$, meaning the distributions are aligned.

Relaxed distribution alignment. Wu et al. [262] proposed a modified Wasserstein distance to achieve asymmetrically-relaxed distribution alignment, namely β-

admissible Wasserstein distance:

$$\mathcal{D}_W^{\beta}(p, q) = \inf_{\gamma \in \Gamma_\beta(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[d(x, y)], \tag{4.9}$$

where $\Gamma_\beta(p, q)$ is the set of all measures γ on $\mathcal{X} \times \mathcal{X}$ such that $\int \gamma(x, y)dy = p(x), \forall x$ and $\int \gamma(x, y)dx \leq (1 + \beta)q(y), \forall y$. With the relaxed marginal constraints, one could choose a transportation plan γ which transports probability mass from p to a modified distribution q' rather than the original distribution q as long as q' satisfies the constraint $q'(x) \leq (1+\beta)q(x), \forall x$. Therefore, $\mathcal{D}_W^{\beta}(p, q)$ is zero if and only if $p(x) \leq (1+\beta)q(x), \forall x$. In [262], β is normally set to a positive finite number to achieve the asymmetric-relaxation of distribution alignment, and it is shown that $\mathcal{D}_W^{0}(p, q) = \mathcal{D}_W(p, q)$. We can extend $\mathcal{D}_W^{\beta}(p, q)$ to a symmetric version, which we term β_1, β_2-admissible Wasserstein distance:

$$\mathcal{D}_W^{\beta_1,\beta_2}(p, q) = \mathcal{D}_W^{\beta_1}(p, q) + \mathcal{D}_W^{\beta_2}(q, p). \tag{4.10}$$

The aforementioned property of β-admissible Wasserstein distance implies that $\mathcal{D}_W^{\beta_1,\beta_2}(p, q) = 0$ if and only if $p(x) \leq (1 + \beta_1)q(x), \forall x$ and $q(x) \leq (1 + \beta_2)p(x), \forall x$, in which case we call p and q "(β_1, β_2)-aligned", with β_1 and β_2 controlling the transportation assignment tolerances.

Support alignment. The term $\mathbb{E}_p[d(x, \text{supp}(q))]$ in (4.1) represents the average distance from samples in p to the support of q. From the optimal transport perspective, this value is the minimal transportation cost of transporting the probability mass of p into the support of q. We show that SSD divergence can be considered as a transportation cost in the limit of infinite assignment tolerance.

Proposition 4.4.1. $\mathcal{D}_W^{\infty,\infty}(p, q) \coloneqq \lim_{\beta_1,\beta_2\to\infty} \mathcal{D}_W^{\beta_1,\beta_2}(p, q) = \mathcal{D}_\triangle(p, q)$.

We now have completed the spectrum of alignment objectives defined within the optimal transport framework. The following proposition establishes the relationship within the spectrum.

Proposition 4.4.2. *Let p and q be two distributions in $\mathcal{P}$. Then,*

1. $\mathcal{D}_W(p, q) = 0$ *implies* $\mathcal{D}_W^{\beta_1,\beta_2}(p, q) = 0$ *for all finite* $\beta_1, \beta_2 > 0$.

2. $\mathcal{D}_W^{\beta_1,\beta_2}(p,q)=0$ *for some finite* $\beta_1,\beta_2>0$ *implies* $\mathcal{D}_\triangle(p,q)=0$*.*

3. The converse of statements 1 and 2 are false.

In addition to the result presented in Theorem 4.2.2, we can show that the log-loss discriminator can also "preserve" the existing notions of alignment.

Proposition 4.4.3. *Let* f^* *be the optimal discriminator (4.2) for given distributions* p *and* q*. Then,*

1. $\mathcal{D}_W(p,q)=0$ *iff* $\mathcal{D}_W(f^*{}_\sharp p, f^*{}_\sharp q)=0$*;*

2. $\mathcal{D}_W^{\beta_1,\beta_2}(p,q)=0$ *iff* $\mathcal{D}_W^{\beta_1,\beta_2}(f^*{}_\sharp p, f^*{}_\sharp q)=0$*.*

The proof of this result, provided in Appendix B.1.7, is based on the following property of pushforward distributions $f^*{}_\sharp p$ and $f^*{}_\sharp q$.

Proposition 4.4.4. *Let* f^* *be the optimal log-loss discriminator (4.2) between* p *and* q*. Then,*

$$\frac{[f^*{}_\sharp p](t)}{[f^*{}_\sharp p](t)+[f^*{}_\sharp q](t)}=t, \qquad \forall\, t\in \operatorname{supp}(f^*{}_\sharp p)\cup\operatorname{supp}(f^*{}_\sharp q). \tag{4.11}$$

Intuitively, this proposition states the following. If for some $x\in\mathcal{X}$ we have $f^*(x)=t\in[0,1]$, t directly corresponds to the ratio of densities not only in the original space $t=p(x)/(p(x)+q(x))$, but also in the 1D discriminator output space $t=[f^*{}_\sharp p](t)/([f^*{}_\sharp p](t)+[f^*{}_\sharp q](t))$. Figure 4-3 provides a visual illustration of the statement of the proposition.

4.4.2 Algorithmic connections

The result of Proposition 4.4.3 suggests methods similar to our ASA algorithm presented in Section 4.3 can achieve different notions of alignment by minimizing objectives discussed in Section 4.4.1 between the 1D pushforward distributions. We consider the setup used in Section 4.3 but without history buffers to simplify the analysis, as their usage is orthogonal to our discussion in this section.

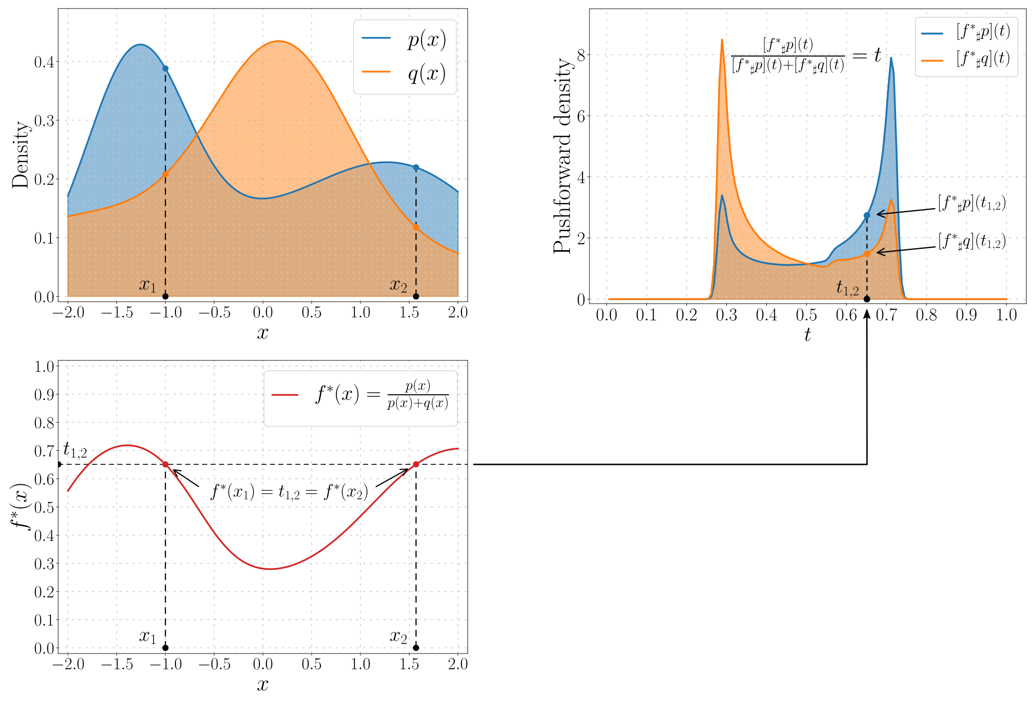


Figure 4-3: Visual illustration of the statement of Proposition 4.4.4. The top-left panel shows two example PDFs $p(x)$, $q(x)$ on closed interval $[-2, 2]$. The bottom-left panel shows the optimal discriminator function $f^*(x) = p(x)/(p(x)+q(x))$ as a function of x on $[-2, 2]$. The top-right panel shows the PDFs $[f^*{}_\sharp p](t)$, $[f^*{}_\sharp q](t)$ of the pushforward distributions $f^*{}_\sharp p$, $f^*{}_\sharp q$ induced by the discriminator mapping f^*. f^* maps $[-2, 2]$ to $[0, 1]$ and $[f^*{}_\sharp p]$, $[f^*{}_\sharp q]$ are defined on $[0, 1]$.
Consider point $x_1 \in [-2, 2]$. The value $f^*(x_1)$ characterizes the ratio of densities $p(x_1)/(p(x_1) + q(x_1))$ at x_1. For another point x_2 mapped to the same value $f^*(x_2) = f^*(x_1) = t_{1,2}$, the ratio of densities $p(x_2)/(p(x_2) + q(x_2))$ is the same as $p(x_1)/(p(x_1)+q(x_1))$. All points x mapped to $t_{1,2}$ share the same ratio of the densities $p(x)/(p(x) + q(x))$. This fact implies that the ratio of the pushforward densities $[f^*{}_\sharp p](t_{1,2})/([f^*{}_\sharp p](t_{1,2}) + [f^*{}_\sharp q](t_{1,2}))$ at $t_{1,2}$ must be the same as the ratio of densities $p(x_1)/(p(x_1) + q(x_1)) = t_{1,2}$ at x_1 (or x_2). The pushforward PDFs $[f^*{}_\sharp q](t)$, $[f^*{}_\sharp q](t)$ satisfy property $[f^*{}_\sharp p](t)/([f^*{}_\sharp p](t) + [f^*{}_\sharp q](t)) = t$ for all $t \in \text{supp}(f^*{}_\sharp p) \cup \text{supp}(f^*{}_\sharp q)$.

Recall that we work with a mini-batch setting, where $\{x_i^p\}_{i=1}^m$ and $\{x_i^q\}_{i=1}^m$ are sampled from p and q respectively, and g is the adversarial log-loss discriminator. We denote the corresponding 1D outputs from the log-loss discriminator by $o^p = \{o_i^p\}_{i=1}^m = \{g(x_i^p)\}_{i=1}^m$ and o^q (defined similarly).

Distribution alignment. We adapt (4.8) for $\{o_i^p\}_{i=1}^m$ and $\{o_i^q\}_{i=1}^m$:

$$\mathcal{D}_W(o^p, o^q) = \inf_{\gamma \in \Gamma(o^p, o^q)} \frac{1}{m} \sum_{i=1}^m \sum_{j=1}^m \gamma_{ij} d(o_i^p, o_j^q), \tag{4.12}$$

where $\Gamma(o^p, o^q)$ is the set of $m \times m$ doubly stochastic matrices. Since o^p and o^q are sets of 1D samples with the same size, it can be shown [199] that the optimal γ^* corresponds to an assignment π^*, which pairs points in the sorting order and can be computed efficiently by sorting both sets o^p and o^q. The transportation cost is zero if and only if there exists an invertible $\underline{\text{1-to-1}}$ assignment π^* such that $o_i^p = o_{\pi^*(i)}^q$. GAN training algorithms proposed in [52, 53] utilize the above sorting procedure to estimate the maximum sliced Wasserstein distance.

Relaxed distribution alignment. Similarly, we can adapt (4.9):

$$\mathcal{D}_W^\beta(o^p, o^q) = \inf_{\gamma \in \Gamma_\beta(o^p, o^q)} \frac{1}{m} \sum_{i=1}^m \sum_{j=1}^m \gamma_{ij} d(o_i^p, o_j^q), \tag{4.13}$$

where $\Gamma_\beta(o^p, o^q)$ is the set of $m \times m$ matrices with non-negative real entries, such that $\sum_{j=1}^m \gamma_{ij} = 1, \forall i$ and $\sum_{i=1}^m \gamma_{ij} \le 1 + \beta, \forall j$. The optimization goal in (4.13) is to find a "soft-assignment" γ which describes the transportation of probability mass from points o_i^p in o^p to points o_i^q in o^q. The parameter β controls the set of admissible assignments Γ_β, which is similar to its role discussed in Section 4.4.1: with transportation assignment tolerance β, the total mass of points in o^p transported to each of the points o_i^q cannot exceed $1 + \beta$. We refer to such assignments as $\underline{(\beta + 1)\text{-to-1}}$ assignment. The transportation cost is zero if and only if there exists such an assignment between o^p and o^q.

It can be shown (see Appendix B.2) that for integer value of β, the set of minimizers of (4.13) must contain a "hard-assignment" transportation plan, which assigns each point o_i^p to exactly one point o_j^q. Then $(1+\beta)$ gives the upper bound on the number of points o_i^p that can be transported to given point o_j^q. This hard assignment problem can be solved quasi-linearly with worst case time complexity $\mathcal{O}\left((\beta + 1)m^2\right)$ [28], which, combined with Proposition 4.4.3, can lead to new algorithms for relaxed distribution alignment besides those proposed in Wu et al. [262].

Support alignment. When $\beta = \infty$, the sum $\sum_{i=1}^m \gamma_{ij}$ is unconstrained for all j, and each point o_i^p can be assigned to any of the points o_j^q. The optimal solution is simply 1-nearest neighbor assignment, or to follow the above terminology, $\underline{\infty\text{-to-1}}$

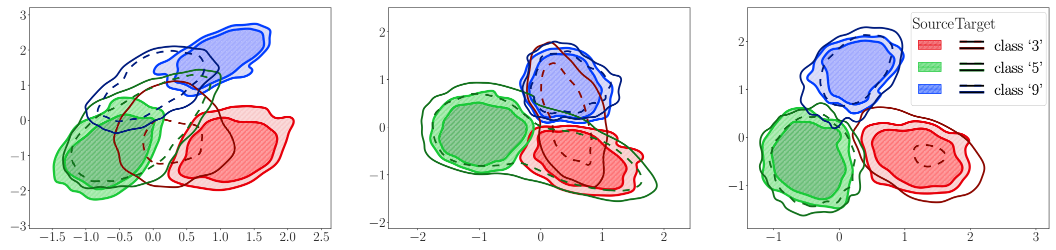
assignment.

4.5 Related Work

Distribution alignment. Apart from the works, e.g. [3, 9, 52, 53, 72–74, 78, 87, 132, 145, 146, 155, 160, 191, 200, 220, 224, 238, 246, 255, 260, 277], that focus on distribution alignment, there are methods [153, 154, 194, 234, 235] based on the alignment of some characteristics of the distribution, such as first or second moments. Our work is concerned with a different problem, support alignment, which is a novel objective in this line of work. In terms of methodology, our use of the discriminator output space to work with easier optimization in 1D is inspired by a line of work [52, 53, 74, 220, 260] on sliced Wasserstein distance based models. Our result in Proposition 4.4.3 also provides theoretical insight on the practical effectiveness of 1D OT in [53].

Relaxed distribution alignment. In Section 4.4, we have already covered in detail the connections between our work and [262]. Balaji et al. [10] introduced relaxed distribution alignment with a different focus, aiming to be insensitive to outliers. Chamfer distance/divergence (CD) is used to compute similarity between images/3D point clouds [69, 180]. For text data, Kusner et al. [133] presented Relaxed Word Mover's Distance (RWMD) to prune candidates of similar documents. CD and RWMD are essentially the same as the empirical version of SSD divergence (4.6) with $d(\cdot, \cdot)$ being the Euclidean distance. All of these distances between empirical measures are computed by finding the nearest neighbor assignments. Our subroutine of calculating the support distance in the 1D discriminator output space is done similarly by finding nearest neighbors within the current batch and history buffers.

Support estimation. There exists a series of work, e.g. [37, 51, 102, 126, 195, 215, 222, 241, 271], on novelty/anomaly detection problem, which can be casted as support estimation. We consider a fundamentally different problem setting. Our goal is to align the supports and our approach does not directly estimate the supports. Instead, we implicitly learn the relationships between supports (density ratio to be



(a) No DA (avg acc: 63%) $\mathcal{D}_W(p_Z^\theta, q_Z^\theta) = 0.78$ $\mathcal{D}_\triangle(p_Z^\theta, q_Z^\theta) = 0.10$

(b) DANN (avg acc: 75%) $\mathcal{D}_W(p_Z^\theta, q_Z^\theta) = 0.07$ $\mathcal{D}_\triangle(p_Z^\theta, q_Z^\theta) = 0.02$

(c) ASA-abs (avg acc: 94%) $\mathcal{D}_W(p_Z^\theta, q_Z^\theta) = 0.59$ $\mathcal{D}_\triangle(p_Z^\theta, q_Z^\theta) = 0.03$

Figure 4-4: Visualization of learned 2D embeddings on 3-class USPS→MNIST with label distribution shift. In source domain, all classes have equal probability $\frac{1}{3}$. The target probabilities of classes '3', '5', '9' are $[23\%, 65\%, 12\%]$. Each panel shows 2 level sets (outer one approximates the support) of the kernel density estimates of embeddings in source (filled regions) and target domains (solid/dashed lines). We report the average class accuracy of the target domain, $\mathcal{D}_W$ and $\mathcal{D}_\triangle$ between embeddings.

specific) via a discriminator.

4.6 Experiments

Problem setting. We evaluate our proposed ASA method in the setting of unsupervised domain adaptation (UDA). The goal of UDA algorithms is to train and "adapt" a classification model $M : \mathcal{X} \to \mathcal{Y}$ from source domain distribution $p_{X,Y}$ to target domain distribution $q_{X,Y}$ given the access to a labeled source dataset $\{x_i^p, y_i^p\}_{i=1}^{N^p} \sim p_{X,Y}$ and an unlabeled target dataset $\{x_i^q\}_{i=1}^{N^q} \sim q_X$.

A common approach for UDA is to represent M as $C^\phi \circ F^\theta$: a classifier $C^\phi : \mathcal{Z} \to \mathcal{Y}$ and a feature extractor $F^\theta : \mathcal{X} \to \mathcal{Z}$, and train C^ϕ and F^θ by minimizing: 1) classification loss ℓ_{cls} on source examples; 2) alignment loss $\mathcal{D}_{\text{align}}$ measuring discrepancy between $p_Z^\theta = F^\theta{}_\sharp p_X$ and $q_Z^\theta = F^\theta{}_\sharp q_X$:

$$\min_{\phi,\theta} \frac{1}{N^p} \sum_{i=1}^{N^p} \ell_{\text{cls}}(C^\phi(F^\theta(x_i^p)), y_i^p) + \lambda \cdot \mathcal{D}_{\text{align}}\left(\{F^\theta(x_i^p)\}_{i=1}^{N^p}, \{F^\theta(x_i^q)\}_{i=1}^{N^q}\right), \quad (4.14)$$

In practice $\mathcal{D}_{\text{align}}$ is an estimate of a divergence measure via an adversarial discriminator

g^ψ. Choices of $\mathcal{D}_{\text{align}}$ include f-divergences [73, 183] and Wasserstein distance [9] to enforce distribution alignment and versions of re-weighted/relaxed distribution divergences [238, 262] to enforce relaxed distribution alignment. For support alignment, we apply the proposed ASA method as the alignment subroutine in (4.14) with log-loss discriminator g^ψ (4.5) and $\mathcal{D}_{\text{align}}$ computed as (4.7).

Task specifications. We consider 3 UDA tasks: USPS→MNIST, STL→CIFAR, and VisDA-2017, and 2 versions of ASA: **ASA-sq**, **ASA-abs** corresponding to squared and absolute distances respectively for $d(\cdot,\cdot)$ in (4.7). We compare ASA with: **No DA** (no domain adaptation), **DANN** [73] (distribution alignment with JS divergence), **VADA** [225] (distribution alignment with virtual adversarial training), **IWDAN**, **IWCDAN** [238] (relaxed distribution alignment via importance weighting) **sDANN-β** [262] (relaxed/β-admissible JS divergence via re-weighting). Further experiment details are provided in Appendix B.3.

To evaluate the robustness of the methods, we simulate label distribution shift by subsampling source and target dataset, so that source has balanced label distribution and target label distribution follows the power law $q_Y(y) \propto \sigma(y)^{-\alpha}$, where σ is a random permutation of class labels $\{1, \dots, K\}$ and α controls the severity of the shift ($\alpha = 0$ means balanced label distribution). For each task, we generate 5 random permutations σ for 4 different shift levels $\alpha \in \{0, 1, 1.5, 2\}$. Essentially we transform each (source, target) dataset pair to $5 \times 4 = 20$ tasks of different difficulty levels, since classes are not equally difficult and different permutations can give them different weights.

Evaluation metrics. We choose the average (per-)class accuracy and minimum (per-)class accuracy on the target test set as evaluation metrics. Under the average class accuracy metric, all classes are treated as equally important (despite the unequal representation during training for $\alpha > 0$), and the minimum class accuracy focuses on model's worst within-class performance. In order to account for the variability of task difficulties across random permutations of target labels, we report robust statistics, median and a 25-75 percentile interval, across 5 runs.

Illustrative example. First we consider a simplified setting to intuitively under-

Table 4.1: Average and minimum class accuracy (%) on USPS→MNIST with different levels of shifts in label distributions (higher α implies more severe imbalance). We report median (the main number), and 25 (subscript) and 75 (superscript) percentiles across 5 runs.

	$\alpha = 0.0$		$\alpha = 1.0$		$\alpha = 1.5$		$\alpha = 2.0$	
Algorithm	average	min	average	min	average	min	average	min
No DA	$71.9^{72.9}_{70.4}$	$20.3^{22.9}_{17.6}$	$72.9^{74.7}_{72.0}$	$25.8^{31.8}_{18.3}$	$71.3^{72.5}_{71.2}$	$27.5^{37.3}_{24.2}$	$71.3^{73.0}_{70.6}$	$16.6^{26.8}_{10.8}$
DANN	$97.8^{97.8}_{97.6}$	$96.0^{96.1}_{95.8}$	$83.5^{84.6}_{76.7}$	$25.1^{36.9}_{08.4}$	$70.0^{71.2}_{63.9}$	$01.1^{01.5}_{01.0}$	$57.8^{60.4}_{52.0}$	$00.9^{01.6}_{00.5}$
VADA	$\mathbf{98.0}^{98.0}_{97.9}$	$96.2^{96.3}_{95.9}$	$88.2^{89.9}_{88.1}$	$48.9^{50.0}_{47.8}$	$78.2^{83.1}_{70.7}$	$06.6^{23.5}_{02.4}$	$61.9^{65.4}_{56.3}$	$01.4^{01.5}_{00.8}$
IWDAN	$97.5^{97.5}_{97.4}$	$95.7^{95.9}_{95.7}$	$95.7^{95.8}_{92.6}$	$81.3^{82.3}_{67.1}$	$86.5^{87.8}_{80.2}$	$15.2^{55.0}_{04.2}$	$74.4^{78.6}_{70.0}$	$07.3^{22.4}_{06.3}$
IWCDAN	$\mathbf{98.0}^{98.1}_{97.9}$	$\mathbf{96.6}^{96.9}_{96.4}$	$\mathbf{96.7}^{97.5}_{93.3}$	$85.1^{93.9}_{65.3}$	$91.3^{93.8}_{90.5}$	$66.5^{74.5}_{64.1}$	$77.5^{82.3}_{77.3}$	$22.2^{45.4}_{02.7}$
sDANN-4	$87.4^{95.7}_{87.2}$	$05.6^{90.0}_{05.6}$	$94.9^{94.9}_{94.7}$	$\mathbf{85.7}^{87.7}_{84.4}$	$86.8^{89.1}_{85.5}$	$21.6^{50.3}_{15.4}$	$81.5^{83.1}_{81.3}$	$39.3^{56.2}_{37.9}$
ASA-sq	$93.7^{93.9}_{93.3}$	$89.2^{89.4}_{88.4}$	$92.3^{93.6}_{91.5}$	$83.5^{88.7}_{80.8}$	$90.9^{92.1}_{89.6}$	$69.9^{82.0}_{66.6}$	$87.2^{89.3}_{85.8}$	$62.5^{69.3}_{46.4}$
ASA-abs	$94.1^{94.5}_{93.8}$	$88.9^{91.2}_{87.0}$	$92.8^{93.2}_{89.3}$	$78.9^{82.9}_{65.1}$	$\mathbf{92.5}^{92.9}_{90.9}$	$\mathbf{82.4}^{85.4}_{74.5}$	$\mathbf{90.4}^{90.7}_{89.2}$	$\mathbf{68.4}^{73.0}_{67.5}$

stand and directly analyze the behavior of our proposed support alignment method in domain adaptation under label distribution shift. We consider a 3-class USPS→MNIST problem by selecting a subset of examples corresponding to digits '3', '5', and '9', and use a feature extractor network with 2D output space. We introduce label distribution shift as described above with $\alpha = 1.5$, i.e. the probabilities of classes in the target domain are 12%, 23%, and 65%. We compare No DA, DANN, ASA-abs by their average target classification accuracy, Wasserstein distance $\mathcal{D}_W(p_Z^\theta, q_Z^\theta)$ and SSD divergence $\mathcal{D}_\triangle(p_Z^\theta, q_Z^\theta)$ between the learned embeddings of source and target domain. We apply a global affine transformation to each embedding space in order to have comparable distances between different spaces: we center the embeddings so that their average is 0 and re-scale them so that their average norm is 1. The results are shown in Figure 4-4 and Table 4.5. Compared to No DA, both DANN and ASA achieve support alignment. DANN enforces distribution alignment, and thus places some target embeddings into regions corresponding to the wrong class. In comparison, ASA does not enforce distribution alignment and maintains good class correspondence across the source and target embeddings.

Main results. The results of the main experimental evaluations are shown in Tables 4.1, 4.2, 4.3. Without any alignment, source only training struggles relatively

Table 4.2: Results on STL→CIFAR. Same setup and reporting metrics as Table 4.1.

	$\alpha = 0.0$		$\alpha = 1.0$		$\alpha = 1.5$		$\alpha = 2.0$	
Algorithm	average	min	average	min	average	min	average	min
No DA	$69.9^{70.0}_{69.8}$	$49.8^{50.6}_{45.3}$	$68.8^{69.3}_{68.3}$	$47.2^{48.2}_{45.3}$	$66.8^{67.2}_{66.4}$	$46.0^{47.0}_{45.8}$	$65.8^{66.7}_{64.8}$	$43.7^{44.6}_{41.6}$
DANN	$75.3^{75.4}_{74.9}$	$54.6^{56.6}_{54.2}$	$69.9^{70.1}_{68.6}$	$44.8^{45.1}_{40.7}$	$64.9^{67.1}_{63.7}$	$34.9^{36.8}_{33.9}$	$63.3^{64.8}_{57.4}$	$27.0^{28.5}_{21.2}$
VADA	$\mathbf{76.7}^{76.7}_{76.6}$	$\mathbf{56.9}^{58.3}_{53.5}$	$70.6^{71.0}_{70.0}$	$47.7^{48.8}_{44.0}$	$66.1^{66.5}_{65.4}$	$35.7^{39.3}_{33.3}$	$63.2^{64.7}_{60.2}$	$25.5^{28.0}_{25.2}$
IWDAN	$69.9^{70.7}_{69.9}$	$50.5^{50.6}_{47.9}$	$68.7^{69.1}_{68.6}$	$45.8^{50.5}_{44.8}$	$67.1^{67.3}_{65.9}$	$44.7^{44.8}_{40.4}$	$64.4^{64.9}_{63.6}$	$36.8^{37.9}_{34.5}$
IWCDAN	$70.1^{70.2}_{70.1}$	$47.8^{49.3}_{42.4}$	$69.4^{69.4}_{69.1}$	$47.1^{51.3}_{46.3}$	$66.1^{67.2}_{65.0}$	$39.9^{40.8}_{37.7}$	$64.5^{65.1}_{63.9}$	$37.0^{40.2}_{35.5}$
sDANN-4	$71.8^{72.1}_{71.7}$	$52.1^{52.8}_{52.1}$	$\mathbf{71.1}^{71.7}_{70.4}$	$49.9^{51.8}_{48.1}$	$69.4^{70.0}_{68.7}$	$\mathbf{48.6}^{49.0}_{43.5}$	$66.4^{67.9}_{66.2}$	$39.0^{47.1}_{33.6}$
ASA-sq	$71.7^{71.9}_{71.7}$	$52.9^{53.4}_{46.7}$	$70.7^{71.0}_{70.4}$	$\mathbf{51.6}^{52.7}_{46.8}$	$69.2^{69.3}_{69.2}$	$45.6^{52.0}_{43.3}$	$\mathbf{68.1}^{68.2}_{67.2}$	$\mathbf{44.7}^{45.9}_{39.8}$
ASA-abs	$71.6^{71.7}_{71.2}$	$49.0^{53.5}_{48.4}$	$70.9^{71.0}_{70.8}$	$49.2^{50.0}_{47.3}$	$\mathbf{69.6}^{69.9}_{69.6}$	$43.2^{49.5}_{42.1}$	$67.8^{68.2}_{66.6}$	$40.9^{49.0}_{35.4}$

Table 4.3: Results on VisDA17. Same setup and reporting metrics as Table 4.1.

	$\alpha = 0.0$		$\alpha = 1.0$		$\alpha = 1.5$		$\alpha = 2.0$	
Algorithm	average	min	average	min	average	min	average	min
No DA	$49.5^{50.5}_{49.4}$	$22.2^{24.6}_{22.2}$	$50.2^{50.8}_{49.2}$	$21.2^{21.3}_{20.7}$	$47.1^{47.6}_{46.6}$	$18.6^{22.2}_{18.6}$	$45.3^{46.5}_{45.2}$	$19.5^{19.8}_{14.4}$
DANN	$\mathbf{75.4}^{76.2}_{74.4}$	$36.7^{40.9}_{35.6}$	$64.1^{65.3}_{62.8}$	$25.0^{29.3}_{24.8}$	$52.1^{52.3}_{51.4}$	$11.5^{12.4}_{11.4}$	$43.1^{44.3}_{39.1}$	$03.6^{14.3}_{03.6}$
VADA	$75.3^{76.0}_{74.8}$	$40.5^{41.8}_{39.7}$	$64.6^{65.1}_{61.2}$	$22.8^{28.2}_{21.7}$	$53.0^{54.2}_{51.6}$	$14.8^{21.7}_{13.7}$	$43.9^{44.7}_{40.9}$	$08.5^{11.1}_{05.0}$
IWDAN	$73.2^{73.3}_{72.9}$	$31.7^{34.8}_{22.8}$	$64.4^{64.6}_{61.1}$	$12.1^{24.7}_{05.0}$	$51.3^{56.6}_{51.0}$	$04.6^{10.4}_{02.1}$	$45.1^{48.0}_{41.7}$	$04.6^{13.6}_{01.2}$
IWCDAN	$71.6^{75.2}_{70.6}$	$27.6^{28.0}_{22.8}$	$60.6^{61.0}_{60.2}$	$01.1^{11.3}_{00.7}$	$49.7^{51.9}_{45.6}$	$02.2^{05.7}_{00.2}$	$38.3^{46.2}_{37.3}$	$00.6^{01.7}_{00.3}$
sDANN-4	$72.4^{73.3}_{71.8}$	$37.8^{40.8}_{32.3}$	$\mathbf{68.4}^{68.7}_{66.2}$	$26.6^{29.4}_{26.2}$	$57.2^{57.8}_{56.8}$	$18.6^{23.9}_{16.7}$	$50.7^{51.7}_{49.8}$	$18.6^{20.0}_{17.1}$
ASA-sq	$64.9^{65.0}_{63.7}$	$35.7^{35.8}_{32.1}$	$61.8^{63.2}_{60.6}$	$\mathbf{31.4}^{34.4}_{20.4}$	$\mathbf{57.8}^{58.3}_{55.5}$	$\mathbf{26.7}^{32.1}_{17.3}$	$51.9^{52.0}_{50.8}$	$18.3^{21.2}_{16.9}$
ASA-abs	$64.8^{65.0}_{64.5}$	$\mathbf{40.6}^{41.9}_{36.0}$	$62.0^{62.3}_{60.5}$	$27.3^{29.7}_{16.7}$	$57.1^{58.4}_{56.2}$	$26.0^{31.2}_{13.9}$	$\mathbf{52.5}^{56.6}_{51.9}$	$\mathbf{19.7}^{22.2}_{17.7}$

to adapt to the target domain. Nonetheless, its performance across the imbalance levels remains robust, since the training procedure is the same. Agreeing with the observation and theoretical results from previous work [143, 238, 239, 262, 278], distribution alignment methods (DANN and VADA) perform well when there is no shift but suffer otherwise, whereas relaxed distribution alignment methods (IWDAN, IWCDAN and sDANN-β) show more resilience to shifts. On all tasks with positive α, we observe that it is common for the existing methods to achieve good class average accuracies while suffering significantly on some individual classes. These results suggest that the often-ignored but important min-accuracy metric can be very challenging. Finally, our support alignment methods (ASA-sq and ASA-abs) are the most robust ones against the shifts, while still being competitive in the more balanced settings

($\alpha = 0$ or 1). We achieve best results in the more imbalanced and difficult tasks ($\alpha = 1.5$ or 2) for almost all categories on all datasets. Please refer to Appendix B.3 for ablation studies and additional comparisons.

Effect of history size. To quantify the effects of mini-batch training mentioned in Section 4.3, we explore different sizes of history buffers on USPS→MNIST task with the label distribution shift $\alpha = 1.5$. The results are presented in Figure 4-5 and Table 4.4. Figure 4-6 shows the distributions of outputs of the learned discriminator at the end of the training. While without any alignment objectives neither the densities nor the supports of $g^{\psi}{}_{\sharp}p_Z^{\theta}$ and $g^{\psi}{}_{\sharp}q_Z^{\theta}$ are aligned, both alignment methods approximately satisfy their respective alignment constraints. Compared with DANN results, ASA with small history size performs similarly to distribution alignment, while all history sizes are enough for support alignment. We also observe the correlation between distribution distance and target accuracy: under label distribution shifts, the better distribution alignment is achieved, the more target accuracy suffers. Note that with too big history buffers (e.g. $n = 5000$), we observe a sudden drop in performance and increases in distances. We hypothesize that this could be caused by the fact that the history buffer stores discriminator output values from the past steps while the discriminator parameters constantly evolve during training. As a result, for a large history buffer, the older items might no longer accurately represent the current pushforward distribution as they become outdated.

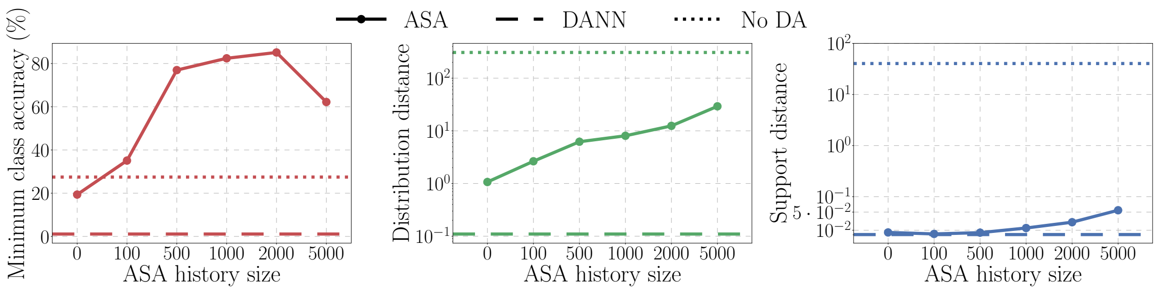


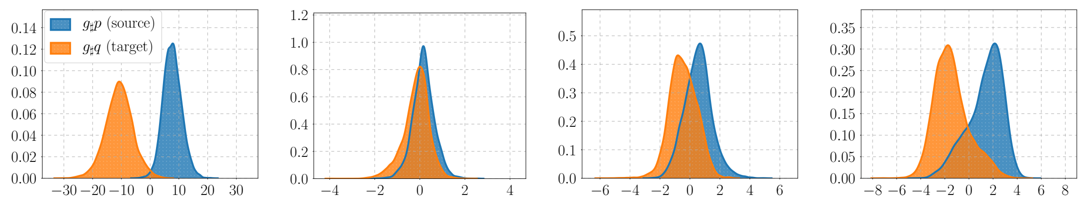
Figure 4-5: Evaluation of history size effect for ASA on MNIST→USPS with the label distribution shift ($\alpha = 1.5$). The panels show (left to right): minimum class accuracy on target test set; Wasserstein distance $\mathcal{D}_W(g^{\psi}{}_{\sharp}p_Z^{\theta}, g^{\psi}{}_{\sharp}q_Z^{\theta})$ between the pushforward distributions of source and target representations induced by the discriminator; SSD divergence $\mathcal{D}_{\triangle}(g^{\psi}{}_{\sharp}p_Z^{\theta}, g^{\psi}{}_{\sharp}q_Z^{\theta})$ between the pushforward distributions. In each panel the dashed lines show the respective quantities for "No DA" and DANN methods.

Direct evaluation of support distance. In order to directly evaluate the ability of ASA (with history buffers) to enforce support alignment, we consider the setting of the illustrative experiment shown in Figure 4-4 (3-class USPS$\to$MNIST adaptation with 2D feature extractor, $\alpha = 1.5$). We compare methods No DA, DANN, and ASA-abs (with different history buffer sizes). For each method we consider the embedding space of the learned feature extractor at the end of training and compute Wasserstein distance $\mathcal{D}_W(p_Z^\theta, q_Z^\theta)$ and SSD divergence $\mathcal{D}_\triangle(p_Z^\theta, q_Z^\theta)$ between the embeddings of source and target domain (note that we compute the distances in the original embedding space directly without projecting data to 1D with the discriminator). To ensure meaningful comparison of the distances between different embedding spaces, we apply a global affine transformation for each embedding space: we center the embeddings so that their average is 0 and re-scale them so that their average norm is 1. The results of this evaluation are shown in Table 4.5. We observe that, compared to no alignment and distribution alignment (DANN) methods, ASA aligns the supports without necessarily aligning the distributions (in this imbalanced setting, distribution alignment implies low adaptation accuracy).

Table 4.4: Analysis of effect history size parameter for ASA on USPS$\to$MNIST with class label distribution shift corresponding to $\alpha = 1.5$. We report distribution and support distances between the pushforward distributions $g^\psi{}_\sharp p_Z^\theta$ and $g^\psi{}_\sharp q_Z^\theta$, as well as the value of discriminator's log-loss.

		Target accuracy (%)		Distribution distances		
Method	History size	average	min	$\mathcal{D}_W(g^\psi{}_\sharp p_Z^\theta, g^\psi{}_\sharp q_Z^\theta)$	$\mathcal{D}_\triangle(g^\psi{}_\sharp p_Z^\theta, g^\psi{}_\sharp q_Z^\theta)$	Log-loss
No DA	—	71.28 $^{72.51}_{71.25}$	27.46 $^{37.26}_{24.21}$	307.56 $^{322.00}_{277.33}$	40.35 $^{46.06}_{32.10}$	00.05 $^{00.07}_{00.04}$
DANN	—	69.96 $^{71.25}_{63.89}$	01.11 $^{01.53}_{00.99}$	00.11 $^{00.11}_{00.10}$	00.00 $^{00.00}_{00.00}$	00.65 $^{00.65}_{00.65}$
ASA-abs	0	62.75 $^{64.35}_{61.78}$	19.36 $^{23.63}_{17.90}$	01.07 $^{01.15}_{00.99}$	00.01 $^{00.01}_{00.00}$	00.57 $^{00.58}_{00.56}$
ASA-abs	100	80.58 $^{81.73}_{78.22}$	35.09 $^{44.37}_{32.10}$	02.64 $^{02.70}_{02.15}$	00.00 $^{00.00}_{00.00}$	00.53 $^{00.53}_{00.52}$
ASA-abs	500	92.02 $^{92.76}_{90.56}$	76.96 $^{83.72}_{70.94}$	06.21 $^{06.48}_{05.69}$	00.00 $^{00.01}_{00.00}$	00.45 $^{00.45}_{00.45}$
ASA-abs	1000	92.54 $^{92.93}_{90.90}$	82.41 $^{85.43}_{74.53}$	08.06 $^{08.19}_{07.97}$	00.01 $^{00.02}_{00.01}$	00.41 $^{00.41}_{00.40}$
ASA-abs	5000	86.03 $^{87.50}_{84.86}$	62.19 $^{71.62}_{46.98}$	29.23 $^{29.63}_{24.54}$	00.05 $^{00.08}_{00.05}$	00.29 $^{00.30}_{00.29}$

Effect of conditional entropy loss. In order to quantify the improvements of the support alignment objective and the conditional entropy objective in separation, we conduct an ablation study. We evaluate all domain adaptation methods (except VADA



(a) No DA (b) DANN (c) ASA $(n = 0)$ (d) ASA $(n = 1000)$

Figure 4-6: Kernel density estimates (in the discriminator output space) of ${g^\psi}_\sharp p_Z^\theta$, ${g^\psi}_\sharp q_Z^\theta$ at the end of the training on USPS→MNIST task with $\alpha = 1.5$. n is the size of ASA history buffers.

Table 4.5: Results of No DA, DANN, and ASA-abs (with different history sizes) on 3-class USPS→MNIST adaptation with 2D feature extractor and label distribution shift corresponding to $\alpha = 1.5$. We report average and minimum target class accuracy, as well as Wasserstein distance $\mathcal{D}_W$ and support divergence $\mathcal{D}_\triangle$ between source p_Z^θ and target q_Z^θ 2D embedding distributions. We report median (the main number), and 25 (subscript) and 75 (superscript) percentiles across 5 runs.

Algorithm	History size	Accuracy (avg)	Accuracy (min)	$\mathcal{D}_W(p_Z^\theta, q_Z^\theta)$	$\mathcal{D}_\triangle(p_Z^\theta, q_Z^\theta)$
No DA	—	$63.0^{69.6}_{62.3}$	$45.3^{53.6}_{37.9}$	$0.78^{00.84}_{00.75}$	$0.10^{0.10}_{0.10}$
DANN	—	$75.6^{83.7}_{72.4}$	$54.8^{55.1}_{49.6}$	$0.07^{0.08}_{0.06}$	$0.02^{0.02}_{0.02}$
ASA-abs	0	$73.9^{84.1}_{73.4}$	$61.8^{72.4}_{54.6}$	$0.23^{0.47}_{0.22}$	$0.03^{0.03}_{0.03}$
ASA-abs	100	$88.5^{95.1}_{86.8}$	$71.4^{93.3}_{70.6}$	$0.54^{0.36}_{0.56}$	$0.03^{0.03}_{0.03}$
ASA-abs	500	$94.5^{94.7}_{88.7}$	$89.0^{90.3}_{83.1}$	$0.59^{0.64}_{0.55}$	$0.03^{0.03}_{0.03}$
ASA-abs	1000	$91.1^{93.0}_{91.1}$	$85.6^{86.2}_{80.7}$	$0.59^{0.62}_{0.55}$	$0.03^{0.03}_{0.03}$
ASA-abs	2000	$94.0^{94.7}_{91.2}$	$88.6^{89.4}_{80.2}$	$0.62^{0.66}_{0.58}$	$0.03^{0.03}_{0.03}$
ASA-abs	5000	$82.1^{83.9}_{81.8}$	$68.9^{70.9}_{65.5}$	$0.64^{0.67}_{0.63}$	$0.04^{0.04}_{0.04}$

which uses the conditional entropy in the original implementation) on STL→CIFAR task without the conditional entropy loss ($\lambda_{\text{ent}} = 0$). The results of the ablation study are presented in Table 4.6. We observe that the effect of the auxiliary conditional entropy is essentially the same for all methods across all imbalance levels: with $\lambda_{\text{ent}} = 0.1$ the accuracy either improves (especially the average class accuracy) or roughly stays on the same level. The relative ranking of distribution alignment, relaxed distribution alignment, and support alignment methods is the same with both $\lambda_{\text{ent}} = 0$ and $\lambda_{\text{ent}} = 0.1$. The results demonstrate that the benefits of support alignment approach and conditional entropy are orthogonal.

Effect of alignment weight. We provide additional experimental results compar-

Table 4.6: Results of ablation experiments of the effect of auxiliary conditional entropy loss on STL→CIFAR data. Same setup and reporting metrics as Table 4.1.

		$\alpha = 0.0$		$\alpha = 1.0$		$\alpha = 1.5$		$\alpha = 2.0$	
Algorithm	λ_{ent}	average	min	average	min	average	min	average	min
DANN	0.0	74.6 $^{75.1}_{74.1}$	51.5 $^{55.0}_{49.9}$	68.4 $^{69.2}_{67.0}$	43.2 $^{43.7}_{41.2}$	65.7 $^{65.9}_{62.8}$	35.5 $^{36.2}_{29.6}$	62.5 $^{64.6}_{60.0}$	27.5 $^{27.5}_{25.7}$
DANN	0.1	75.3 $^{75.4}_{74.9}$	54.6 $^{56.6}_{54.2}$	69.9 $^{70.1}_{68.6}$	44.8 $^{45.1}_{40.7}$	64.9 $^{67.1}_{63.7}$	34.9 $^{36.8}_{33.9}$	63.3 $^{64.8}_{57.4}$	27.0 $^{28.5}_{21.2}$
IWDAN	0.0	70.4 $^{70.7}_{70.2}$	47.2 $^{48.0}_{46.8}$	68.6 $^{68.8}_{68.4}$	43.6 $^{46.3}_{43.2}$	66.7 $^{67.9}_{66.0}$	44.7 $^{46.2}_{43.3}$	63.9 $^{66.1}_{62.9}$	36.5 $^{37.3}_{32.7}$
IWDAN	0.1	69.9 $^{70.7}_{69.9}$	50.5 $^{50.6}_{47.9}$	68.7 $^{69.1}_{68.6}$	45.8 $^{50.5}_{44.8}$	67.1 $^{67.3}_{65.9}$	44.7 $^{44.8}_{40.4}$	64.4 $^{64.9}_{63.6}$	36.8 $^{37.9}_{34.5}$
IWCDAN	0.0	70.1 $^{70.8}_{70.0}$	50.5 $^{50.8}_{49.1}$	68.6 $^{69.4}_{68.2}$	44.2 $^{45.8}_{41.2}$	66.0 $^{66.0}_{65.9}$	45.0 $^{47.8}_{43.7}$	63.8 $^{64.1}_{62.3}$	37.3 $^{37.7}_{33.6}$
IWCDAN	0.1	70.1 $^{70.2}_{70.1}$	47.8 $^{49.3}_{42.4}$	69.4 $^{69.4}_{69.1}$	47.1 $^{51.3}_{46.3}$	66.1 $^{67.2}_{65.0}$	39.9 $^{40.8}_{37.7}$	64.5 $^{65.1}_{63.9}$	37.0 $^{40.2}_{35.5}$
sDANN-4	0.0	69.4 $^{70.0}_{68.8}$	46.5 $^{49.7}_{45.1}$	69.6 $^{69.7}_{69.3}$	49.1 $^{49.2}_{47.4}$	68.0 $^{68.6}_{67.8}$	48.2 $^{48.8}_{42.6}$	66.3 $^{66.4}_{64.2}$	40.7 $^{42.9}_{36.6}$
sDANN-4	0.1	71.8 $^{72.1}_{71.7}$	52.1 $^{52.8}_{52.1}$	71.1 $^{71.7}_{70.4}$	49.9 $^{51.8}_{48.1}$	69.4 $^{70.0}_{68.7}$	48.6 $^{49.0}_{43.5}$	66.4 $^{67.9}_{66.2}$	39.0 $^{47.1}_{33.6}$
ASA-sq	0.0	69.9 $^{70.3}_{69.9}$	48.0 $^{50.1}_{46.6}$	68.8 $^{68.9}_{68.6}$	47.3 $^{49.3}_{45.3}$	68.1 $^{68.7}_{67.2}$	45.4 $^{47.8}_{45.2}$	65.7 $^{66.4}_{65.6}$	43.6 $^{45.0}_{41.3}$
ASA-sq	0.1	71.7 $^{71.9}_{71.7}$	52.9 $^{53.4}_{46.7}$	70.7 $^{71.0}_{70.4}$	51.6 $^{52.7}_{46.8}$	69.2 $^{69.3}_{69.2}$	45.6 $^{52.0}_{43.3}$	68.1 $^{68.2}_{67.2}$	44.7 $^{45.9}_{39.8}$
ASA-abs	0.0	69.8 $^{70.0}_{68.9}$	45.7 $^{48.0}_{45.4}$	68.4 $^{68.6}_{68.4}$	44.3 $^{46.8}_{44.0}$	67.9 $^{68.1}_{67.0}$	46.6 $^{48.4}_{40.4}$	66.3 $^{66.9}_{65.7}$	41.6 $^{44.9}_{40.3}$
ASA-abs	0.1	71.6 $^{71.7}_{71.2}$	49.0 $^{53.5}_{48.4}$	70.9 $^{71.0}_{70.8}$	49.2 $^{50.0}_{47.3}$	69.6 $^{69.9}_{69.6}$	43.2 $^{49.5}_{42.1}$	67.8 $^{68.2}_{66.6}$	40.9 $^{49.0}_{35.4}$

ing ASA with DANN and VADA across different values of the alignment loss weight λ_{align} on STL→CIFAR task. The results are shown in Table 4.7. DANN with a higher alignment weight $\lambda_{\text{align}} = 1.0$ performs better in the balanced ($\alpha = 0$) setting and worse in the imbalanced ($\alpha > 0$) setting compared to a lower weight $\lambda_{\text{align}} = 0.1$, as the distribution alignment constraint is enforced stricter. VADA optimizes a combination of distribution alignment + VAT (virtual adversarial training) objectives [225], and we observe the same trend: with lower alignment weight $\lambda_{\text{align}} = 0.01$, VADA performs worse in the balanced setting and better in the imbalanced setting compared to a higher weight $\lambda_{\text{align}} = 0.1$. Weight $\lambda_{\text{align}} = 0.1$ is a middle ground between having poor performance in the imbalanced setting ($\lambda_{\text{align}} = 1.0$) and not sufficiently enforcing distribution alignment ($\lambda_{\text{align}} = 0.01$). The role of VAT (similarly to that of conditional entropy loss) is orthogonal to alignment objectives. Thus, we provide additional evaluations of combining support alignment and VAT (the "ASA-sq + VAT" entry in Table 4.7) with alignment weight $\lambda_{\text{align}} = 1.0$. The good performance of such combination shows that:

- One could improve our ASA's performance by using auxiliary objectives.

- Support alignment-based method performs qualitatively differently from the distribution alignment-based method since the performance holds with stricter support alignment, while distribution alignment needs to loosen the constraints considerably to reduce the performance degradation in the imbalanced setting.

Table 4.7: Results of comparison of ASA with DANN and VADA across different values of the alignment loss weight λ_{align} on STL→CIFAR data. Same setup and reporting metrics as Table 4.1.

		$\alpha = 0.0$		$\alpha = 1.0$		$\alpha = 1.5$		$\alpha = 2.0$	
Algorithm	λ_{align}	average	min	average	min	average	min	average	min
DANN	0.01	$72.3\,^{72.7}_{72.2}$	$49.5\,^{50.8}_{48.8}$	$70.6\,^{71.2}_{69.7}$	$48.9\,^{51.2}_{41.5}$	$68.5\,^{68.7}_{67.2}$	$46.1\,^{50.0}_{36.2}$	$65.9\,^{66.0}_{64.1}$	$36.7\,^{39.4}_{29.9}$
DANN	0.1	$75.3\,^{75.4}_{74.9}$	$54.6\,^{56.6}_{54.2}$	$69.9\,^{70.1}_{68.6}$	$44.8\,^{45.1}_{40.7}$	$64.9\,^{67.1}_{63.7}$	$34.9\,^{36.8}_{33.9}$	$63.3\,^{64.8}_{57.4}$	$27.0\,^{28.5}_{21.2}$
DANN	1.0	$77.2\,^{77.3}_{76.8}$	$58.5\,^{59.4}_{56.7}$	$66.3\,^{66.8}_{64.5}$	$37.9\,^{41.6}_{37.5}$	$62.8\,^{63.3}_{56.1}$	$27.5\,^{28.9}_{24.6}$	$58.7\,^{59.7}_{52.3}$	$18.5\,^{20.5}_{17.2}$
VADA	0.01	$74.4\,^{74.4}_{74.2}$	$54.2\,^{55.4}_{52.6}$	$71.7\,^{71.7}_{71.7}$	$51.6\,^{52.0}_{45.0}$	$69.5\,^{69.7}_{68.4}$	$47.5\,^{49.8}_{40.0}$	$65.9\,^{66.1}_{64.8}$	$37.2\,^{39.4}_{35.3}$
VADA	0.1	$76.7\,^{76.7}_{76.6}$	$56.9\,^{58.3}_{53.5}$	$70.6\,^{71.0}_{70.0}$	$47.7\,^{48.8}_{44.0}$	$66.1\,^{66.5}_{65.4}$	$35.7\,^{39.3}_{33.3}$	$63.2\,^{64.7}_{60.2}$	$25.5\,^{28.0}_{25.2}$
ASA-sq	0.1	$71.7\,^{71.9}_{71.7}$	$52.9\,^{53.4}_{46.7}$	$70.7\,^{71.0}_{70.4}$	$51.6\,^{52.7}_{46.8}$	$69.2\,^{69.3}_{69.2}$	$45.6\,^{52.0}_{43.3}$	$68.1\,^{68.2}_{67.2}$	$44.7\,^{45.9}_{39.8}$
ASA-sq + VAT	1.0	$74.2\,^{74.5}_{74.0}$	$52.2\,^{52.5}_{51.9}$	$72.2\,^{72.2}_{71.9}$	$53.5\,^{53.6}_{45.4}$	$70.6\,^{70.8}_{70.4}$	$48.9\,^{52.3}_{45.6}$	$67.4\,^{67.7}_{66.8}$	$43.0\,^{46.0}_{39.4}$

Comparison with optimal transport baselines. We provide additional experimental results comparing our method with OT-based methods for domain adaptation. We implement two OT-based methods which we describe below.

- The first method is a variant of the max-sliced Wasserstein distance (which was proposed for GAN training by Deshpande et al. [53]) for domain adaptation. In the table below we refer to this method as DANN-OT. In our implementation DANN-OT minimizes the Wasserstein distance between the pushforward distributions $g^*_\sharp p$, $g^*_\sharp q$ induced by the optimal log-loss discriminator g^* (4.5). As discussed in Section 4.4.2 (paragraph "Distribution alignment") the computation of the Wasserstein distance between 1D distributions can be implemented efficiently via sorting.

- The second method is an OT-based variant of DANN which uses a dual Wasserstein discriminator instead of the log-loss discriminator. In the table below we refer to this method as DANN-WGP. This method minimizes the Wasserstein distance in its dual Kantorovich form. We train the discriminator with the Wasserstein dual objective and a gradient penalty proposed to enforce Lipshitz-norm constraint [87].

We present the evaluation results of the OT-based methods on STL→CIFAR domain adaptation task in the Table 4.8. Note that the OT-based methods aim to enforce distribution alignment constraints. We observe that the OT-based methods follow the same trend as DANN: they deliver improved accuracy compared to No DA in the balanced setting, but suffer in the imbalanced settings ($\alpha > 0$) due to their distribution alignment nature.

Table 4.8: Results of comparison with optimal transport based methods on STL→CIFAR data. Same setup and reporting metrics as Table 4.1.

	$\alpha = 0.0$		$\alpha = 1.0$		$\alpha = 1.5$		$\alpha = 2.0$	
Algorithm	average	min	average	min	average	min	average	min
No DA	$69.9\,^{70.0}_{69.8}$	$49.8\,^{50.6}_{45.3}$	$68.8\,^{69.3}_{68.3}$	$47.2\,^{48.2}_{45.3}$	$66.8\,^{67.2}_{66.4}$	$46.0\,^{47.0}_{45.8}$	$65.8\,^{66.7}_{64.8}$	$43.7\,^{44.6}_{41.6}$
DANN	$75.3\,^{75.4}_{74.9}$	$54.6\,^{56.6}_{54.2}$	$69.9\,^{70.1}_{68.6}$	$44.8\,^{45.1}_{40.7}$	$64.9\,^{67.1}_{63.7}$	$34.9\,^{36.8}_{33.9}$	$63.3\,^{64.8}_{57.4}$	$27.0\,^{28.5}_{21.2}$
DANN-OT	$76.0\,^{76.0}_{75.8}$	$55.2\,^{55.5}_{54.3}$	$67.7\,^{68.9}_{67.1}$	$43.0\,^{43.7}_{36.5}$	$64.5\,^{65.1}_{60.9}$	$34.4\,^{34.6}_{29.3}$	$61.3\,^{62.0}_{54.4}$	$24.3\,^{25.5}_{23.2}$
DANN-WGP	$74.8\,^{75.1}_{74.7}$	$53.5\,^{54.4}_{53.3}$	$67.7\,^{67.9}_{65.3}$	$38.6\,^{41.0}_{34.4}$	$63.3\,^{63.4}_{57.1}$	$27.0\,^{32.4}_{26.3}$	$59.0\,^{61.8}_{54.3}$	$21.9\,^{22.5}_{18.6}$
sDANN-4	$71.8\,^{72.1}_{71.7}$	$52.1\,^{52.8}_{52.1}$	$71.1\,^{71.7}_{70.4}$	$49.9\,^{51.8}_{48.1}$	$69.4\,^{70.0}_{68.7}$	$48.6\,^{49.0}_{43.5}$	$66.4\,^{67.9}_{66.2}$	$39.0\,^{47.1}_{33.6}$
ASA-sq	$71.7\,^{71.9}_{71.7}$	$52.9\,^{53.4}_{46.7}$	$70.7\,^{71.0}_{70.4}$	$51.6\,^{52.7}_{46.8}$	$69.2\,^{69.3}_{69.2}$	$45.6\,^{52.0}_{43.3}$	$68.1\,^{68.2}_{67.2}$	$44.7\,^{45.9}_{39.8}$
ASA-abs	$71.6\,^{71.7}_{71.2}$	$49.0\,^{53.5}_{48.4}$	$70.9\,^{71.0}_{70.8}$	$49.2\,^{50.0}_{47.3}$	$69.6\,^{69.9}_{69.6}$	$43.2\,^{49.5}_{42.1}$	$67.8\,^{68.2}_{66.6}$	$40.9\,^{49.0}_{35.4}$

Wu et al. [262] propose method WDANN-β which minimizes the dual form of the asymmetrically-relaxed Wasserstein distance. However, they observe that sDANN-β outperforms WDANN-β in experiments. Hence, we use sDANN-β as a relaxed distribution alignment baseline in our experiments.

Chapter 5

Compositional Sculpting of Iterative Generative Processes

5.1 Introduction

Large-scale general-purpose pre-training of machine learning models has produced impressive results in computer vision [40, 124, 203], image generation [99, 205, 212], natural language processing [33, 43, 54, 184, 211], robotics [2, 32, 59] and basic sciences [115]. By distilling vast amounts of data, such models can produce powerful inferences that lead to emergent capabilities beyond the specified training objective [27]. However, generic pre-trained models are often insufficient for specialized tasks in engineering and basic sciences. Field-adaptation via techniques such as explicit fine-tuning on bespoke datasets [274], human feedback [186], or cleverly designed prompts [236, 258] is therefore often required. An alternative approach is to compose the desired distribution using multiple simpler component models.

Compositional generation [11, 49, 62–64, 96, 103, 142, 149, 150, 162, 251, 275] views a complex target distribution in terms of simpler pre-trained building blocks which it can learn to mix and match into a tailored solution to a specialized task. Besides providing a way to combine and reuse previously trained models, composition is a powerful modeling approach. A composite model fuses knowledge from multiple sources: base models trained for different tasks, enabling increased capacity beyond that of any of the base models in isolation. If each individual base model captures a certain property of the data, composing such models provides a way to specify distributions over examples that exhibit multiple properties simultaneously [97]. The need to construct complex distributions adhering to multiple constraints arises in numerous practical multi-objective design problems such as multi-objective molecule generation [108, 111, 264]. In the context of multi-objective generation, compositional modeling provides mechanisms for adjustment and control of the resulting distribution, which enables exploration of different trade-offs between the objectives and constraints.

Prior work on generative model composition [62, 96, 97] has developed operations for piecing together Energy-Based Models (EBMs) via algebraic manipulations of their energy functions. For example, consider two distributions $p_1(x) \propto \exp\{-E_1(x)\}$ and $p_2(x) \propto \exp\{-E_2(x)\}$ induced by energy functions E_1 and E_2. Their *product* $p_{\text{prod}}(x) \propto p_1(x)p_2(x) \propto \exp\left(-\big(E_1(x) + E_2(x)\big)\right)$ and *negation* $p_{\text{neg}}(x) \propto p_1(x)/(p_2(x))^\gamma \propto \exp\left(-\big(E_1(x) - \gamma E_2(x)\big)\right)$ correspond to operations on the underlying energy functions.

Iterative generative processes including diffusion models [99, 226, 229, 231] and GFlowNets [16, 19] progressively refine coarse objects into cleaner ones over multiple steps. The realization of effective compositions of these models is complicated by the fact that simple alterations in their generation processes result in non-trivial changes in the distributions of the final objects. For instance, the aforementioned product and negation between EBMs cannot be realized simply by means of adding or subtracting associated score-functions. Prior work addresses these challenges by connecting diffusion models with EBMs through annealed Markov-Chain Monte-Carlo (MCMC) inference. However, Metropolis-Hastings corrections are required to ensure that the annealing process reproduces the desired distribution [64].

Jain et al. [108] develop Multi-Objective GFlowNets (MOGFNs), an extension of GFlowNets for multi-objective optimization tasks. The goal of a vanilla GFlowNet model is to capture the distribution induced by a single reward (objective) function $p_\theta(x) \propto R(x)$ (see Section 5.2.1 for details of GFlowNet formulation). A Multi-Objective GFlowNet aims to learn a single conditional model that can realize distributions corresponding to various combinations (e.g. a convex combination) of multiple reward functions. While a single MOGFN effectively realizes a spectrum of compositions of base reward functions, the approach assumes access to the base rewards at training time. Moreover, MOFGNs require the set of possible composition operations to be specified at generative model training time. In this work, we address post hoc composition of pre-trained GFlowNets (or diffusion models) and provide a way to create compositions that need not be specified in advance.

In this work, we introduce Compositional Sculpting, a general approach for the composition of pre-trained models. We highlight two special examples of binary operations — *harmonic mean*: $(p_1 \otimes p_2)$ and *contrast*: $(p_1 \,◑\, p_2)$. More general compositions are obtained as conditional distributions in a probabilistic model constructed on top of pre-trained base models. We show that these operations can be realized via classifier guidance. We provide results of empirical verification of our method on molecular generation (with GFlowNets) and image generation (with diffusion models).

5.2 Background

5.2.1 Generative Flow Networks (GFlowNets)

GFlowNets [16, 19] are an approach for generating compositional objects (e.g. graphs). The objective of GFlowNet training is specified by a "reward function" $R(x) \geq 0$ defined on the set of objects $\mathcal{X}$. The objective is to learn a generative model $p(x)$ that assigns more probability mass on high-reward objects. Formally, GFlowNets seek to produce the distribution $p(x) = {}^{R(x)}/_{Z}$, where $Z = \sum_x R(x)$.

Generative process. The generation of a complete object x is realized through a sequence of incremental changes of incomplete states $s_0 \to s_1 \to \ldots \to s_{n-1} \to x$ starting at the designated initial state s_0. Formally, the structure of possible generation trajectories $\tau = (s_0 \to s_1 \to \ldots \to s_{n-1} \to x)$ is captured by a DAG $(\mathcal{S}, \mathcal{A})$ where $\mathcal{S}$ is the set of states (both complete and incomplete) and $\mathcal{A}$ is the set of directed edges (actions) $s \to s'$. The set $\mathcal{S}$ has a designated initial state s_0 and the set of complete

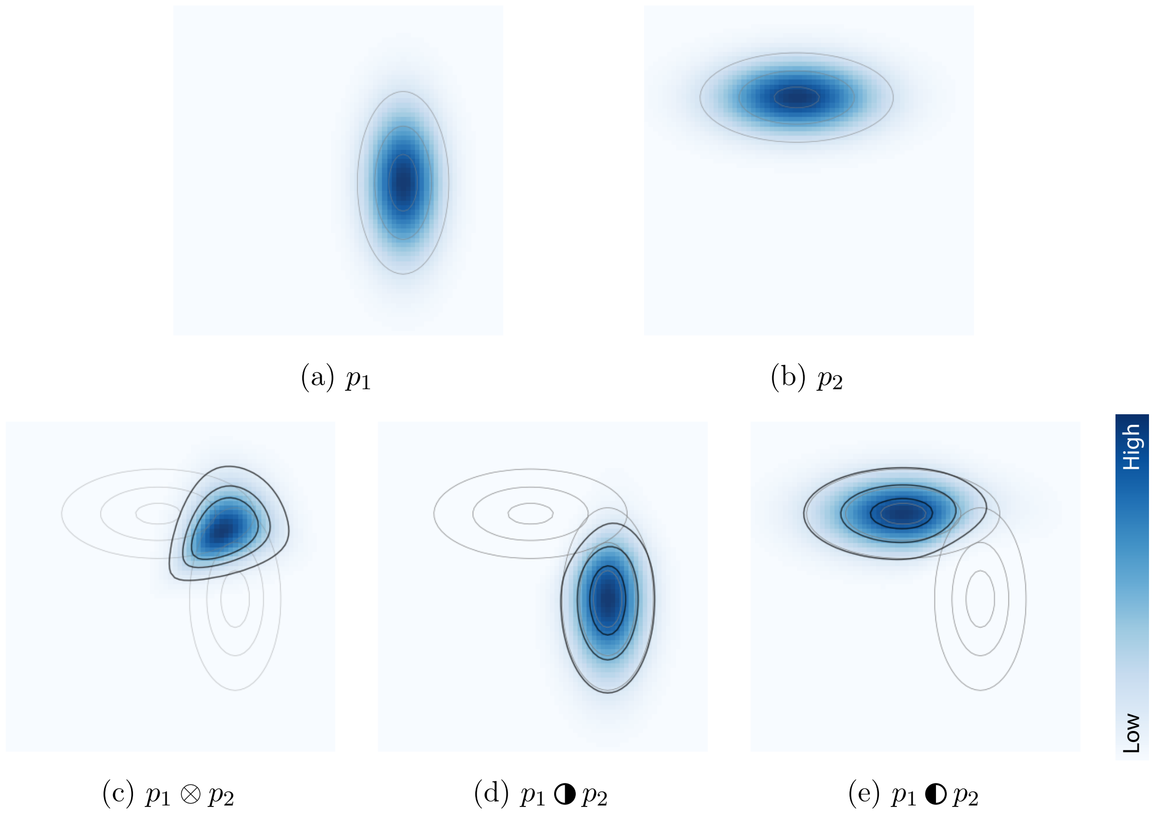


Figure 5-1: **Composition operations.** (a,b) base distributions p_1 and p_2. (c) harmonic mean of p_1 and p_2. (d) contrast of p_1 with p_2 (e) the reverse contrast p_1 ◐ p_2. Note ◑ is asymmetric. Grey lines show contours of PDF level sets of base Gaussian distributions p_1, p_2. Black lines show contours of PDF levels sets of composite distributions.

objects (terminal states) $\mathcal{X}$ is a subset of $\mathcal{S}$. Each generation trajectory τ starts at the initial state s_0, follows the edges $(s \to s') \in \mathcal{A}$ of the DAG, and terminates at one of the terminal states $x \in \mathcal{X}$. We use $|\tau|$ to denote the length of the trajectory (the number of transitions).

This sequential generation process is controlled by a parameterized stochastic "forward policy" $P_F(s'|s;\theta)$ which for each state $s \in \mathcal{S} \setminus \mathcal{X}$ specifies a probability distribution over all possible successor states $s' : (s \to s') \in \mathcal{A}$. Generation is performed by starting at s_0 and sequentially sampling transitions from the forward policy $P_F(\cdot|\cdot)$ until a terminal state is reached.

5.2.2 Diffusion Models

Diffusion models [99, 226, 228, 229, 231] are a family of generative models developed for continuous domains. Given a dataset of samples $\{\hat{x}_i\}_{i=1}^n$ forming the empirical distribution $\hat{p}(x) = \frac{1}{n}\sum_i \delta_{\hat{x}_i}(x)$ in $\mathcal{X} = \mathbb{R}^d$, diffusion models seek to approximate $\hat{p}(x)$ via a generative process $p(x)$, which can then be used to generate new samples.

Stochastic Differential Equation (SDE) perspective. We discuss diffusion models from the perspective of stochastic differential equations (SDE) [231]. A diffusion process is a noising process that gradually destroys the original "clean" data x. It can be specified as a time-indexed collection of random variables $\{x_t\}_{t=0}^T$ in $\mathcal{X} = \mathbb{R}^d$. We use $p_t(\cdot)$ to denote the density of the distribution of x_t. The process interpolates between the data distribution $p_0(x) = \hat{p}(x)$ at $t = 0$, and the prior distribution $p_T(x)$ at $t = T$, which is typically constructed to have a closed form (e.g. standard normal) to enable a simple sampling scheme. The evolution of x_t is described by the "forward SDE" $dx_t = f_t(x_t)\,dt + g_t\,dw_t$, where w_t is the standard Wiener process, the function $f_t : \mathbb{R}^d \to \mathbb{R}^d$ is called the drift coefficient and $g_t \in \mathbb{R}$ is called the diffusion coefficient. Specific choices of f_t and g_t completely determine the process and give rise to the transition kernel $p_{st}(x_t|x_s)$ for $0 \le s < t \le T$ (see [231] for examples).

Generative process. Song et al. [231] invoke a result from the theory of stochastic processes [7] which gives the expression for the reverse-time process or "backward SDE":

$$dx_t = \left[f_t(x_t) - g_t^2 \nabla_x \log p_t(x_t)\right] dt + g_t\, d\overline{w}_t, \tag{5.1}$$

where $\overline{w}_t$ is the standard Wiener process in reversed time.

The backward SDE includes the known coefficients f_t, g_t, and the unknown score function $\nabla_x \log p_t(\cdot)$ of the marginal distribution $p_t(\cdot)$ at time t. This score function is estimated by a deep neural network $s_t(x;\theta) \approx \nabla_x \log p_t(x)$ (called "score-network") with parameters θ. Once the score-network $s_t(\cdot;\theta)$ is trained, samples can be generated via numerical integration of (5.1).

5.2.3 Classifier Guidance in Diffusion Models

Classifier guidance [55, 226] is a technique for controllable generation in diffusion models. Suppose that each example x_0 is accompanied by a discrete class label y. The goal is to sample from the conditional distribution $p_0(x_0|y)$. The Bayes rule $p_t(x_t|y) \propto p_t(x_t)p_t(y|x_t)$ implies the score-function decomposition $\nabla_{x_t} \log p_t(x_t|y) = \nabla_{x_t} \log p_t(x_t) + \nabla_{x_t} \log p_t(y|x_t)$, where the first term is already approximated by a pre-trained unconditional diffusion model and the second term can be derived from a time-dependent classifier $p_t(y|x_t)$. Therefore, the stated goal can be achieved by first training the classifier $p_t(y|x_t)$ using noisy samples x_t from the intermediate steps of the process, and then plugging in the expression for the conditional score into the sampling process (5.1).

5.2.4 "Energy" Operations

Prior work introduced energy operations, "product" and "negation", for energy-based [62, 96] and diffusion [64] models. Given a pair of distributions $p_1(x) \propto \exp\{-E_1(x)\}$, $p_2(x) \propto \exp\{-E_2(x)\}$ corresponding to the respective energy functions E_1 and E_2, the "product" and "negation" operations are defined as

$$(p_1 \,\mathrm{prod}\, p_2)(x) \propto \exp\Big\{-\Big(E_1(x) + E_2(x)\Big)\Big\} \propto p_1(x)p_2(x), \tag{5.2}$$

$$(p_1 \,\mathrm{neg}_\gamma\, p_2)(x) \propto \exp\Big\{-\Big(E_1(x) - \gamma E_2(x)\Big)\Big\} \propto \frac{p_1(x)}{\big(p_2(x)\big)^\gamma}. \tag{5.3}$$

The product distribution $(p_1 \,\mathrm{prod}\, p_2)(x)$: (a) assigns relatively high likelihoods to points x that have sufficiently high likelihoods under both base distributions at the same time; (b) assigns relatively low likelihoods to points x that have close-to-zero likelihood under one (or both) p_1, p_2. The negation distribution $(p_1 \,\mathrm{neg}_\gamma\, p_2)(x)$ (a) assigns relatively high likelihood to points x that are likely under p_1 but unlikely under p_2; (b) assigns relatively low likelihood to points x that have low likelihood under p_1 and high likelihood under p_2. The parameter $\gamma > 0$ controls the strength of negation. Informally, the product concentrates on points that are common in both p_1 and p_2, and the negation concentrates on points that are common in p_1 and uncommon in p_2. If p_1 and p_2 capture objects demonstrating two distinct concepts (e.g. p_1: images of circles; p_2 images of green shapes), it is fair to say (again, informally) that the product and the negation resemble the logical operations of concept-intersection ("circle" `AND` "green") and concept-negation ("circle" `AND NOT` "green") respectively.

The "product" and "negation" can be realized in a natural way in energy-based models through simple algebraic operations on energy functions. However, realizing these operations on diffusion models is not as straightforward. The reason is that sampling in diffusion models requires the coordination of multiple steps of the denoising process. The simple addition of the time-dependent score functions does not result in a score function that represents the diffused product distribution: formally, $\nabla_{x_t} \log \left(\int p_{0t}(x_t|x_0)p_1(x_0)p_2(x_0)\, dx_0\right) \neq \nabla x_t \log \left(\int p_{0t}(x_t|x_0)p_1(x_0)\, dx_0\right) + \nabla x_t \log \left(\int p_{0t}(x_t|x_0)p_2(x_0)\, dx_0\right)$; we refer the reader to [64] for more details on the issue. Du et al. [64] develop a method that corrects the sum-of-scores sampling via additional MCMC iterations nested under each diffusion timestep.

5.3 Related Work

Controllable generation. Generative model composition is a form of post-training control of generative processes, an established area of research in generative modeling. A simple approach to control is conditional generation, which can be achieved by training a conditional generative model $p_\theta(x|c)$ on pairs (x, c) of objects x and conditioning information c. Types of conditioning information can include class labels [55] or more structured data such as text prompts [205, 212, 219], semantic maps, and other images for image-to-image translation [212]. This approach assumes that the generation control operations are specified at training time and the training data is annotated with conditioning attributes. Classifier guidance [226] provides a way to generate samples from conditional distributions that need not be specified at training time. The guidance is realized by a classifier that is trained on examples x_t (both clean and noisy) accompanied by conditioning labels c. Dhariwal and Nichol [55] apply classifier guidance on top of unconditional or conditional diffusion models to improve the fidelity of generated images. Ho and Salimans [98] develop classifier-free guidance where the conditional and unconditional score functions are trained simultaneously and combined at inference time to guide the generation. In ControlNet [274], an additional network is trained to enable a pre-trained diffusion model to incorporate additional, previously unavailable, conditioning information.

Similar to conditional diffusion models, conditional GFlowNets have been used to condition generation on reward exponents [16] or combinations of multiple predefined reward functions [108].

Note that the methods developed in this work can be combined with conditional generative models, for example, conditional diffusion models (or GFlowNets) $p(x|c_1), \dots, p(x|c_m)$ can act as base generative models to be composed.

Image inpainting is an extensively explored controllable generation task. The goal of inpainting is to restore or reconstruct missing or corrupted parts of an image, or, formally, sampling $x_{\text{hidden}}|x_{\text{obs}} \sim p(x_{\text{hidden}}|x_{\text{obs}})$, the unobserved part of the image x_{hidden} given the observed part x_{obs}. Lugmayr et al. [156], Song et al. [231] proposed inpainting techniques based on the "replacement" of observed pixels with their known values (with noise applied at an appropriate) at each step of the diffusion denoising process. Chung et al. [44], Ho et al. [100] utilized "reconstruction"-based approaches which introduce an additional guiding term in the denoising update with the goal of bringing the values in the observed part of the image closer to the known target values. Trippe et al. [244] developed a method based on particle filtering for inpainting of protein backbones in 3D. Saharia et al. [218] proposed a diffusion model trained specifically for image-to-image translation tasks including inpainting.

The task of inpainting is a member of the family of inverse problems, which can be cast as conditional generation $x|y(x) \sim p(x|y(x))$, where $y(x)$ is an observation function that extracts a limited information summary from x (e.g., $y(x)$ might represent a downsampled version of image x). A line of work [42, 118, 207, 218, 231, 259] is focused on diffusion-based solutions for inverse problems in the image generation domain. Recently, Mariani et al. [162] used diffusion models for the inverse problem of audio source separation. Ben-Hamu et al. [14] utilized the probability flow interpretation of diffusion models and addressed image and audio inverse problems and conditional molecular generation via differentiation through ODE sampler.

Image editing is another instance of controllable generation problems. In this case, the goal is to modify specific aspects of an existing image according to user-specified goals while preserving image coherence and fidelity. To address semantic image editing, Meng et al. [166] and Couairon et al. [50] proposed to first partially noise and then denoise an image to generate an edited version, possibly conditioned on a segmentation mask [50]. Collage diffusion [221] is a diffusion model extension that processes multiple base images accompanied by text prompts and desired locations and produces a collage by editing the base images and combining layers of edited images. Another line of work [71, 93, 119, 216] introduced techniques for personalization, concept learning, prompt manipulation, and direct editing of real images guided by natural language inputs.

Fine-tuning diffusion models with reward functions or human preference data has emerged as a promising form of controllable generation. Black et al. [26] interpreted diffusion as a sequential decision process and employed reinforcement learning to optimize objectives such as image compressibility, prompt-image alignment (based on vision-language model feedback), and aesthetic quality (derived from human feedback). Clark et al. [45] used backpropagation through samplers to optimize differentiable rewards such as scores from human preference models. Wallace et al. [254] adapted Direct Preference Optimization [204] to diffusion models, which enables fine-tuning diffusion models on human preference data directly without the need for the auxiliary reward model.

Generative model composition and coordination. Compositional generation approaches combine multiple diffusion processes to control sampling distributions, re-use pre-trained models, and extend their capabilities. Existing works on compositional generation differ in how they approach the composition of diffusion processes. Notable approaches include the resolution of individual steps of generation [11, 49, 103, 142, 162, 275], or exact specification of adjusted distributions [64, 97, 261, and our work].

MultiDiffusion [11] performs controlled image generation via the fusion of diffusion sampling paths. The authors address the generation of panoramic images and scenes with complex structures. The complex scene is divided into several regions, and the desired content of each region is described with a specific text prompt. The generation is performed via the coordination of multiple prompt-conditioned generation paths. The coordination method maintains a shared global image for the whole space and reconciles contradicting updates in the intersections of regions by averaging the updates of individual models whose regions cover a specific intersection. SyncDiffusion [142] is an extension of MultiDiffusion that aims to address the issue of incoherent patches in large panoramic images. To that end, SyncDiffusion introduces an additional step to the MultiDiffusion update to perform a local optimization step on the perceptual similarity across patches. Zhang et al. [275] extended panoramic generation to more general scenarios of large image generation supporting arbitrary graph structure of the overlapping patches comprising the large image (e.g., linear chain, cycle, grid, cube map). The proposed method, called DiffCollage, translates a given graph structure into a closed-form formula for the total score expressed through marginal scores of individual patches. Zhang et al. [275] empirically demonstrated that coordinating multiple diffusion processes run in parallel outperforms inpainting-based panoramic image generation [44, 156] in terms of image quality and generation speed.

Corso et al. [49] proposed Particle Guidance, a method for improving sample efficiency in diffusion models by running multiple diffusion chains with a time-evolving repulsion force that promotes sample diversity.

Mariani et al. [162] demonstrated that the combination of multiple single-instrument diffusion models outperforms a joint model in the music source separation problem. The proposed source separation approach is based on the guidance methods for inverse problems proposed in [231].

Hinton [97] developed a contrastive divergence minimization procedure for training products of tractable energy-based models. Learning mixtures of Generative Adversarial Networks has been addressed in [101], where the mixture components are learned simultaneously, and in [242], where the components are learned one by one in an adaptive boosting fashion. Grover and Ermon [86] developed algorithms for additive and multiplicative boosting of generative models. Following up on energy-based model operations [96, 97], Du et al. [62] studied the composition of deep energy-based models. Du et al. [64] developed algorithms for sampling from energy-based compositions (products, negations) of diffusion models, related to the focus of our work. The algorithm in [64] introduces additional MCMC sampling steps at each diffusion generation step to correct the originally biased sampling process (based on an algebraic combination of individual score functions) toward the target composition. Wu et al. [261] built on compositions of energy-based diffusion models [62, 64] and developed methods for the generation of solutions to inverse design problems: multi-body physical simulations and 2D airfoil design.

Our work proposes a new way to compose pre-trained diffusion models and introduces an unbiased sampling process based on classifier guidance to sample from the compositions. This avoids the need for corrective MCMC sampling required in prior work. Our work further applies to GFlowNets, and is, to the best of our knowledge, the first to address the composition of pre-trained GFlowNets.

This work focuses on the composition of pre-trained models. Assuming that each pre-trained model represents the distribution of examples demonstrating certain concepts (e.g. molecular properties), the composition of models is equivalent to concept composition (e.g. property "A" and property "B" satisfied simultaneously). The inverse problem is known as "unsupervised concept discovery", where the goal is to automatically discover composable concepts from data. Unsupervised concept discovery and concept composition methods have been proposed for energy-based models [63] and for text-to-image diffusion models [151].

Compositional generalization. The notion of compositionality has a broad spectrum of interpretations across a variety of disciplines including linguistics, cognitive science, and philosophy. Hupkes et al. [105] collect a list of aspects of compositionality from linguistical and philosophical theories and designs practical tests for neural language models covering all aspects. Conwell and Ullman [47] empirically examine the relational understanding of DALL-E 2 [206], a text-guided image generation model, and point out limitations in the model's ability to capture relations such as "in", "on", "hanging over", etc. In this work, we focus on a narrow but well-defined type of composition where we seek to algebraically combine (compose) probability densities in a controllable fashion, such that we can emphasize or de-emphasize regions in the data space where specific base distributions have high density. Our methods are developed for the setting where we are given access to GFlowNets or diffusion models which can generate samples from the probability distributions we wish to compose.

Connections between GFlowNets and diffusion models. We develop composition operations and methods for sampling from composite distributions for both GFlowNets and diffusion models. The fact that similar methods apply to both is rooted in deep connections between the two modeling frameworks. GFlowNets were initially developed for generating discrete (structured) data [16] and diffusion models were initially developed for continuous data [99, 226]. Lahlou et al. [134] develop an extension of GFlowNets for DAGs with continuous state-action spaces. Zhang et al. [273] point out unifying connections between GFlowNets and other generative model families, including diffusion models. Diffusion models in a fixed-time discretization can be interpreted as continuous GFlowNets of a certain structure. Zhang et al. [273] notice that the discrete DAG flow-matching condition, central to mathematical foundations of GFlowNets [19], is analogous to the Fokker-Planck equation (Kolmogorov forward equation), underlying mathematical analysis of continuous-time diffusion models [231]. In this work, we articulate another aspect of the relation between GFlowNets and diffusion models: in Section 5.5.2 we derive the expressions for mixture GFlowNet policies and classifier-guided GFlowNet policies analogous to those derived for diffusion models in prior work [55, 148, 192, 226].

5.4 Compositional Sculpting of Generative Models

Consider a scenario where we can access a number of pre-trained generative models. Each of these "base models" gives rise to a generative distribution $p_i(x)$ over a common domain $\mathcal{X}$. We may wish to compose these distributions such that we can, say, draw samples that are likely to arise from $p_1(x)$ and $p_2(x)$, or that are likely to arise from $p_1(x)$ but not from $p_2(x)$. In other words, we wish to specify a composition whose generative distribution we can shape to emphasize and de-emphasize specific base models.

5.4.1 Binary Composition Operations

For the moment, let us focus on controllably composing two base models. One option is to specify the composition as a weighted combination $\widetilde{p}(x) = \sum_{i=1}^{2} \omega_i p_i(x)$ with positive weights ω_1, ω_2 which sum to one. These weights allow us to set the prevalence of each base model in the composition. However, beyond that our control over the composition is limited. We cannot emphasize regions where, say, p_1 and p_2 both have high density, or de-emphasize regions where p_2 has high density.

A much more flexible method for shaping a prior distribution $\widetilde{p}(x)$ to our desires is conditioning. Following Bayesian inference methodology, we know that when we condition x on some observation y, the resulting posterior takes the form $\widetilde{p}(x|y) \propto \widetilde{p}(y|x)\widetilde{p}(x)$. Points x that match observation y according to $\widetilde{p}(y|x)$ will have increased density, whereas the density of points that do not match it decreases. Intuitively, the term $\widetilde{p}(y|x)$ has shaped the prior $\widetilde{p}(x)$ according to y.

If we define the observation $y_1 \in \{1, 2\}$ as the event that x was generated by a specific base model, we can shape the prior based on the densities of the base models. We start by defining a uniform prior over y_k, and by defining the conditional density $p(x|y_1 = i)$ to represent the fact that x was generated from $p_i(x)$. This gives us the following model:

$$\widetilde{p}(x|y_1{=}1) = p_1(x), \quad \widetilde{p}(x|y_1{=}2) = p_2(x), \quad \widetilde{p}(y_1{=}1) = \widetilde{p}(y_1{=}2) = \frac{1}{2}, \tag{5.4}$$

$$\widetilde{p}(x) = \sum_{i=1}^{2} \widetilde{p}(x|y_1\!=\!i)\widetilde{p}(y_1\!=\!i), \tag{5.5}$$

Notice that under this model, the prior $\widetilde{p}(x)$ is simply a uniform mixture of the base models. The posterior probability $\widetilde{p}(y_1\!=\!1|x)$ implied by this model tells us how likely it is that x was generated by $p_1(x)$ rather than $p_2(x)$:

$$\widetilde{p}(y_1\!=\!1|x) = 1 - \widetilde{p}(y_1\!=\!2|x) = \frac{p_1(x)}{p_1(x) + p_2(x)}, \tag{5.6}$$

Note that $\widetilde{p}(y_1 = 1|x)$ is the output of the optimal classifier trained to tell apart distributions $p_1(x)$ and $p_2(x)$.

The goal stated at the beginning of this section was to realize compositions which would generate samples likely to arise from both $p_1(x)$ and $p_2(x)$ or likely to arise from $p_1(x)$ but not $p_2(x)$. To this end we introduce a second observation $y_2 \in \{1, 2\}$ such that y_1 and y_2 are independent and identically distributed given x. The resulting full model and inferred posterior are:

$$\widetilde{p}(x, y_1, y_2) = \widetilde{p}(x)\widetilde{p}(y_1|x)\widetilde{p}(y_2|x), \tag{5.7}$$

$$\widetilde{p}(x) = \frac{1}{2}p_1(x) + \frac{1}{2}p_2(x), \quad \widetilde{p}(y_k\!=\!i|x) = \frac{p_i(x)}{p_1(x) + p_2(x)}, \; k, i \in \{1, 2\}, \tag{5.8}$$

$$\widetilde{p}(x|y_1\!=\!i, y_2\!=\!j) \propto \widetilde{p}(x)\widetilde{p}(y_1\!=\!i|x)\widetilde{p}(y_2\!=\!j|x) \propto \frac{p_i(x)p_j(x)}{p_1(x) + p_2(x)}. \tag{5.9}$$

The posterior $\widetilde{p}(x|y_1\!=\!i, y_2\!=\!j)$ shows clearly how conditioning on the observations y_1, y_2 has shaped the prior mixture into a new expression which accentuates regions in the posterior where the observed base models i, j have high density.

Conditioning on observations $y_1\!=\!1$ ("x is likely to have been drawn from p_1 rather than p_2") and $y_2 = 2$ ("x is likely to have been drawn from p_2 rather than p_1"), or equivalently $y_1\!=\!2, y_2\!=\!1$, results in the posterior distribution

$$(p_1 \otimes p_2)(x) := p(x|y_1\!=\!1, y_2\!=\!2) \propto \frac{p_1(x)p_2(x)}{p_1(x) + p_2(x)}. \tag{5.10}$$

We will refer to this posterior as the "harmonic mean of p_1 and p_2", and denote it as a binary operation $p_1 \otimes p_2$. Its value is high only at points that have high likelihood under both $p_1(x)$ and $p_2(x)$ at the same time (Figure 5-1(c)). Thus, the harmonic mean is an alternative to the product operation for EBMs. The harmonic mean is commutative ($p_1 \otimes p_2 = p_2 \otimes p_1$) and is undefined when p_1 and p_2 have disjoint supports, since then the RHS of (5.10) is zero everywhere.

Conditioning on observations $y_1{=}1$ ("x is likely to have been drawn from p_1 rather than p_2") and $y_2{=}1$ (same) results in the posterior distribution

$$(p_1 \mathbin{\RIGHTcircle} p_2)(x) := \widetilde{p}(x|y_1{=}1, y_2{=}1) \propto \frac{\big(p_1(x)\big)^2}{p_1(x) + p_2(x)}. \tag{5.11}$$

We refer to this operation, providing an alternative to the negation operation in EBMs, as the "contrast of p_1 and p_2", and will denote it as a binary operator $(p_1 \mathbin{\RIGHTcircle} p_2)(x)$. The ratio in equation (5.11) is strictly increasing as a function of $p_1(x)$ and strictly decreasing as a function of $p_2(x)$, so that the ratio is high when $p_1(x)$ is high and $p_2(x)$ is low (Figure 5-1(d)). The contrast is not commutative ($p_1 \mathbin{\RIGHTcircle} p_2 \neq p_2 \mathbin{\RIGHTcircle} p_1$, unless $p_1 = p_2$). We will denote the reverse contrast as $p_1 \mathbin{\LEFTcircle} p_2 = p_2 \mathbin{\RIGHTcircle} p_1$.

Note that the original distributions p_1 and p_2 can be expressed as mixtures of the harmonic mean and the contrast distributions:

$$p_1 = Z_{\otimes}(p_1 \otimes p_2) + Z_{\RIGHTcircle}(p_1 \mathbin{\RIGHTcircle} p_2), \quad p_2 = Z_{\otimes}(p_1 \otimes p_2) + Z_{\RIGHTcircle}(p_2 \mathbin{\RIGHTcircle} p_1),$$
$$Z_{\otimes} = \sum_x \frac{p_1(x)p_2(x)}{p_1(x) + p_2(x)} = 1 - Z_{\RIGHTcircle}.$$

Controlling the individual contributions of p_1 and p_2 to the composition. We modify model (5.7) and introduce an interpolation parameter α in order to have more control over the extent of individual contributions of p_1 and p_2 to the composition:

$$\widetilde{p}(x, y_1, y_2; \alpha) = \widetilde{p}(x)\widetilde{p}(y_1|x)\widetilde{p}(y_2|x; \alpha), \quad \widetilde{p}(x) = \frac{1}{2}p_1(x) + \frac{1}{2}p_2(x), \tag{5.12a}$$

$$\widetilde{p}(y_1{=}i|x) = \frac{p_i(x)}{p_1(x) + p_2(x)}, \tag{5.12b}$$

$$\widetilde{p}(y_2 = i | x; \alpha) = \frac{\big(\alpha p_1(x)\big)^{[i=1]} \cdot \big((1-\alpha) p_2(x)\big)^{[i=2]}}{\alpha p_1(x) + (1-\alpha) p_2(x)}, \tag{5.12c}$$

where $\alpha \in (0, 1)$ and $[\cdot]$ denotes the indicator function. Conditional distributions in this model give **harmonic interpolation**[1] and **parameterized contrast**:

$$(p_1 \otimes_{(1-\alpha)} p_2)(x) \propto \frac{p_1(x) p_2(x)}{\alpha p_1(x) + (1-\alpha) p_2(x)}, \tag{5.13}$$

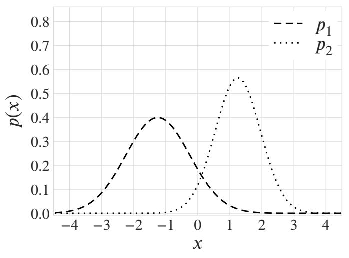
$$(p_1 \RIGHTcircle_{(1-\alpha)} p_2)(x) \propto \frac{\big(p_1(x)\big)^2}{\alpha p_1(x) + (1-\alpha) p_2(x)}. \tag{5.14}$$

Operation chaining. As the binary operations we have introduced result in proper distributions, we can create new N-ary operations by chaining binary (and N-ary) operations together. For instance, chaining binary harmonic means gives the harmonic mean of three distributions

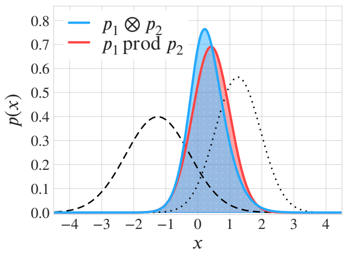
$$\begin{aligned}((p_1 \otimes p_2) \otimes p_3)(x) &= (p_1 \otimes (p_2 \otimes p_3))(x) \\ &\propto \frac{p_1(x) p_2(x) p_3(x)}{p_1(x) p_2(x) + p_1(x) p_3(x) + p_2(x) p_3(x)}.\end{aligned} \tag{5.15}$$

Comparison with "energy" operations. The harmonic mean and contrast operations we have introduced here are analogous to the product and negation operations for EBMs respectively. Although the harmonic mean and product operations are quite similar in practice, unlike the negation operation our proposed contrast operation always results in a valid probability distribution. Figure 5-2 shows the results of these operations applied to two Gaussian distributions. The harmonic mean and product, shown in panel (b), are both concentrated on points that have high probability under both Gaussians. Figure 5-2(c) shows parameterized contrasts $p_1 \RIGHTcircle_{(1-\alpha)} p_2$ at different values of α, and panel (d) shows negations $p_1 \operatorname{neg}_\gamma p_2$ at different values of γ. The effect of negation at $\gamma = 0.1$ resembles the effect of the contrast operation: the density retreats from the high likelihood region of p_2. However, as γ increases to 0.5 the distribution starts to concentrate excessively on the values $x < -3$. This is

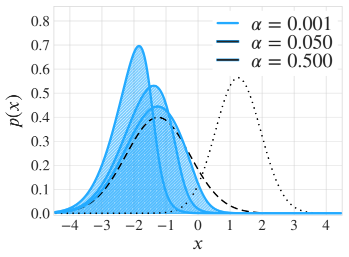
[1]the harmonic interpolation approaches p_1 when $\alpha \to 0$ and p_2 when $\alpha \to 1$



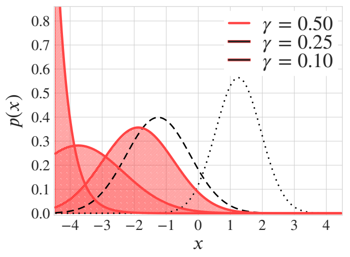
(a) Base distributions p_1, p_2



(b) HM (ours), Product



(c) Contrasts $p_1 ◑_{(1-\alpha)} p_2$ (ours)



(d) Negations $p_1 \operatorname{neg}_\gamma p_2$

Figure 5-2: **Compositional sculpting and energy operations applied to 1D Gaussian distributions.** (a) Densities of base 1D Gaussian distributions $p_1(x) = \mathcal{N}(x; -5/4, 1)$, $p_2(x) = \mathcal{N}(x; 5/4, 1/2)$. (b) harmonic mean $p_1 \otimes p_2$ and product $p_1 \operatorname{prod} p_2$ (c) parameterized contrasts $p_1 ◑_{(1-\alpha)} p_2$ at different values of α (d) negations $p_1 \operatorname{neg}_\gamma p_2$ at different values of γ. The curves show the probability density functions of base distributions and their compositions.

due to the instability of division $p_1(x)/(p_2(x))^\gamma$ in regions where $p_2(x) \to 0$. Below, we formally discuss the properties of compositional sculpting operations and energy-based operations.

Harmonic mean and product are not defined for pairs of distributions p_1, p_2 which have disjoint supports. In such cases, attempts at evaluation of the expressions for $p_1 \otimes p_2$ and $p_1 \operatorname{prod} p_2$ will lead to impossible probability distributions that have zero probability mass (density) everywhere[2]. The result of both harmonic mean and product are correctly defined for any pair of distributions p_1, p_2 that have non-empty support intersection.

Notably, contrast is well-defined for any input distributions while negation is ill-defined for some input distributions p_1, p_2 as formally stated below (see Figure 5-2 (d) for a concrete example).

[2]Informal interpretation: distributions with disjoint supports have empty "intersections" (think of the intersection of sets analogy)

Proposition 5.4.1.

1. *For any $\alpha \in (0,1)$ the parameterized contrast operation $p_1 \,◑_{(1-\alpha)}\, p_2$ (5.13) is well-defined: gives a proper distribution for any pair of distributions p_1, p_2.*

2. *For any $\gamma \in (0,1)$ there are infinitely many pairs of distributions p_1, p_2 such that the negation $p_1 \operatorname{neg}_\gamma p_2$ (5.3) results in an improper (non-normalizable) distribution.*

5.4.2 Compositional Sculpting: General Approach

The approach we used above for specifying compositions of two base models controlled by two observations can be generalized to compositions of m base models $p_1(x), \ldots, p_m(x)$ controlled by n observations. At the end of the previous section we showed that operator chaining can already realize compositions of m base models. However, our generalized method allows us to specify compositions more flexibly, and results in different compositions from operator chaining. We propose to define an augmented probabilistic model $\widetilde{p}(x, y_1, \ldots, y_n)$ as a joint distribution over the original objects $x \in \mathcal{X}$ and n observation variables $y_1 \in \mathcal{Y}, \ldots, y_n \in \mathcal{Y}$ where $\mathcal{Y} = \{1, \ldots, m\}$. By defining appropriate conditionals $p(y_k|x)$ we will be able to controllably shape a prior $\widetilde{p}(x)$ into a posterior $\widetilde{p}(x|y_1, \ldots, y_n)$ based on the base models.

As in the binary case, we propose to use a uniformly-weighted mixture of the base distributions $\widetilde{p}(x) = \frac{1}{m}\sum_{i=1}^m p_i(x)$. The support of this mixture is the union of the supports of the base models: $\bigcup_{i=1}^m \operatorname{supp}\{p_i(x)\} = \operatorname{supp}\{\widetilde{p}(x)\}$. This is essential as the prior can only be shaped in places where it has non-zero density. As before we define the conditionals $p(y_k\!=\!i|x)$ to correspond to the observation that x was generated by base model i. This resulting full model is

$$\widetilde{p}(x, y_1, \ldots, y_n) = \widetilde{p}(x)\prod_{k=1}^n \widetilde{p}(y_k|x), \qquad \widetilde{p}(x) = \frac{1}{m}\sum_{i=1}^m p_i(x), \tag{5.16}$$

$$\widetilde{p}(y_k\!=\!i) = \frac{1}{m}\ \forall k \in \{1, \ldots, n\}, \quad \widetilde{p}(y_k\!=\!i|x) = \frac{p_i(x)}{\sum_{j=1}^m p_j(x)}\ \forall k \in \{1, \ldots, n\}. \tag{5.17}$$

Note that under this model the mixture can be represented as the marginal of the joint distribution $\widetilde{p}(x, y_k) = \widetilde{p}(x|y_k)\widetilde{p}(y_k)$ where $y \in \{1, \ldots, m\}$ for any one of the observations y_k.

The inferred posterior over x for this model is

$$\begin{aligned}
\widetilde{p}(x|y_1{=}i_1, \ldots, y_n{=}i_n) &\propto \widetilde{p}(x)\widetilde{p}(y_1{=}i_1, \ldots, y_n{=}i_n|x) && (5.18)\\
&\propto \widetilde{p}(x)\prod_{k=1}^{n}\widetilde{p}(y_k{=}i_k|x) && (5.19)\\
&\propto \left(\prod_{k=1}^{n} p_{i_k}(x)\right) \Bigg/ \left(\sum_{j=1}^{m} p_j(x)\right)^{n-1}. && (5.20)
\end{aligned}$$

The posterior $\widetilde{p}(x|y_1{=}i_1, \ldots, y_n{=}i_n)$ is a composition of distributions $\{p_i(x)\}_{i=1}^m$ that can be adjusted by choosing values for $y_1, \ldots, y_n$. By adding or omitting an observation $y_k = i$ we can *sculpt* the posterior to our liking, emphasizing or de-emphasizing regions of $\mathcal{X}$ where p_i has high density. The observations can be introduced with multiplicities (e.g., $y_1 = 1, y_2 = 1, y_3 = 2$) to further strengthen the effect. Moreover, one can choose to introduce all observations simultaneously as in (5.18) or sequentially as in (5.19). As we show below (Section 5.5.1 for GFlowNets; Section 5.5.3 for diffusion models), the composition (5.18) can be realized by a sampling policy that can be expressed as a function of the pre-trained (base) sampling policies.

Special instances and general formulation. The general approach outlined in this section is not limited to choices we made to construct the model in equation (5.16), i.e. $\widetilde{p}(x)$ does not have to be a uniformly weighted mixture of the base distributions, $y_1, \ldots, y_n$ do not have to be independent and identically distributed given x, and different choices of the likelihood $\widetilde{p}(y{=}i|x)$ are possible. For instance, the parameterized binary operations (5.13) are derived from a model where the likelihoods of the observations $\widetilde{p}(y_1|x)$, $\widetilde{p}(y_2|x)$ differ.

Sampling from conditional distributions via classifier guidance. In Section 5.5 we introduce a method that allows us to sample from compositions of distributions

$p_1, \ldots, p_m$ implied by a chosen set of variables $y_1, \ldots, y_n$. To do this, we note the similarity between (5.18) and classifier guidance. Indeed, we can sample from the posterior by applying classifier guidance to $\widetilde{p}(x)$. Classifier guidance can either be applied in a single shot as in (5.18), or sequentially as in (5.19). Any chain of operations can be realized via sequential guidance with a new classifier trained at each stage of the chaining. The classifier can be trained on samples generated from the pre-trained base models $p_1, \ldots, p_m$. We show how to apply this idea to GFlowNets (Sections 5.5.1, 5.5.2) and diffusion models (Sections 5.5.3, 5.5.4).

5.5 Compositional Sculpting of Iterative Generative Processes

5.5.1 Composition of GFlowNets

We will now cover how the model above can be applied to compose GFlowNets, and how one can use classifier guidance to sample from the composition. Besides a sample x from $p_i(x)$, a GFlowNet also generates a trajectory τ which ends in the state x. Thus, we extend the model $\widetilde{p}(x, y_1, \ldots, y_n)$, described above, and introduce τ as a variable with conditional distribution $\widetilde{p}(\tau | y_k = i) = \prod_{t=0}^{|\tau|-1} p_{i,F}(s_{t+1}|s_t)$, where $p_{i,F}$ is the forward policy of the GFlowNet that samples from p_i.

Our approach for sampling from the composition is conceptually simple. Given m base GFlowNets that sample from $p_1, \ldots, p_m$ respectively, we start by defining the prior $\widetilde{p}(x)$ as the uniform mixture of these GFlowNets. Proposition 5.5.1 shows that this mixture can be realized by a GFlowNet policy which can be constructed directly from the forward policies of the base GFlowNets. We then apply classifier guidance to this mixture to sample from the composition. Proposition 5.5.2 shows that classifier guidance results in a new GFlowNet policy which can be constructed directly from the GFlowNet being guided.

Proposition 5.5.1 (GFlowNet mixture policy). *Suppose distributions $p_1(x), \ldots, p_m(x)$ are realized by GFlowNets with forward policies $p_{1,F}(\cdot|\cdot), \ldots, p_{m,F}(\cdot|\cdot)$. Then, the* *mixture distribution* $p_M(x) = \sum_{i=1}^m \omega_i p_i(x)$ *with* $\omega_1, \ldots, \omega_m \geq 0$ *and* $\sum_{i=1}^m \omega_i = 1$ *is realized by the GFlowNet forward policy*

$$p_{M,F}(s'|s) = \sum_{i=1}^{m} p(y = i|s) p_{i,F}(s'|s), \tag{5.21}$$

where y *is a random variable such that the joint distribution of a GFlowNet trajectory* τ *and* y *is given by* $p(\tau, y{=}i) = \omega_i p_i(\tau)$ *for* $i \in \{1, \ldots, m\}$.

The proof of Proposition 5.5.1 is provided in Appendix C.3.1.

Proposition 5.5.2 (GFlowNet classifier guidance)**.** *Consider a joint distribution* $p(x, y)$ *over a discrete space* $\mathcal{X} \times \mathcal{Y}$ *such that the marginal distribution* $p(x)$ *is realized by a GFlowNet with forward policy* $p_F(\cdot|\cdot)$*. Further, assume that the joint distribution of* x*,* y*, and GFlowNet trajectories* $\tau = (s_0 \to \ldots \to s_n = x)$ *decomposes as* $p(\tau, x, y) = p(\tau, x)p(y|x)$*, i.e.* y *is independent of the intermediate states* $s_0, \ldots, s_{n-1}$ *in* τ *given* x*. Then,*

1. *For all non-terminal nodes* $s \in \mathcal{S} \setminus \mathcal{X}$ *in the GFlowNet DAG* $(\mathcal{S}, \mathcal{A})$*, the probabilities* $p(y|s)$ *satisfy*

$$p(y|s) = \sum_{s' : (s \to s') \in \mathcal{A}} p_F(s'|s) p(y|s'). \tag{5.22}$$

2. *The conditional distribution* $p(x|y)$ *is realized by the classifier-guided policy*

$$p_F(s'|s, y) = p_F(s'|s) \frac{p(y|s')}{p(y|s)}. \tag{5.23}$$

Note that (5.22) ensures that $p_F(s'|s, y)$ is a valid policy, i.e. $\sum_{s' : (s \to s') \in \mathcal{A}} p_F(s'|s, y) = 1$. The proof of Proposition 5.5.2 is provided in Appendix C.3.2.

Proposition 5.5.1 is an analogous to results on mixtures of diffusion models (Peluchetti 192, Theorem 1, Lipman et al. 148, Theorem 1). Proposition 5.5.2 is analogous to classifier guidance for diffusion models [55, 226]. To the best of our knowledge, our work is the first to derive both results for GFlowNets.

Both equations (5.21) and (5.23) involve the inferential distribution $p(y|s)$. Practical implementations of both mixture and conditional forward policies, therefore, require training a classifier on trajectories sampled from the given GFlowNets.

Theorem 5.5.3 summarizes our approach.

Theorem 5.5.3. *Suppose distributions $p_1(x), \ldots, p_m(x)$ are realized by GFlowNets with forward policies $p_{1,F}(\cdot|\cdot)$, ..., $p_{m,F}(\cdot|\cdot)$ respectively. Let $y_1, \ldots, y_n$ be random variables defined by* (5.16)*. Then, the conditional $\widetilde{p}(x|y_1, \ldots, y_n)$ is realized by the forward policy*

$$p_F(s'|s, y_1, \ldots, y_n) = \frac{\widetilde{p}(y_1, \ldots, y_n|s')}{\widetilde{p}(y_1, \ldots, y_n|s)} \sum_{i=1}^{m} p_{i,F}(s'|s)\widetilde{p}(y{=}i|s) \qquad (5.24)$$

Note that the result of conditioning on observations $y_1, \ldots, y_n$ is just another GFlowNet policy. Therefore, to condition on more observations and build up the composition further, we can simply apply classifier guidance again to the policy constructed in Theorem 5.5.3.

5.5.2 Classifier Training (GFlowNets)

The evaluation of policy (5.24) requires knowledge of the probabilities $\widetilde{p}(y_1, \ldots, y_n|s)$. The probabilities $\widetilde{p}(y|s)$ required for constructing the mixture can be derived from $\widetilde{p}(y_1, \ldots, y_n|s)$. These probabilities can be estimated by a classifier fitted to trajectories sampled from the base GFlowNets $p_1, \ldots, p_m$. Below we specify the sampling scheme and the objective for this classifier.

Let $\widetilde{Q}_\phi(y_1, \ldots, y_n|s)$ be a classifier with parameters ϕ that we wish to train to approximate the ground-truth conditional: $\widetilde{Q}_\phi(y_1, \ldots, y_n|s) \approx \widetilde{p}(y_1, \ldots, y_n|s)$. Note that $\widetilde{Q}_\phi$ represents the joint distribution of $y_1, \ldots, y_n$ given a state s. Under the model (5.16) the variables $y_1, \ldots, y_n$ are dependent given a state $s \in \mathcal{S} \setminus \mathcal{X}$, but, are independent given a terminal state $x \in \mathcal{X}$. This observation motivates separate treatment of terminal and non-terminal states.

Learning the terminal state classifier. For a terminal state x, the variables $y_1, \ldots, y_n$ are independent and identically distributed. Hence we can use the factorization $\widetilde{Q}_\phi(y_1, \ldots, y_n|x) = \prod_{k=1}^n \widetilde{Q}_\phi(y_k|x)$. Moreover, all distributions on the *r.h.s.* must be the same. In other words, for the terminal classifier it is, therefore, enough to learn just $\widetilde{Q}_\phi(y_1|x)$. This marginal classifier can be learned by minimizing the cross-entropy loss

$$\mathcal{L}_\mathrm{T}(\phi) = \mathop{\mathbb{E}}_{(\widehat{x},\widehat{y}_1)\sim\widetilde{p}(x,y_1)} \left[-\log \widetilde{Q}_\phi(y_1 = \widehat{y}_1 | x = \widehat{x}) \right]. \tag{5.25}$$

Sampling from $\widetilde{p}(x, y_1)$ can be performed according to the factorization $\widetilde{p}(y_1)\widetilde{p}(x|y_1)$. First, $\widehat{y}_1$ is sampled from $\widetilde{p}(y_1)$, which is uniform under our choice of $\widetilde{p}(x)$. Then, $\widehat{x}|(y_1 = \widehat{y}_1)$ is generated from the base GFlowNet $p_{\widehat{y}_1}$. For our choice of $\widetilde{p}(x)$, we can derive from (5.16) that $\widetilde{p}(x|y = \widehat{y}_1) = p_{\widehat{y}_1}(x)$.

Learning the non-terminal state classifier. Given a non-terminal state $s \in \mathcal{S} \setminus \mathcal{X}$, we need to model $y_1, \ldots, y_n$ jointly. In order to train the classifier one needs to sample tuples $(\widehat{s}, \widehat{y}_1, \ldots, \widehat{y}_n)$. Non-terminal states s can be generated as intermediate states in trajectories $\tau = (s_0 \to s_1 \to \ldots \to x)$. Given a sampled trajectory $\widehat{\tau}$ and a set of labels $\widehat{y}_1, \ldots, \widehat{y}_n$ we denote the total cross-entropy loss of all non-terminal states in $\widehat{\tau}$

Algorithm 2 Compositional Sculpting: classifier training

```
1: Initialize $\phi$ and set $\overline{\phi} = \phi$
2: for step = $1, \ldots,$ num_steps do
3:     for $i = 1, \ldots, m$ do
4:         Sample $\widehat{\tau}_i \sim p_i(\tau)$
5:     end for
6:     $\mathcal{L}_T(\phi) = -\sum_{i=1}^{m} \log \widetilde{Q}_\phi(y_1 = i | x = \widehat{x}_i)$    {Terminal state loss (5.25)}
7:     $w_i(\widehat{x}_j; \overline{\phi}) = \widetilde{Q}_{\overline{\phi}}(y_k = i | x = \widehat{x}_j),\ \ i, j \in \{1, \ldots m\}$    {Probability estimates}
8:     $\mathcal{L}_N(\phi, \overline{\phi}) = \sum_{\widehat{y}_1=1}^{m} \ldots \sum_{\widehat{y}_n=1}^{m} \left( \prod_{k=2}^{n} w_{\widehat{y}_k}(\widehat{x}_{\widehat{y}_1}; \overline{\phi}) \right) \ell(\widehat{\tau}_{\widehat{y}_1}, \widehat{y}_1, \ldots \widehat{y}_n; \phi)$    {Non-terminal
       state loss (5.26)-(5.27)}
9:     $\mathcal{L}(\phi, \overline{\phi}) = \mathcal{L}_T(\phi) + \gamma(\text{step}) \cdot \mathcal{L}_N(\phi, \overline{\phi})$
10:    Update $\phi$ using $\nabla_\phi \mathcal{L}(\phi, \overline{\phi})$; update $\overline{\phi} = \beta\overline{\phi} + (1 - \beta)\phi$
11: end for
```

by

$$\ell(\widehat{\tau}, \widehat{y}_1, \dots, \widehat{y}_n; \phi) = \sum_{t=0}^{|\tau|-1} \left[-\log \widetilde{Q}_\phi(y_1 = \widehat{y}_1, \dots, y_n = \widehat{y}_n | s = \widehat{s}_t) \right]. \tag{5.26}$$

The pairs $(\widehat{\tau}, \widehat{y}_1)$ can be generated via a sampling scheme similar to the one used for the terminal state classifier loss above: 1) $\widehat{y}_1 \sim \widetilde{p}(y_1)$ and 2) $\widehat{\tau} \sim p_{\widehat{y}_1}(\tau)$. Sampling $\widehat{y}_2, \dots, \widehat{y}_n$ given $\widehat{\tau}$ (and $\widehat{x}$, the terminal state of $\widehat{\tau}$) requires access to the values $\widetilde{p}(y_k = \widehat{y}_k | \widehat{x})$, but these are not directly available. However, if the terminal classifier is learned as described above, the estimates $w_i(\widehat{x}; \phi) = \widetilde{Q}_\phi(y_1 = i | x = \widehat{x})$ can be used instead.

We described a training scheme where the loss and the sampling procedure for the non-terminal state classifier rely on the estimates produced by the terminal state classifier. In principle, one could use a two-phase procedure by first learning the terminal state classifier, and then learning the non-terminal state classifier using the estimates provided by the fixed terminal state classifier. However, it is possible to train both classifiers in one run, provided that we address potential training instabilities due to the feedback loop between the non-terminal and terminal classifiers. We employ the "target network" technique developed in the context of deep Q-learning [172]. We introduce a "target network" parameter vector $\overline{\phi}$ which is used to produce the estimates $w_i(\widehat{x}; \overline{\phi})$ for the non-terminal state loss. We update $\overline{\phi}$ as the exponential moving average of the recent iterates of ϕ.

After putting all components together the training loss for the non-terminal state classifier is

$$\mathcal{L}_N(\phi, \overline{\phi}) = \mathop{\mathbb{E}}_{(\widehat{\tau}, \widehat{y}_1) \sim \widetilde{p}(\tau, y_1)} \left[\sum_{\widehat{y}_2=1}^{m} \cdots \sum_{\widehat{y}_n=1}^{m} \left(\prod_{k=2}^{n} w_{\widehat{y}_k}(\widehat{x}; \overline{\phi}) \right) \ell(\widehat{\tau}, \widehat{y}_1, \dots, \widehat{y}_n; \phi) \right]. \tag{5.27}$$

We refer the reader to Appendix C.3.4 for a more detailed derivation of the loss (5.27).

Note that equation (5.27) involves summation over $\widehat{y}_2, \dots \widehat{y}_n$ with m^{n-1} terms in the sum. If values of n and m are small, the sum can be evaluated directly. In general, one could trade off estimation accuracy for improved speed by replacing the summation with Monte Carlo estimation. In this case, the values $\widehat{y}_k$ are sampled from the categorical distributions $Q_{\overline{\phi}}(y|x)$. Note that labels can be sampled in parallel since y_i are independent given x.

Algorithm 2 shows the complete classifier training procedure.

5.5.3 Composition of Diffusion Models

In this section, we show how the method introduced above can be applied to diffusion models. First, we adapt the model we introduced in (5.16)-(5.19) to diffusion models. A diffusion model trained to sample from $p_i(x)$ generates a trajectory $\tau = \{x_t\}_{t=0}^{T}$ over a range of time steps which starts with a randomly sampled state x_T and ends in x_0, where x_0 has distribution $p_{i,t=0}(x) = p_i(x)$. Thus, we must adapt our model to reflect this. We introduce a set of mutually dependent variables x_t for $t \in (0, T]$ with as conditional distribution the transition kernel of the diffusion model $p_i(x_t|x_0)$.

Given m base diffusion models that sample from $p_1, \dots, p_m$ respectively, we define the prior $\widetilde{p}(x)$ as a mixture of these diffusion models. Proposition 5.5.4 shows that this mixture is a diffusion model that can be constructed directly from the base diffusion models. We then apply classifier guidance to this mixture to sample from the composition. We present an informal version of the proposition below. The required assumptions and the proof are provided in Appendix C.3.5.

Proposition 5.5.4 (Diffusion mixture SDE). *Suppose distributions $p_1(x), \dots, p_m(x)$ are realized by diffusion models with forward SDEs $dx_{i,t} = f_{i,t}(x_{i,t})\,dt + g_{i,t}\,dw_{i,t}$ and score functions $s_{i,t}(\cdot)$, respectively. Then, the mixture distribution $p_M(x) = \sum_{i=1}^{m} \omega_i p_i(x)$ with $\omega_1 \dots \omega_m \geq 0$ and $\sum_{i=1}^{m} \omega_i = 1$ is realized by a diffusion model with forward SDE*

$$dx_t = \underbrace{\left[\sum_{i=1}^{m} p(y{=}i|x_t) f_{i,t}(x_t)\right]}_{f_{M,t}(x_t)} dt + \underbrace{\sqrt{\sum_{i=1}^{m} p(y{=}i|x_t) g_{i,t}^2}}_{g_{M,t}(x_t)}\, dw_t, \tag{5.28}$$

and backward SDE

$$dx_t = \left[\sum_{i=1}^{m} p(y{=}i|x_t)\Big(f_{i,t}(x_t) - g_{i,t}^2 s_{i,t}(x_t)\Big)\right] dt + \sqrt{\sum_{i=1}^{m} p(y{=}i|x_t) g_{i,t}^2}\, d\overline{w}_t, \quad (5.29)$$

with

$$p(y{=}i|x_t) = \frac{\omega_i p_{i,t}(x_t)}{\sum_{j=1}^{m} \omega_j p_{j,t}(x_t)}. \quad (5.30)$$

If the base diffusion models have a common forward SDE $dx_{i,t} = f_t(x_{i,t})\, dt + g_t\, dw_{i,t}$, equations (5.28)-(5.29) simplify to

$$dx_t = f_t(x_t)dt + g_t dw_t,\ dx_t = \left[f_t(x_t) - g_t^2\left(\sum_{i=1}^{m} p(y{=}i|x_t) s_{i,t}(x_t)\right)\right] dt + g_t d\overline{w}_t. \quad (5.31)$$

Theorem 5.5.5 summarizes the overall approach.

Theorem 5.5.5. *Suppose distributions $p_1(x), \ldots, p_m(x)$ are realized by diffusion models with forward SDEs $dx_{i,t} = f_{i,t}(x_{i,t})\, dt + g_{i,t}\, dw_{i,t}$ and score functions $s_{i,t}(\cdot)$, respectively. Let $y_1, \ldots y_n$ be random variables defined by* (5.16)*. Then, the conditional $\widetilde{p}(x|y_1, \ldots, y_n)$ is realized by a classifier-guided diffusion with backward SDE*

$$dx_t = \left[\sum_{i=1}^{m} \widetilde{p}(y{=}i|x_t)\Big(f_{i,t}(x_t) - g_{i,t}^2\Big(s_{i,t}(x_t) + \nabla_{x_t} \log \widetilde{p}(y_1, \ldots, y_n|x_t)\Big)\Big)\right] dt + \sqrt{\sum_{i=1}^{m} \widetilde{p}(y{=}i|x_t) g_{i,t}^2}\ d\overline{w}_t. \quad (5.32)$$

The proof of Theorem 5.5.5 is provided in Appendix C.3.6.

5.5.4 Classifier Training (Diffusion Models)

We approximate the inferential distributions in equations (5.31) and (5.32) with a time-conditioned classifier $\widetilde{Q}_\phi(y_1, \ldots, y_n|x_t)$ with parameters ϕ. Contrary to GFlowNets, which employed a terminal and non-terminal state classifier, here we only need a single time-dependent classifier. The classifier is trained with different objectives on terminal and non-terminal states. The variables $y_1, \ldots, y_n$ are dependent given a state x_t for $t \in [0, T)$, but are independent given the terminal state x_T. Thus, when training on terminal states we can exploit this independence. Furthermore, we generally found it beneficial to initially train only on terminal states. The loss for the non-terminal states depends on classifications of the terminal state of the associated trajectories, thus by minimizing the classification error of terminal states first, we reduce noise in the loss calculated for the non-terminal states later.

For a terminal state x_0, the classifier $\widetilde{Q}_\phi(y_1, \ldots, y_n | x_t)$ can be factorized as $\prod_{k=1}^n \widetilde{Q}_\phi(y_k | x_0)$. Hence we can train $\widetilde{Q}$ by minimizing the cross-entropy loss

$$\mathcal{L}_{\mathrm{T}}(\phi) = \mathop{\mathbb{E}}_{(\widehat{x}_0, \widehat{y}_1) \sim \widetilde{p}(x, y_1)} \left[-\log \widetilde{Q}_\phi(y_1 {=} \widehat{y}_1 | x_0 {=} \widehat{x}_0) \right]. \tag{5.33}$$

Samples $\widetilde{p}(x_0, y_1)$ can be generated according to the factorization $\widetilde{p}(y_1)\widetilde{p}(x_0|y_1)$. First, $\widehat{y}_1$ is sampled from $\widetilde{p}(y_1)$, which is uniform under our choice of $\widetilde{p}(x_0)$. Then, $\widehat{x}_0|(y_1 {=} \widehat{y}_1)$ is generated from the reverse SDE of base diffusion model $p_{\widehat{y}_1}(x)$. Note that equation (5.17) implies that all observations have the same conditional distribution given x. Thus, $\widetilde{Q}_\phi(y_1|x_0)$ is also a classifier for observations $y_2, \ldots, y_n$.

For a non-terminal state x_t with $t \in (0, T]$, we must train $\widetilde{Q}$ to predict $y_1, \ldots, y_n$ jointly. For a non-terminal state $\widehat{x}_t$ and observations $\widehat{y}_1, \ldots, \widehat{y}_n$, the cross-entropy loss is

$$\ell(\widehat{x}_t, \widehat{y}_1, \ldots, \widehat{y}_n; \phi) = -\log \widetilde{Q}_\phi(y_1 {=} \widehat{y}_1, \ldots, y_n {=} \widehat{y}_n | x_t {=} \widehat{x}_t). \tag{5.34}$$

Tuples $(\widehat{x}_t, \widehat{y}_1, \ldots, \widehat{y}_n)$ are obtained as follows: 1) $\widehat{y}_1 \sim \widetilde{p}(y_1)$; 2) A trajectory $\tau = \{x_t\}_{t=0}^T$ is sampled from the reverse SDE of diffusion model y_1. At this point, we would ideally sample $\widehat{y}_2, \ldots, \widehat{y}_n$ given $\widehat{x}_0$ but this requires access to $\widetilde{p}(y_k {=} \widehat{y}_k | \widehat{x}_0)$. Instead, we approximate this with $w_i(\widehat{x}; \phi) = \widetilde{Q}_\phi(y_1 {=} i | x_0 {=} \widehat{x}_0)$ and marginalize over $\widehat{y}_2, \ldots, \widehat{y}_n$ to obtain the cross-entropy loss

$$\mathcal{L}_N(\phi, \overline{\phi}) = \mathop{\mathbb{E}}_{\substack{(\widehat{\tau}, \widehat{y}_1) \\ \sim \\ \widetilde{p}(\tau, y_1)}} \left[\sum_{\widehat{x}_t \in \widehat{\tau} \setminus \{\widehat{x}_0\}} \sum_{\widehat{y}_2=1}^{m} \cdots \sum_{\widehat{y}_n=1}^{m} \left(\prod_{k=2}^{n} w_{\widehat{y}_k}(\widehat{x}_0; \overline{\phi}) \right) \ell(\widehat{x}_t, \widehat{y}_1, \ldots, \widehat{y}_n; \phi) \right]. \tag{5.35}$$

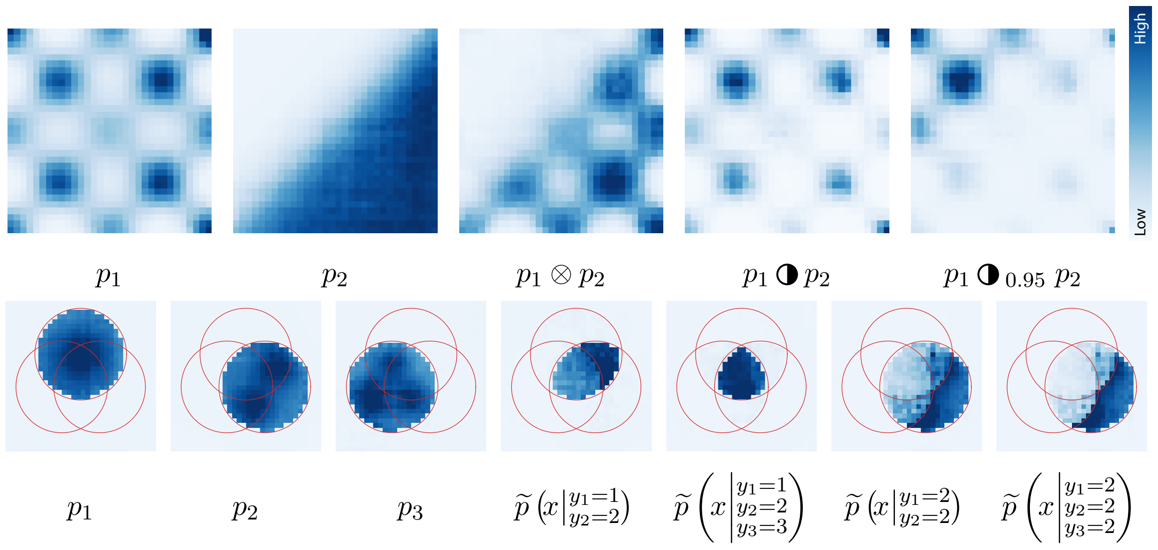


Figure 5-3: **Composed GFlowNets on** 32×32 **grid domain.** Color indicates cell probability, darker is higher. (Top) operations on two distributions. (Bottom) operations on three distributions. The red circles indicate the high probability regions of p_1, p_2, p_3.

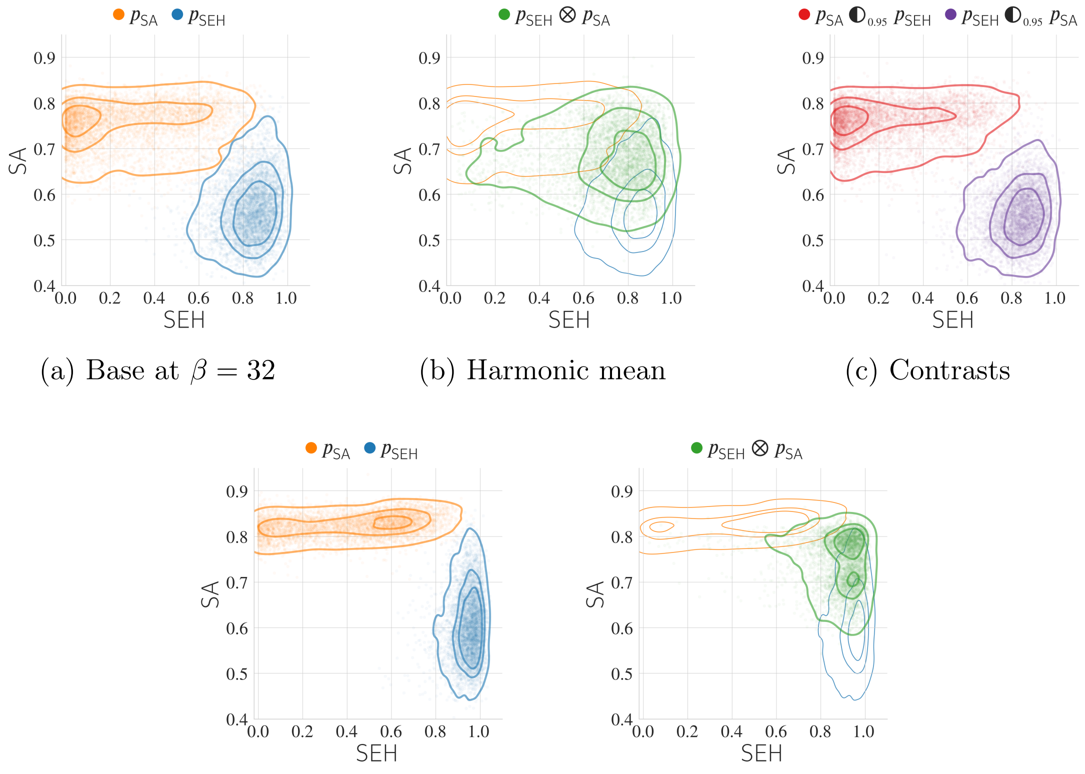
5.6 Experiments

5.6.1 2D Distributions via GFlowNet

We validate GFlowNet compositions obtained with our framework on 2D grid domain [16]. The goal of this experiment is to validate our approach in a controlled setting, where the ground-truth composite distributions can be evaluated directly.

In the 2D grid domain, the states are the cells of an $H \times H$ grid. The starting state is the upper-left cell $s_0 = (0, 0)$. At each state, the allowed actions are: 1) move right; 2) move down; 3) a stop action that indicates termination of the trajectory at the current position. For this experiment, we first trained GFlowNets $p_i(x) \propto R_i(x)$ with reward functions $R_i(x) > 0$ defined on the grid, and then trained classifiers and constructed GFlowNet compositions following 5.5.3.

Figure 5-3 (top row) shows the distributions obtained by composing two pre-trained GFlowNets (top row; left). The harmonic mean $p_1 \otimes p_2$, covers the regions that have high probability under both p_1 and p_2 and excludes locations where either of the distributions is low. p_1 ◑ p_2 resembles p_1 but the relative masses of the modes of p_1 are modulated by p_2: regions with high p_2 have lower probability under contrast. The



(a) Base at $\beta = 32$ (b) Harmonic mean (c) Contrasts

(d) Base at $\beta = 96$ (e) Harmonic mean

Figure 5-4: **Reward distributions in the molecular generation domain.** (a) Base GFlowNets at $\beta\!=\!32$: p_{SEH} and p_{SA} are trained with $R_{\mathrm{SEH}}(x)^{32}$ and $R_{\mathrm{SA}}(x)^{32}$. (b) harmonic mean of p_{SEH} and p_{SA}, (c) contrasts. (d) base GFlowNets at $\beta\!=\!96$. (e) harmonic mean. The contours indicate the level sets of the kernel density estimates in the $(R_{\mathrm{SEH}}, R_{\mathrm{SA}})$ plane.

parameterized contrast $p_1 \LEFTcircle_{0.95}\, p_2$ with $\alpha = 0.05$ magnifies the contrasting effect: high $p_2(x)$ implies very low $(p_1 \LEFTcircle_{0.95}\, p_2)(x)$.

The bottom row of Figure 5-3 shows the operations on 3 distributions. The conditional $\widetilde{p}(x|y_1\!=\!1, y_2\!=\!2)$ is concentrated on the points that have high likelihood under both p_1 and p_2. Similarly, the value $\widetilde{p}(x|y_1 = 1, y_2 = 2, y_3 = 3)$ is high if x is likely to be observed under all three distributions at the same time. The conditionals $\widetilde{p}(x|y_1\!=\!2, y_2\!=\!2)$ and $\widetilde{p}(x|y_1\!=\!2, y_2\!=\!2, y_3\!=\!2)$ highlight the points with high $p_2(x)$ but low $p_1(x)$ and $p_3(x)$. Conditioning on three labels results in a sharper distribution compared to double-conditioning. Note that the operations can be thought of as generalized set-theoretic operations (set intersection and set difference). We provide quantitative results and further details in Appendix C.5.1. The classifier learning curves are provided in Appendix 5.6.4.

5.6.2 Molecule Generation via GFlowNet

Next, we evaluate our method for GFlowNet composition on a large and highly structured data space, and asses the effect that composition operations have on resulting data distributions in a practical setting. To that end, we conducted experiments with GFlowNets trained for the molecular generation task proposed by Bengio et al. [16].

Domain. In the molecule generation task, the objects $x \in \mathcal{X}$ are molecular graphs. The non-terminal states $s \in \mathcal{S} \setminus \mathcal{X}$ are incomplete molecular graphs. The transitions from a given non-terminal state s are of two types: 1) fragment addition $s \to s'$: new molecular graph s' is obtained by attaching a new fragment to the molecular graph s; 2) stop action $s \to x$: if $s \neq s_0$, then the generation process can be terminated at the molecular graph corresponding to the current state (note that new terminal state $x \in \mathcal{X}$ is different from $s \in \mathcal{S} \setminus \mathcal{X}$, but both states correspond to the same molecular graph).

Rewards. We trained GFlowNets using 3 reward functions: **SEH**, a reward computed by an MPNN [76] that was trained by Bengio et al. [16] to estimate the binding energy of a molecule to the soluble epoxide hydrolase protein; **SA**, an estimate of synthetic accessibility [66] computed with tools from `RDKit` library [136]; **QED**, a quantitative estimate of drug-likeness [22] which is also computed with `RDKit`. We normalized all reward functions to the range $[0, 1]$. Higher values of SEH, SA, and QED correspond to stronger binding, higher synthetic accessibility, and higher drug-likeness respectively. Following Bengio et al. [16], we introduced the parameter β which controls the sharpness (temperature) of the target distribution: $p(x) \propto R(x)^{\beta}$, increasing β results in a distribution skewed towards high-reward objects. We experimented with two β values, 32 and 96 (Figure 5-4(a),5-4(d)).

Training and evaluation. After training the base GFlowNets with the reward functions described above, we trained classifiers with Algorithm 2. The classifier was parameterized as a graph neural network based on a graph transformer architecture [268]. Further details of the classifier parameterization and training are provided in Appendix C.5.2. Compared to the 2D grid domain (Section 5.6.1), we can not directly evaluate the distributions obtained by our approach. Instead, we analyzed the samples generated by the composed distributions. We sampled 5 000 molecules from each composed distribution obtained with our approach as well as the base GFlowNets. We evaluated the sample collections with the two following strategies. **Reward evaluation**: we analyzed the distributions of rewards across the sample collections. The goal is to see whether the composition of GFlowNets trained for different rewards leads to noticeable changes in reward distribution. **Distribution distance evaluation**: we used the samples to estimate the pairwise distances between the distributions. Specifically, for a given pair of distributions represented by two collections of samples $\mathcal{D}_A = \{x_{A,i}\}_{i=1}^{n}$, $\mathcal{D}_B = \{x_{B,i}\}_{i=1}^{n}$ we computed the earth mover's distance $d(\mathcal{D}_A, \mathcal{D}_B)$ with ground molecule distance given by $d(x, x') = (\max\{s(x, x'), 10^{-3}\})^{-1} - 1$, where $s(x, x') \in [0, 1]$ is the Tanimoto similarity over Morgan fingerprints of molecules x and x'.

Table 5.1: Reward distributions of composite GFlowNets.

SEH	low				high			
SA	low		high		low		high	
QED	low	high	low	high	low	high	low	high
p_{SEH}	0	0	0	0	**62**	**9**	**24**	**5**
p_{SA}	0	0	**73**	**4**	0	0	**18**	**5**
p_{QED}	0	**40**	0	**26**	0	**21**	0	**13**
(a) $y = \{\text{SEH}, \text{SA}\}$	1	0	16	2	6	3	**54**	**18**
(b) $y = \{\text{SEH}, \text{QED}\}$	0	11	0	4	1	**48**	4	**32**
(c) $y = \{\text{SA}, \text{QED}\}$	0	15	1	**42**	0	8	2	**32**
(d) $y = \{\text{SEH}, \text{SA}, \text{QED}\}$	0	7	2	11	2	19	10	**49**
(e) $y = \{\text{SEH}, \text{SEH}, \text{SEH}\}$	0	0	0	0	**63**	9	24	4
(f) $y = \{\text{SA}, \text{SA}, \text{SA}\}$	0	0	**74**	5	0	0	17	4
(g) $y = \{\text{QED}, \text{QED}, \text{QED}\}$	0	**40**	0	23	0	23	0	14

In each row, the numbers show the percentage of the samples from the respective model that fall into one of 8 bins according to rewards. The "low" and "high" categories are decided by thresholding SEH: 0.5, SA: 0.6, QED: 0.25.

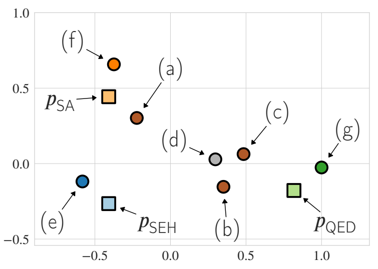


Figure 5-5: 2D t-SNE embeddings of three base GFlowNets trained with $R_{\text{SEH}}(x)^{\beta}$, $R_{\text{SA}}(x)^{\beta}$, $R_{\text{QED}}(x)^{\beta}$, at $\beta = 32$ and their compositions. The t-SNE embeddings are computed based on pairwise earth mover's distances between the distributions. Labels (a)-(g) match rows in Table 5.1.

Results. Figure 5-4 shows reward distributions of base GFlowNets (trained with SEH and SA rewards at $\beta \in \{32, 96\}$) and their compositions. Base GFlowNet distributions are concentrated on examples that score high in their respective rewards. For each model, there is considerable variation in the reward that was not used for training. The harmonic mean operation (Figures 5-4(b), 5-4(e)) results in distributions that are concentrated on the samples scoring high in both rewards. The contrast operation (Figure 5-4(c)) has the opposite effect: the distributions are skewed towards the examples scoring high in only one of the original rewards. Note that the tails of the contrast distributions are retreating from the area covered by the harmonic mean.

We show reward distribution statistics of three GFlowNets (trained with SEH, SA, and QED at $\beta = 32$) and their compositions in Table 5.1. Each row of the table gives a breakdown (percentages) of the samples from a given model into one of $2^3 = 8$ bins according to rewards. For all three base models, the majority of the samples fall into the "high" category according to the respective reward, while the rewards that were not used for training show variation. Conditioning on two different labels (e.g. $y = \{\text{SEH}, \text{QED}\}$) results in concentration on examples that score high in two selected rewards, but not necessarily scoring high in the reward that was not selected. The conditional $y = \{\mathrm{SEH}, \mathrm{QED}, \mathrm{SA}\}$ shifts the focus to examples that have all three properties.

Figure 5-5 shows 2D embeddings of the distributions appearing in Table 5.1. The embeddings were computed with t-SNE based on the pairwise earth mover's distances. The configuration of the embeddings gives insight into the configuration of base models and conditionals in the distribution space. We see that points corresponding to pairwise conditionals lie in between the two base models selected for conditioning. Conditional $y = \{\mathrm{SEH}, \mathrm{SA}, \mathrm{QED}\}$ appears to be near the centroid of the triangle $(p_{\mathrm{SEH}}, p_{\mathrm{SA}}, p_{\mathrm{QED}})$ and lies close the the pairwise conditionals. The distributions obtained by repeated conditioning on the same label value (e.g. $y = \{\mathrm{SEH}, \mathrm{SEH}, \mathrm{SEH}\}$) are spread out to the boundary and lie closer to the respective base distributions while being relatively far from pairwise conditionals. We provide a complete summary of the distribution distances in Table 5.2. The sample diversity statistics of base GFlowNets at different values of β are provided in Appendix C.6.1.

Table 5.2: Estimated pairwise earthmover's distances between distributions shown in Table 5.1.

	SEH	SA	QED	SEH SA	SEH QED	SA QED	SEH SA QED	SEH × 3	SA × 3	QED × 3
SEH	0	4.42	5.77	3.39	4.20	4.88	4.10	2.46	4.44	5.73
SA	4.42	0	5.88	3.26	5.15	4.59	4.20	4.39	2.55	5.89
QED	5.77	5.88	0	5.40	4.02	3.85	4.20	5.80	5.90	3.20
SEH,SA	3.39	3.26	5.40	0	4.25	4.19	3.68	3.39	3.30	5.39
SEH,QED	4.20	5.15	4.02	4.25	0	3.80	3.67	4.22	5.19	4.00
SA,QED	4.88	4.59	3.85	4.19	3.80	0	3.65	4.91	4.59	3.87
SEH,SA,QED	4.10	4.20	4.20	3.68	3.67	3.65	0	4.12	4.23	4.20
SEH × 3	2.46	4.39	5.80	3.39	4.22	4.91	4.12	0	4.43	5.73
SA × 3	4.44	2.55	5.90	3.30	5.19	4.59	4.23	4.43	0	5.90
QED × 3	5.73	5.89	3.20	5.39	4.00	3.87	4.20	5.73	5.90	0

5.6.3 Colored MNIST Generation via Diffusion Models

Finally, we empirically test our method for the composition of diffusion models on image generation task.

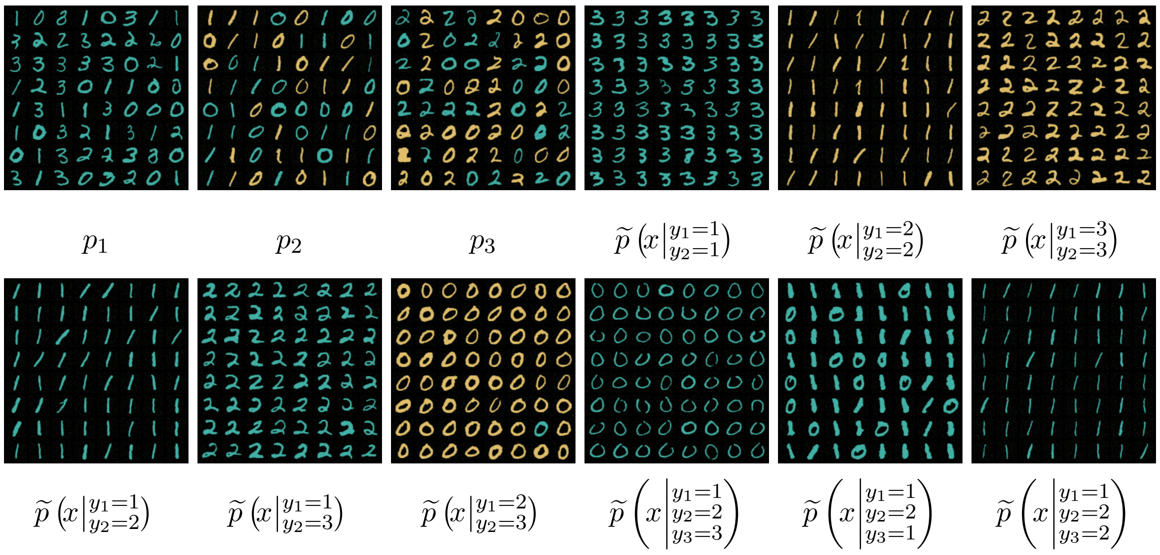

p_1
p_2
p_3
$\widetilde{p}\left(x\middle|\begin{smallmatrix}y_1=1\\y_2=1\end{smallmatrix}\right)$
$\widetilde{p}\left(x\middle|\begin{smallmatrix}y_1=2\\y_2=2\end{smallmatrix}\right)$
$\widetilde{p}\left(x\middle|\begin{smallmatrix}y_1=3\\y_2=3\end{smallmatrix}\right)$
$\widetilde{p}\left(x\middle|\begin{smallmatrix}y_1=1\\y_2=2\end{smallmatrix}\right)$
$\widetilde{p}\left(x\middle|\begin{smallmatrix}y_1=1\\y_2=3\end{smallmatrix}\right)$
$\widetilde{p}\left(x\middle|\begin{smallmatrix}y_1=2\\y_2=3\end{smallmatrix}\right)$
$\widetilde{p}\left(x\middle|\begin{smallmatrix}y_1=1\\y_2=2\\y_3=3\end{smallmatrix}\right)$
$\widetilde{p}\left(x\middle|\begin{smallmatrix}y_1=1\\y_2=2\\y_3=1\end{smallmatrix}\right)$
$\widetilde{p}\left(x\middle|\begin{smallmatrix}y_1=1\\y_2=2\\y_3=2\end{smallmatrix}\right)$

Figure 5-6: **Composed diffusion models on colored MNIST.** Samples from 3 pre-trained diffusion models and their various compositions.

In this experiment, we composed three diffusion models that are pre-trained to generate colored MNIST digits [138]. Model p_1 was trained to generate cyan digits less than 4, p_2 to generate cyan and beige digits less than 2, and p_3 to generate cyan and beige even digits less than 4. In essence, each model was trained to generate digits with a specific property: p_1 generates cyan digits, p_2 generates digits less than 2, and p_3 generates even digits.

We built the composition iteratively by factorizing the posterior as $\widetilde{p}(x|y_1, y_2, y_3) \propto \widetilde{p}(x)\widetilde{p}(y_1, y_2|x)\widetilde{p}(y_3|x, y_1, y_2)$. To this end, we first trained a classifier $\widetilde{Q}(y_1, y_2|x_t)$ on trajectories sampled from the base models. This allows us to generate samples from $\widetilde{p}(x|y_1, y_2)$. We then trained an additional classifier $\widetilde{Q}(y_3|x_t, y_1, y_2)$ on trajectories from compositions defined by (y_1, y_2) to allow us to sample from $\widetilde{p}(x|y_1, y_2, y_3)$. Additional details can be found in Appendix C.5.3.

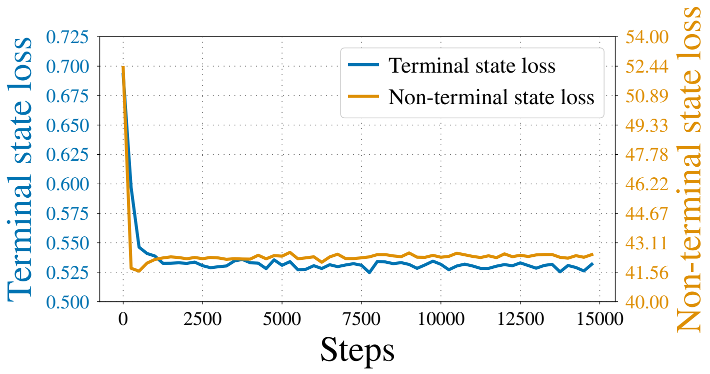
Figure 5-6 shows samples from the pre-trained models and from selected compositions. The negating effect of *not* conditioning on observations is clearly visible in the compositions using two variables. For example, $\widetilde{p}(x|y_1\!=\!1, y_2\!=\!1)$ only generates cyan 3 digits. Because there we *do not* condition on p_2 or p_3, the composition excludes digits that have high probability under p_2 or p_3, i.e. those that are less than 2 or even. We can make a similar analysis of $\widetilde{p}(x|y_1\!=\!1, y_2\!=\!3)$. Cyan even digits have high density under both p_1 and p_3, but because p_2 is not conditioned on, the composition excludes digits less than two (i.e. cyan 0's). Finally, $\widetilde{p}(x|y_1\!=\!1, y_2\!=\!2, y_3\!=\!3)$ generates only cyan 0 digits, on which all base models have high density.

5.6.4 Classifier Learning Curves and Training Time

We empirically evaluated classifier training time and learning curves. The results are shown in Figures 5-7, 5-8, 5-9 and Tables 5.3, 5.4.

Figures 5-7, 5-8, 5-9 show the cross-entropy loss of the classifier for terminal (5.25) and non-terminal states (5.27) as a function of the number of training steps for the GFlowNet 2D grid domain, the molecular generation domain, and the Colored MNIST digits domain respectively. They show that the loss drops quickly but remains above 0. Figure 5-7 further shows the distances between the learned compositions and the ground truth distributions as a function of the number of training steps of the classifier. For all compositions, as the classifier training progresses, the distance to the ground truth distribution decreases. Compared to the distance at initialization we observe almost an order of magnitude distance reduction by the end of the training.

The runtime of classifier training is shown in Tables 5.3 and 5.4. We report the total runtime, as well as separate measurements for the time spent sampling trajectories and training the classifier. The classifier training time is comparable to the base generative model training time. However, most of the classifier training time (more than 70%, or even 90%) was spent on sampling trajectories from base generative models. Our implementation of the training could be improved in this regard, e.g. by sampling a smaller set of trajectories once and re-using trajectories for training and by reducing the number of training steps (the loss curves in Figures 5-7, 5-8, 5-9 show that classification losses plateau quickly).



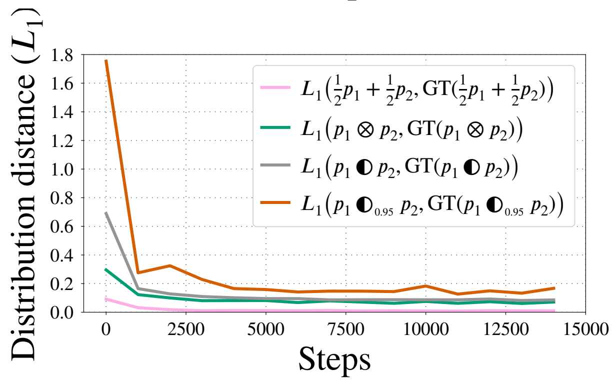


Figure 5-7: Training curves of the classifier $\widetilde{Q}_\phi(y_1, y_2|\cdot)$ in GFlowNet 2D grid domain. The experimental setup corresponds to Section 5.6.1 and Figure 5-3 (top row). **Top**: Terminal state loss and non-terminal state loss (as defined in Algorithm 2) as functions of the number of training steps. **Bottom**: L_1 distance between learned distributions (compositions obtained through classifier-based mixture and guidance) and ground-truth composition distributions as the function of the number of training steps. $L_1(p, q) = \sum_{x \in \mathcal{X}} |p(x) - q(x)|$.

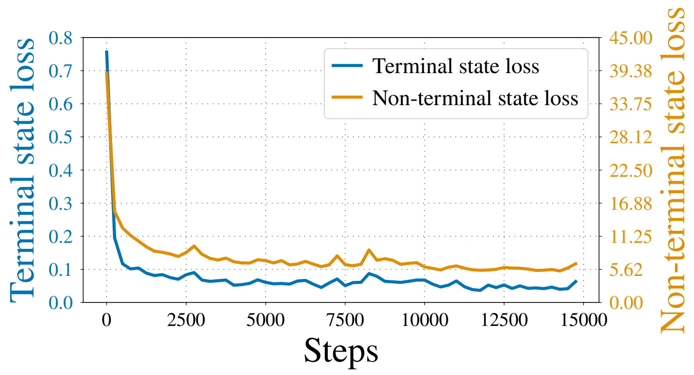


Figure 5-8: Training curves of the classifier $\widetilde{Q}_\phi(y_1, y_2|\cdot)$ in GFlowNet molecule generation domain. The experimental setup corresponds to Section 5.6.2 and Figure 5-4 (a-c). The curves show terminal state loss and non-terminal state loss (as defined in Algorithm 2) as functions of the number of training steps.

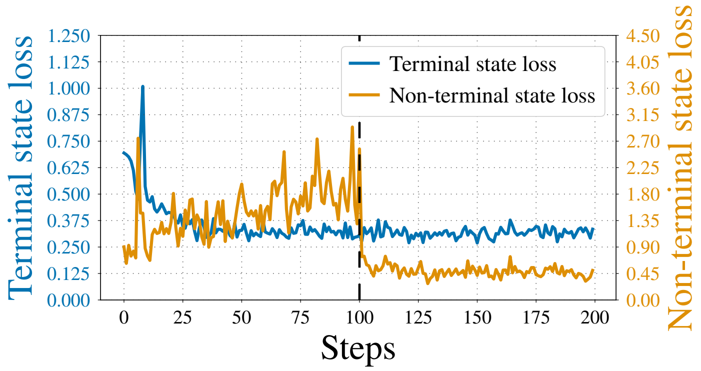


Figure 5-9: Training curves of the classifier $\widetilde{Q}_\phi(y_1, y_2|\cdot)$ in diffusion MNIST image generation domain. The experimental setup corresponds to Section C.6.2 and Figure C-1. The curves show terminal state loss and non-terminal state loss (as defined in equations (5.33), (5.35)) as functions of the number of training steps. The non-terminal loss optimization begins after the first 100 training steps (shown by the black dashed line).

Table 5.3: Summary of base GFlowNet and classifier training time in molecule generation domain. The experimental setup corresponds to Section 5.6.2 & Figure 5-4 (a-c). All models were trained with a single GeForce RTX 2080 Ti GPU.

Base GFlowNet training steps	20 000
Base GFlowNet batch size	64
Base GFlowNet training elapsed real time	6h 47m 11s
Classifier training steps	15 000
Classifier batch size	8 trajectories per base model (all states used)
Classifier training total elapsed real time	9h 2m 19s
Classifier training data generation time	6h 35m 58s (73%)

Table 5.4: Summary of base diffusion and classifier training time in MNIST image generation domain. The experimental setup corresponds to Section C.6.2 & Figure C-1 in the main paper. All models were trained with a single Tesla V100 GPU.

Base diffusion training steps	200
Base diffusion batch size	32
Base diffusion training elapsed real time	10m 6s
Classifier training steps	200
Classifier batch size	128 trajectories per base model (35 time-steps per trajectory)
Classifier training total elapsed real time	30m 12s
Classifier training data generation time	29m 22s (97%)

Chapter 6

Discussion

Deep probabilistic models are a core component of modern artificial intelligence systems. This thesis contributes to the methodology of training and inference in deep probabilistic models using auxiliary models trained for separate tasks. The research presented in this thesis focused on principled methods with guarantees on the optimality of target distribution configurations and sampling from target distributions.

Pairwise Discriminators for Adversarial Training. In Chapter 3, we introduced PairGAN, a formulation of adversarial training where the training dynamics does not suffer from the instability of the alignment. Our theoretical results constitute first steps in understanding convergence guarantees for PairGAN. Interestingly, in our setup, one can formalize the balance of power between the discriminator and the generator with the notion of sufficient discriminators, which is not present in the standard formulation of GANs. Throughout our analysis, PairGAN enjoys flexibility which permits the use of different loss functions and model architectures.

Future Work. Directions for future work include further theoretical understanding of convergence guarantees and properties of sufficient discriminators. More extensive experiments with different design choices are necessary to understand the general improvements that PairGAN can bring. While the PairGAN formulation addresses a particular issue contributing to the instability of adversarial training, the overall problem of training instability is not fully resolved. In particular the stability of the distribution alignment is only relevant when the generator is close to the target distribution. In practice the entire learning trajectory of the generator might lie far from the target demonstrating an unstable learning behavior not addressed by PairGAN. Further research is required for detailed characterization and mitigation of other instability issues arising in practice.

Adversarial Support Alignment. In Chapter 4, we studied the problem of aligning the supports of distributions. We formalized the theoretical connections between support alignment and existing notions of alignment. We proposed a practical method that uses a signal from a Jensen-Shannon discriminator to guide distributions towards support alignment. We demonstrated the effectiveness of the approach in domain adaptation under label distribution shift.

Future Work. We believe that our methodology opens possibilities for the design of more nuanced and structured alignment constraints, suitable for various use cases. One natural extension is support containment, achievable with only one term in (4.1). This approach is fitting for partial domain adaptation, where some source domain classes do not appear in the target domain. Another interesting direction is unsupervised domain transfer, where support alignment is more desired than existing distribution alignment methods due to mode imbalance [23]. Furthermore, support alignment can be used as the objective for generative modeling. Potential use cases includes training a generative model by aligning the model support with the data support, thus ensuring that the model can generate samples similar to data examples without reproducing frequency biases (i.e. relative probabilities of various regions) present in the training data. Related to this idea is the problem of diverse sampling (see e.g. [49]), which can be formalized as uniformly covering the volume of the fixed support.

Support alignment lifts the restrictive constraints of strict distribution alignment. Naturally, this relaxation expands the set of valid solutions, as there are many ways to modify distribution densities while keeping the supports aligned, Optimizing for support alignment in isolation is an underspecifed problem. The algorithm based on support alignment can choose one of the valid support alignment solutions. Clearly, some of the support alignment solutions are less desirable than others. For instance, in the case of undsupervised domain adaptation, support alinged solutions include the configuration where the marginal supports of source and target embeddings are aligned, but individual class supports of target distribution are matched with incorrect class supports of source distribution. To design algorithms that avoid such pathological solutions, one needs to introduce additional explicit regularization or implicit inductive biases in the model and learning algorithm. We leave the research on the exploration of such regularization strategies for future work.

Compositional Sculpting of Iterative Generative Processes. In Chapter 5, we introduced Compositional Sculpting, a general approach for composing iterative generative models. Compositions are defined through "observations", which enable us to emphasize or de-emphasize the density of the composition in regions where specific base models have high density. We highlighted two binary compositions, harmonic mean and contrast, which are analogous to the product and negation operations defined on EBMs. A crucial feature of the compositions we have introduced is that thet can be realized via guided diffusion or GFlowNet generative policies. By extending classifier guidance we are able to leverage the generative capabilities of the base models to produce samples from the composition. Through empirical experiments, we validated our approach for composing diffusion models and GFlowNets on toy domains, molecular generation, and image generation.

Future Work. Our work focused on principled approaches with guarantees on sampling from distributions defined through algebraic composition operations. On the methodological front, the exploration of composition and coordination mechanisms built on alternative principles is an exciting avenue for future research. We suspect that the design of novel supervision or model selection signals for learning coordination mechanisms would play a vital role in this research direction. Another path to improved coordination techniques lies in the development of new model families that can be efficiently composed and adapted for new tasks. We argue that the landscape of the models is not extensively explored as the tradeoffs between model representation power and composition ability are poorly understood. For example, energy models are relatively harder to train but easy to compose. In contrast, diffusion models enable stable training but can not be composed by simple manipulations on learned score functions without losing guarantees.

On the application front, developing methods for collaboration between multiple models or human-AI collaboration is a promising direction. In engineering, sciences, and arts, complex designs involve the use of several different types of expertise governed by different interacting components. For example, aircraft design involves the collaboration of various specialists: structural engineers, material engineers, aerodynamicists, avionics engineers, safety engineers, and others, each primarily focused on respective specific aspects of design. Naturally, arriving at the final design requires an extensive exploration of complex trade-offs between constraints associated with different expertise domains. Generative models only recently reached capabilities enabling the generation of complex objects in domains such as

- generation of coherent music from different instruments [162];
- multi-objective engineering design [75, 164];
- molecule generation [16, 48, 112];
- generation of biomolecules [29, 36, 267] and large geometric arrangements of molecules [257];
- generative multi-task control [4, 41, 109];
- multi-agent motion prediction and control [56, 110].

We believe that research on cooperative generative design will pave the way for a new generation of AI-assisted design systems that enhance creativity and efficiency.

Bibliography

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and others (OpenAI team). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Jayant Joshi, Ryan C. Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego M Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR, 2023.

[3] Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

[4] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision

making? In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=sP1fo2K9DFG`.

[5] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions, 2023. URL `https://arxiv.org/abs/2303.08797`.

[6] Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://arxiv.org/abs/2209.15571`.

[7] Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[8] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*, 2017.

[9] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[10] Yogesh Balaji, Rama Chellappa, and Soheil Feizi. Robust optimal transport with applications in generative modeling and domain adaptation. *Advances in Neural Information Processing Systems Foundation (NeurIPS)*, 2020.

[11] Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. Multidiffusion: Fusing diffusion paths for controlled image generation. In *International Conference on Machine Learning*, pages 1737–1752. PMLR, 2023.

[12] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007.

[13] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[14] Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-flow: Differentiating through flows for controlled generation. *arXiv preprint arXiv:2402.14017*, 2024.

[15] BIG bench authors. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL `https://openreview.net/forum?id=uyTL5Bvosj`.

[16] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

[17] Yoshua Bengio and Samy Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. *Advances in Neural Information Processing Systems*, 12, 1999.

[18] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

[19] Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J. Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023. URL `http://jmlr.org/papers/v24/22-0364.html`.

[20] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[21] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2:3, 2023.

[22] G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

[23] Mikolaj Binkowski, Devon Hjelm, and Aaron Courville. Batch weight for domain adaptation with mass shift. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1844–1853, 2019.

[24] Christopher M Bishop. *Pattern recognition and machine learning.* Springer New York, NY, 2006.

[25] Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=r1lUOzWCW`.

[26] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=YCWjhGrJFD`.

[27] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric

Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, J. F. Nyarko, Giray Ogut, Laurel J. Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan, Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *ArXiv*, abs/2108.07258, 2021.

[28] Nicolas Bonneel and David Coeurjolly. Spot: sliced partial optimal transport. *ACM Transactions on Graphics (TOG)*, 38(4):1–13, 2019.

[29] Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian FATRAS, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael M. Bronstein, and Alexander Tong. SE(3)-stochastic flow matching for protein backbone generation. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=kJFIH23hXb`.

[30] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

[31] Damiano Brigo. The general mixture-diffusion sde and its relationship with an uncertain-volatility option model with volatility-asset decorrelation. *ArXiv*, abs/0812.4052, 2008.

[32] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally

Jesmonth, Nikhil J. Joshi, Ryan C. Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Anand Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Ho Vuong, F. Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. *ArXiv*, abs/2212.06817, 2022.

[33] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[34] Eric Budish, Yeon-Koo Che, Fuhito Kojima, and Paul Milgrom. Implementing random assignments: A generalization of the birkhoff-von neumann theorem. In *2009 Cowles Summer Conference*, 2009.

[35] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[36] Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. *arXiv preprint arXiv:2402.04997*, 2024.

[37] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Robust, deep and inductive anomaly detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 36–51. Springer, 2017.

[38] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[39] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. In *International conference on machine learning*, pages 864–872. PMLR, 2018.

[40] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2818–2829, 2023.

[41] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[42] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. Ilvr: Conditioning method for denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14367–14376, October 2021.

[43] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311, 2022.

[44] Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. *Advances in Neural Information Processing Systems*, 35:25683–25696, 2022.

[45] Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=1vmSEVL19f`.

[46] 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.

[47] Colin Conwell and Tomer Ullman. Testing relational understanding in text-guided image generation. *ArXiv*, abs/2208.00005, 2022.

[48] Gabriele Corso, Bowen Jing, Regina Barzilay, Tommi Jaakkola, et al. Diffdock: Diffusion steps, twists, and turns for molecular docking. In *International Conference on Learning Representations (ICLR 2023)*, 2023.

[49] Gabriele Corso, Yilun Xu, Valentin De Bortoli, Regina Barzilay, and Tommi S. Jaakkola. Particle guidance: non-i.i.d. diverse sampling with diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=KqbCvIFBY7`.

[50] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=3lge0p5o-M-`.

[51] Lucas Deecke, Robert Vandermeulen, Lukas Ruff, Stephan Mandt, and Marius Kloft. Image anomaly detection with generative adversarial networks. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 3–17. Springer, 2018.

[52] Ishan Deshpande, Ziyu Zhang, and Alexander G Schwing. Generative modeling using the sliced wasserstein distance. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3483–3491, 2018.

[53] Ishan Deshpande, Yuan-Ting Hu, Ruoyu Sun, Ayis Pyrros, Nasir Siddiqui, Sanmi Koyejo, Zhizhen Zhao, David Forsyth, and Alexander G Schwing. Max-sliced wasserstein distance and its use for gans. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, pages 10648–10656, 2019.

[54] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2018.

[55] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[56] Christopher Diehl, Tobias Klosek, Martin Krueger, Nils Murzyn, Timo Osterburg, and Torsten Bertram. Energy-based potential games for joint motion forecasting and control. In *7th Annual Conference on Robot Learning*, 2023. URL `https://openreview.net/forum?id=Eyb4e3GBuuR`.

[57] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[58] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[59] Danny Driess, F. Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Ho Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Peter R. Florence. Palm-e: An embodied multimodal language model. *ArXiv*, abs/2303.03378, 2023.

[60] Yilun Du and Leslie Pack Kaelbling. Position: Compositional generative modeling: A single model is not all you need. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=SoNexFx8qz`.

[61] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy-based models. *Advances in Neural Information Processing Systems*, 32, 2019.

[62] Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. *Advances in Neural Information Processing Systems*, 33: 6637–6647, 2020.

[63] Yilun Du, Shuang Li, Yash Sharma, Josh Tenenbaum, and Igor Mordatch. Unsupervised learning of compositional energy concepts. *Advances in Neural Information Processing Systems*, 34:15608–15620, 2021.

[64] Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International Conference on Machine Learning*, pages 8489–8510. PMLR, 2023.

[65] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[66] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1:1–11, 2009.

[67] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[68] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024.

[69] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network

for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[70] Farzan Farnia and Asuman Ozdaglar. Do gans always have nash equilibria? In *International Conference on Machine Learning*, pages 3029–3039. PMLR, 2020.

[71] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit Haim Bermano, Gal Chechik, and Daniel Cohen-or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=NAQvF08TcyG`.

[72] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[73] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[74] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617. PMLR, 2018.

[75] Giorgio Giannone, Akash Srivastava, Ole Winther, and Faez Ahmed. Aligning optimization trajectories with diffusion models for constrained design generation. *Advances in Neural Information Processing Systems*, 36, 2024.

[76] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[77] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[78] 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.

[79] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJxgknCcK7`.

[80] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[81] Karol Gregor and Yann LeCun. Learning representations by maximizing compression. *arXiv preprint arXiv:1108.1169*, 2011.

[82] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In *International Conference on Machine Learning*, pages 1242–1250. PMLR, 2014.

[83] Ulf Grenander and Michael I Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56 (4):549–581, 1994.

[84] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520, 2007.

[85] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[86] Aditya Grover and Stefano Ermon. Boosted generative models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[87] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5769–5779, 2017.

[88] William Harvey, Saeid Naderiparizi, and Frank Wood. Conditional image generation by conditioning variational auto-encoders. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=7MV6uLzOChW`.

[89] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[90] Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[91] 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 2016.

[92] 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, 2016.

[93] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-or. Prompt-to-prompt image editing with cross-attention control. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=_CDixzkzeyb`.

[94] 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.

[95] Geoffrey Hinton, Simon Osindero, Max Welling, and Yee-Whye Teh. Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive science*, 30(4):725–731, 2006.

[96] Geoffrey E Hinton. Products of experts. In *Ninth International Conference on Artificial Neural Networks*, volume 1, 1999.

[97] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[98] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL `https://openreview.net/forum?id=qw8AKxfYbI`.

[99] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[100] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

[101] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. Mgan: Training generative adversarial nets with multiple generators. In *International conference on learning representations*, 2018.

[102] Heiko Hoffmann. Kernel pca for novelty detection. *Pattern recognition*, 40(3): 863–874, 2007.

[103] Ziqi Huang, Kelvin C.K. Chan, Yuming Jiang, and Ziwei Liu. Collaborative diffusion for multi-modal face generation and editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6080–6090, June 2023.

[104] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.

[105] Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020.

[106] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[107] Oleg Ivanov, Michael Figurnov, and Dmitry Vetrov. Variational autoencoder with arbitrary conditioning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=SyxtJh0qYm`.

[108] Moksh Jain, Sharath Chandra Raparthy, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-objective GFlownets. In *International Conference on Machine Learning*, pages 14631–14653. PMLR, 2023.

[109] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pages 9902–9915. PMLR, 2022.

[110] Chiyu Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, Dragomir Anguelov, et al. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9644–9653, 2023.

[111] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Multi-objective molecule generation using interpretable substructures. In *International conference on machine learning*, pages 4849–4859. PMLR, 2020.

[112] Bowen Jing, Gabriele Corso, Jeffrey Chang, Regina Barzilay, and Tommi Jaakkola. Torsional diffusion for molecular conformer generation. *Advances in Neural Information Processing Systems*, 35:24240–24253, 2022.

[113] Fredrik D Johansson, David Sontag, and Rajesh Ranganath. Support and invertibility in domain-invariant representations. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 527–536. PMLR, 2019.

[114] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard GAN. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1erHoR5t7.

[115] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[116] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[117] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=k7FuTOWMOc7.

[118] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *Advances in Neural Information Processing Systems*, 35:23593–23606, 2022.

[119] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6007–6017, 2023.

[120] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[121] diederik p kingma and jimmy ba. Adam: a method for stochastic optimization. In *international conference on learning representations*, 2015.

[122] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6114`.

[123] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[124] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. *ArXiv*, abs/2304.02643, 2023.

[125] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 971–980, 2017.

[126] Edwin M Knorr, Raymond T Ng, and Vladimir Tucakov. Distance-based outliers: algorithms and applications. *The VLDB Journal*, 8(3):237–253, 2000.

[127] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[128] Andrei N. Kolmogorov. *Grundbegriffe der wahrscheinlichkeitsrechnung*. Springer, Berlin, 1933. A Russian translation by G. M. Bavli, appeared under the title *Основные понятия теории вероятностей* (Nauka, Moscow) in 1936 [279], with a second edition, slightly expanded by Kolmogorov with the assistance of A. N. Shiryaev, in 1974, and a third edition (FAZIS, Moscow) in 1998. An English translation by N. Morrison appeared under the title *Foundations of the Theory of Probability* (Chelsea, New York) in 1950 [129], with a second edition in 1956.

[129] Andrey Kolmogorov. *Foundations of the theory of probability.* Chelsea Publishing Co., 1950.

[130] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[131] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[132] Abhishek Kumar, Prasanna Sattigeri, Kahini Wadhawan, Leonid Karlinsky, Rogerio Feris, Bill Freeman, and Gregory Wornell. Co-regularized alignment for unsupervised domain adaptation. *Advances in Neural Information Processing Systems*, 31:9345–9356, 2018.

[133] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR, 2015.

[134] Salem Lahlou, Tristan Deleu, Pablo Lemos, Dinghuai Zhang, Alexandra Volokhova, Alex Hernández-Garcıa, Léna Néhale Ezzine, Yoshua Bengio, and Nikolay Malkin. A theory of continuous generative flow networks. In *International Conference on Machine Learning*, pages 18269–18300. PMLR, 2023.

[135] Guanghui Lan. *First-order and stochastic optimization methods for machine learning*, volume 1. Springer, 2020.

[136] Greg Landrum. Rdkit: Open-source cheminformatics, 2010. URL `https://www.rdkit.org/`.

[137] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 29–37. JMLR Workshop and Conference Proceedings, 2011.

[138] Yann LeCun. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

[139] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

[140] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998.

[141] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. 2006.

[142] Yuseung Lee, Kunho Kim, Hyunjin Kim, and Minhyuk Sung. Syncdiffusion: Coherent montage via synchronized joint diffusions. *Advances in Neural Information Processing Systems*, 36:50648–50660, 2023.

[143] Bo Li, Yezhen Wang, Tong Che, Shanghang Zhang, Sicheng Zhao, Pengfei Xu, Wei Zhou, Yoshua Bengio, and Kurt Keutzer. Rethinking distributional matching based domain adaptation. *arXiv preprint arXiv:2006.13352*, 2020.

[144] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pages 2203–2213, 2017.

[145] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018.

[146] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 624–639, 2018.

[147] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1498–1507, 2018.

[148] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=PqvMRDCJT9t`.

[149] Nan Liu, Shuang Li, Yilun Du, Josh Tenenbaum, and Antonio Torralba. Learning to compose visual relations. *Advances in Neural Information Processing Systems*, 34:23166–23178, 2021.

[150] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*, pages 423–439. Springer, 2022.

[151] Nan Liu, Yilun Du, Shuang Li, Joshua B Tenenbaum, and Antonio Torralba. Unsupervised compositional concepts discovery with text-to-image generative models. *ArXiv*, abs/2306.05357, 2023.

[152] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=XVjTT1nw5z`.

[153] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[154] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pages 2208–2217. PMLR, 2017.

[155] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, page 1640–1650, 2018.

[156] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11461–11471, June 2022.

[157] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[158] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlownets. In *Advances in Neural Information Processing Systems*, volume 35, pages 5955–5967, 2022.

[159] Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward J Hu, Katie E Everett, Dinghuai Zhang, and Yoshua Bengio. GFlownets and variational inference. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=uKiE0VIluA-`.

[160] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[161] Dimitra Maoutsa, Sebastian Reich, and Manfred Opper. Interacting particle solutions of fokker–planck equations through gradient–log–density estimation. *Entropy*, 22(8):802, 2020.

[162] Giorgio Mariani, Irene Tallini, Emilian Postolache, Michele Mancusi, Luca Cosmo, and Emanuele Rodolà. Multi-source diffusion models for simultaneous music generation and separation. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=h922Qhkmx1`.

[163] Manel Mateos, Alejandro González, and Xavier Sevillano. Guiding gans: How to control non-conditional pre-trained gans for conditional image generation. *arXiv preprint arXiv:2101.00990*, 2021.

[164] François Mazé and Faez Ahmed. Diffusion models beat gans on topology optimization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 9108–9116, 2023.

[165] Eric Mazumdar, Lillian J Ratliff, and S Shankar Sastry. On gradient-based learning in continuous games. *SIAM Journal on Mathematics of Data Science*, 2(1):103–131, 2020.

[166] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=aBsCjcPu_tE`.

[167] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *Advances in Neural Information Processing Systems*, pages 1825–1835, 2017.

[168] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International Conference on Machine Learning*, pages 3481–3490, 2018.

[169] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[170] Takeru Miyato and Masanori Koyama. cGANs with projection discriminator. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=ByS1VpgRZ`.

[171] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[172] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[173] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62, 2020.

[174] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL `http://probml.github.io/book1`.

[175] Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023. URL `http://probml.github.io/book2`.

[176] Vaishnavh Nagarajan and J Zico Kolter. Gradient descent gan optimization is locally stable. In *Advances in Neural Information Processing Systems*, pages 5585–5595, 2017.

[177] Radford M Neal. Mcmc using hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, pages 113–162. Chapman and Hall/CRC, 2011.

[178] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[179] Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14, 2001.

[180] Trung Nguyen, Quang-Hieu Pham, Tam Le, Tung Pham, Nhat Ho, and Binh-Son Hua. Point-set distances for learning representations of 3d point clouds. *arXiv preprint arXiv:2102.04014*, 2021.

[181] 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.

[182] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. *Advances in Neural Information Processing Systems*, 32, 2019.

[183] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 271–279, 2016.

[184] OpenAI. Chatgpt (mar 14, 2023 version). Large language model, 2023. URL `https://chat.openai.com/chat`.

[185] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL `https://openreview.net/forum?id=a68SUt6zFt`.

[186] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744, 2022.

[187] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

[188] Giorgio Parisi. Correlation functions and computer simulations. *Nuclear Physics B*, 180(3):378–384, 1981.

[189] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[190] Judea Pearl. *Causality.* Cambridge University Press, 2 edition, 2009.

[191] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-adversarial domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[192] Stefano Peluchetti. Non-denoising forward-time diffusions, 2022. URL `https://openreview.net/forum?id=oVfIKuhqfC`.

[193] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge, 2017.

[194] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1406–1415, 2019.

[195] Pramuditha Perera, Ramesh Nallapati, and Bing Xiang. Ocgan: One-class novelty detection using gans with constrained latent representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2898–2906, 2019.

[196] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms.* The MIT Press, 2017.

[197] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[198] LS Pontryagin, VG Boltyanskii, RV Gamkrelidze, and EF Mishchenko. *Mathematical Theory of Optimal Processes.* CRC Press, 1986. Originally published in Russian as *Математическая теория оптимальных процессов* (Nauka, Moscow) in 1961 [280].

[199] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 435–446. Springer, 2011.

[200] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[201] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. 2018.

[202] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[203] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[204] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=HPuSIXJaa9`.

[205] 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.

[206] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *ArXiv*, abs/2204.06125, 2022.

[207] Mengwei Ren, Mauricio Delbracio, Hossein Talebi, Guido Gerig, and Peyman Milanfar. Multiscale structure guided diffusion for image deblurring. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10721–10733, October 2023.

[208] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[209] 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.

[210] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[211] Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alexandru Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier García, Jianmo Ni, Andrew Chen, Kathleen Kenealy, J. Clark, Stephan Lee, Daniel H Garrette, James Lee-Thorp, Colin Raffel, Noam M. Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy B. Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. Scaling up models and data with `t5x` and `seqio`. *ArXiv*, abs/2203.17189, 2022.

[212] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[213] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[214] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Advances in Neural Information Processing Systems*, pages 2018–2028, 2017.

[215] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR, 2018.

[216] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023.

[217] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[218] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10, 2022.

[219] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep

language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[220] Tim Salimans, Han Zhang, Alec Radford, and Dimitris Metaxas. Improving GANs using optimal transport. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rkQkBnJAb`.

[221] Vishnu Sarukkai, Linden Li, Arden Ma, Christopher Ré, and Kayvon Fatahalian. Collage diffusion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4208–4217, 2024.

[222] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[223] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. *Advances in neural information processing systems*, 28, 2015.

[224] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[225] Rui Shu, Hung Bui, Hirokazu Narui, and Stefano Ermon. A DIRT-t approach to unsupervised domain adaptation. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=H1q-TM-AW`.

[226] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[227] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=St1giarCHLP`.

[228] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[229] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33: 12438–12448, 2020.

[230] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

[231] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=PxTIG12RRHS`.

[232] Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for machine learning*. Mit Press, 2011.

[233] Łukasz Struski, Marcin Mazur, Paweł Batorski, Przemysław Spurek, and Jacek Tabor. Bounding evidence and estimating log-likelihood in vae. In *International Conference on Artificial Intelligence and Statistics*, pages 5036–5051. PMLR, 2023.

[234] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.

[235] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[236] D'idac Sur'is, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. *ArXiv*, abs/2303.08128, 2023.

[237] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[238] Remi Tachet des Combes, Han Zhao, Yu-Xiang Wang, and Geoffrey J Gordon. Domain adaptation with conditional distribution matching and generalized label shift. *Advances in Neural Information Processing Systems*, 33, 2020.

[239] Shuhan Tan, Xingchao Peng, and Kate Saenko. Class-imbalanced domain adaptation: An empirical odyssey. In *European Conference on Computer Vision*, pages 585–602. Springer, 2020.

[240] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

[241] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[242] Ilya O Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. Adagan: Boosting generative models. *Advances in neural information processing systems*, 30, 2017.

[243] Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL `https://openreview.net/forum?id=CD9Snc73AW`. Expert Certification.

[244] Brian L. Trippe, Jason Yim, Doug Tischer, David Baker, Tamara Broderick, Regina Barzilay, and Tommi S. Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. In *The Eleventh*

International Conference on Learning Representations, 2023. URL `https://openreview.net/forum?id=6TxBxqNME1Y`.

[245] Christos Tsirigotis, Devon Hjelm, Aaron; Courville, and Pericles Mitkas. Objectives towards stable adversarial training without gradient penalties. In *Smooth Games Optimization and Machine Learning Workshop, NeurIPS*, 2019.

[246] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[247] Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *International Conference on Machine Learning*, pages 467–475. PMLR, 2014.

[248] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016.

[249] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[250] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[251] Ramakrishna Vedantam, Ian Fischer, Jonathan Huang, and Kevin Murphy. Generative models of visually grounded imagination. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HkCsm6lRb`.

[252] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

[253] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[254] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. *arXiv preprint arXiv:2311.12908*, 2023.

[255] Jing Wang, Jiahong Chen, Jianzhe Lin, Leonid Sigal, and Clarence W de Silva. Discriminative feature alignment: Improving transferability of unsupervised domain adaptation by gaussian-guided latent alignment. *Pattern Recognition*, 116:107943, 2021.

[256] Wei Wang, Yuan Sun, and Saman Halgamuge. Improving MMD-GAN training with repulsive loss function. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HygjqjR9Km`.

[257] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

[258] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[259] Jay Whang, Mauricio Delbracio, Hossein Talebi, Chitwan Saharia, Alexandros G. Dimakis, and Peyman Milanfar. Deblurring via stochastic refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16293–16303, June 2022.

[260] Jiqing Wu, Zhiwu Huang, Dinesh Acharya, Wen Li, Janine Thoma, Danda Pani Paudel, and Luc Van Gool. Sliced wasserstein generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[261] Tailin Wu, Takashi Maruyama, Long Wei, Tao Zhang, Yilun Du, Gianluca Iaccarino, and Jure Leskovec. Compositional generative inverse design. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=wmX0CqFSd7`.

[262] Yifan Wu, Ezra Winston, Divyansh Kaushik, and Zachary Lipton. Domain adaptation with asymmetrically-relaxed distribution alignment. In *International Conference on Machine Learning*, pages 6872–6881. PMLR, 2019.

[263] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International conference on machine learning*, pages 2635–2644. PMLR, 2016.

[264] Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. MARS: Markov Molecular Sampling for Multi-objective Drug Discovery. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=kHSu4ebxFXY`.

[265] Yilun Xu, Ziming Liu, Max Tegmark, and Tommi Jaakkola. Poisson flow generative models. *Advances in Neural Information Processing Systems*, 35: 16782–16795, 2022.

[266] Yilun Xu, Ziming Liu, Yonglong Tian, Shangyuan Tong, Max Tegmark, and Tommi Jaakkola. Pfgm++: Unlocking the potential of physics-inspired generative models. In *International Conference on Machine Learning*, pages 38566–38591. PMLR, 2023.

[267] Jason Yim, Andrew Campbell, Emile Mathieu, Andrew YK Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S Veeling, Frank Noé, et al. Improved motif-scaffolding with se (3) flow matching. *arXiv preprint arXiv:2401.04082*, 2024.

[268] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J

Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.

[269] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

[270] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *ArXiv*, abs/1212.5701, 2012.

[271] Houssam Zenati, Manon Romain, Chuan-Sheng Foo, Bruno Lecouat, and Vijay Chandrasekhar. Adversarially learned anomaly detection. In *2018 IEEE International conference on data mining (ICDM)*, pages 727–736. IEEE, 2018.

[272] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023. `https://D2L.ai`.

[273] Dinghuai Zhang, Ricky TQ Chen, Nikolay Malkin, and Yoshua Bengio. Unifying generative models with gflownets. *ArXiv*, abs/2209.02606, 2022.

[274] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *ArXiv*, abs/2302.05543, 2023.

[275] Qinsheng Zhang, Jiaming Song, Xun Huang, Yongxin Chen, and Ming-Yu Liu. Diffcollage: Parallel generation of large content with diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10188–10198. IEEE, 2023.

[276] Weiwei Zhang, Jian Sun, and Xiaoou Tang. Cat head detection-how to effectively exploit shape and texture features. In *European Conference on Computer Vision*, pages 802–816. Springer, 2008.

[277] Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. In *Advances in Neural Information Processing Systems*, pages 8568–8579, 2018.

[278] Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *International Conference on Machine Learning*, pages 7523–7532. PMLR, 2019.

[279] Андрей Николаевич Колмогоров. *Основные понятия теории вероятностей.* Наука, 1936.

[280] Лев Семёнович Понтрягин, Владимир Григорьевич Болтянский, Реваз Валерианович Гамкрелидзе, and Евгений Фролович Мищенко. *Математическая теория оптимальных процессов*. Наука, 1961.

Appendices

Appendix A

Pairwise-Discriminator Objectives for Generative Adversarial Networks

A.1 Proof of Proposition 3.5.1

Recall that we consider a particular instance of the game (3.7) corresponding to:

$$f_1(t) = -\log(t),\ f_2(t) = -\log(1-t),\ g(t) = \log(t).$$

First, we expand the expression for the discriminator loss (3.7a) in PairSGAN:

$$\begin{aligned} L_{\mathcal{D}}(D, q) = &+ \mathbb{E}_{p(x)p(y)}\left[-\log D(x,y)\right] + \mathbb{E}_{q(x)q(y)}\left[-\log D(x,y)\right] \\ &+ \mathbb{E}_{p(x)q(y)}\left[-\log(1-D(x,y))\right] + \mathbb{E}_{q(x)p(y)}\left[-\log(1-D(x,y))\right]. \end{aligned}$$

We expand all expectations as the integrals and obtain:

$$\begin{aligned} L_{\mathcal{D}} = \iint \Big[&-\log D(x,y)\Big(p(x)p(y) + q(x)q(y)\Big) \\ &- \log(1 - D(x,y))\Big(p(x)q(y) + q(x)p(y)\Big)\Big]\, dx\, dy. \end{aligned}$$

We minimize the integral by minimizing the expression inside the integral w.r.t $D(x, y)$

point-wise. Solving for the optimal $D(x,y) \in (0,1)$, we obtain:

$$D^*(x,y) = \frac{p(x)p(y) + q(x)q(y)}{p(x)p(y) + q(x)q(y) + p(x)q(y) + q(x)p(y)}.$$

We rewrite this expression as the function of the mixture distributions (3.3)

$$D^*(x,y) = \frac{M^+_{p,q}(x,y)}{2 \cdot M_{p,q}(x,y)}.$$

Next, we substitute D^* to the generator loss (3.7b):

$$\begin{aligned} L_{\mathcal{G}}(D^*, q) = &+ \mathbb{E}_{p(x)p(y)}\left[\log D^*(x,y)\right] + \mathbb{E}_{q(x)q(y)}\left[\log D^*(x,y)\right] \\ &+ \mathbb{E}_{p(x)q(y)}\left[-\log D^*(x,y)\right] + \mathbb{E}_{q(x)p(y)}\left[-\log D^*(x,y)\right]. \end{aligned}$$

We add and substract the terms $\mathbb{E}_{p(x)p(y)}\left[\log D^*(x,y)\right] + \mathbb{E}_{q(x)q(y)}\left[\log D^*(x,y)\right]$ to the expression above, and rewrite it as:

$$\begin{aligned} L_{\mathcal{G}}(D^*, q) &= 4\,\mathbb{E}_{M^+_{p,q}(x,y)}\left[\log D^*(x,y)\right] - 4\,\mathbb{E}_{M_{p,q}(x,y)}\left[\log D^*(x,y)\right] \\ &= 4\,\mathbb{E}_{M^+_{p,q}(x,y)}\left[\log \frac{M^+_{p,q}(x,y)}{M_{p,q}(x,y)} - \log(2)\right] \\ &\quad + 4\,\mathbb{E}_{M_{p,q}(x,y)}\left[\log \frac{M_{p,q}(x,y)}{M^+_{p,q}(x,y)} + \log(2)\right]. \end{aligned}$$

After cancelling out the constant $\log(2)$ terms, the two expectations above give KL and reverse-KL divergences between $M_{p,q}$ and $M^+_{p,q}$. Thus, we have shown that

$$L_{\mathcal{G}}(D^*, q) = 4 \cdot \Big(\mathrm{KL}(M^+_{p,q} \| M_{p,q}) + \mathrm{KL}(M_{p,q} \| M^+_{p,q})\Big).$$

The symmetrized KL-divergence above is non-negative and is equal to zero iff

$$M_{p,q}(x,y) = M^+_{p,q}(x,y) \quad \forall\, x, y.$$

We transform the last equation in the following way:

$$\frac{1}{2}(M^-_{p,q}(x,y) + M^+_{p,q}(x,y)) = M^+_{p,q}(x,y)$$

$$\Updownarrow$$

$$M^-_{p,q}(x,y) = M^+_{p,q}(x,y)$$

$$\Updownarrow$$

$$p(x)q(y) + q(x)p(y) = p(x)p(y) + q(x)q(y)$$

$$\Updownarrow$$

$$\big(p(x) - q(x)\big) \cdot \big(p(y) - q(y)\big) = 0$$

The last equation holds for all (x, y) iff $p(\cdot) = q(\cdot)$.

A.2 Proof of Proposition 3.5.2

We expand the expression for the discriminator loss for PairGAN-Z (3.8):

$$\begin{aligned} L_{\mathcal{G}}(D, q) = &+ \mathbb{E}_{p(x)p(y)}\left[\log D(x,y)\right] + \mathbb{E}_{q(x)q(y)}\left[\log D(x,y)\right] \\ &- \mathbb{E}_{p(x)q(y)}\left[\log D(x,y)\right] - \mathbb{E}_{q(x)p(y)}\left[\log D(x,y)\right]. \end{aligned}$$

We expand all expectations as integrals and obtain:

$$L_{\mathcal{G}} = \iint \Big[\Big(p(x)p(y) + q(x)q(y) - p(x)q(y) - q(x)p(y)\Big) \log D(x,y)\Big]\, dx\, dy.$$

We introduce the function $F(x, y)$ as:

$$F(x,y) = p(x)p(y) + q(x)q(y) - p(x)q(y) - q(x)p(y),$$

and re-write the loss as

$$L_{\mathcal{G}} = \iint \Big[F(x,y) \log D(x,y)\Big]\, dx\, dy.$$

Recall, that in PairGAN-Z the discriminator aims to maximize $L_{\mathcal{G}}$. Therefore, our goal is to maximize the expression in the integral pointwise w.r.t. $D(x, y) \in [\varepsilon, 1]$. The optimal discriminator D^* is given by[1]:

$$D^*(x, y) = \begin{cases} 1, & F(x, y) \geq 0 \\ \varepsilon, & F(x, y) < 0 \end{cases}.$$

The logarithm of D^* can be written as:

$$\log D^*(x, y) = \log(\varepsilon) \cdot \mathbb{I}[F(x, y) < 0].$$

We substitute $\log D^*$ to the generator loss and obtain:

$$L_{\mathcal{G}}(D, q) = \log(\varepsilon) \iint F(x, y) \cdot \mathbb{I}[F(x, y) < 0] \, dx \, dy,$$

where the integral is exactly the negative total variation distance between $M^+_{p,q}$ and $M^-_{p,q}$. Thus, we have shown that:

$$L_{\mathcal{G}}(D^*, q) = -\log(\varepsilon) \cdot \delta_{\text{TV}}(M^+_{p,q} \| M^-_{p,q}).$$

Similarly to the case of symmetrized KL-divergence in Proposition 3.5.1 proved in Appendix A.1, the total variation distance is non-negative and equals to zero iff

$$M^+_{p,q} = M^-_{p,q} \iff p = q.$$

A.3 Hessian of the Generator Loss

For a parametric generator $q(\cdot; \theta)$, we expand the generator loss (3.7b):

$$L_{\mathcal{G}}(D, q_\theta) = + \mathbb{E}_{p(x)p(y)} \left[g(D(x, y)) \right] + \mathbb{E}_{q(x;\theta)q(y;\theta)} \left[g(D(x, y)) \right]$$

[1]We restrict the discriminator output $D(x, y) \geq \varepsilon$, in order for the discriminator loss to be bounded. For $F(x, y) < 0$, an unrestricted discriminator can drive $\log D(x, y)$ to $-\infty$.

$$- \mathbb{E}_{p(x)q(y;\theta)}\left[g(D(x,y))\right] - \mathbb{E}_{q(x;\theta)p(y)}\left[g(D(x,y))\right].$$

Now we compute the gradient, by expanding each expectation to an integral and exchanging the order of differentiation and integration:

$$\nabla_\theta L_{\mathcal{G}}(D, q_\theta) = \iint \Big(\nabla_\theta q(x;\theta) \cdot q(y;\theta) + q(x;\theta) \cdot \nabla_\theta q(y;\theta) \\ - p(x) \cdot \nabla_\theta q(y;\theta) - \nabla_\theta q(x;\theta) \cdot p(y) \Big) g(D(x,y))\, dx\, dy.$$

We compute the Hessian by differentiating the gradient:

$$\nabla_\theta L_{\mathcal{G}}(D, q_\theta) = \iint \Bigg[\Big(2[\nabla_\theta q(x;\theta)][\nabla_\theta q(y;\theta)]^T + \nabla^2_{\theta\theta} q(x;\theta) \cdot q(y;\theta) + q(x;\theta) \cdot \nabla^2_{\theta\theta} q(y;\theta) \\ - \nabla^2_{\theta\theta} q(x;\theta) \cdot p(y) - p(x) \cdot \nabla^2_{\theta\theta} q(y;\theta) \Big) g(D(x,y)) \Bigg]\, dx\, dy.$$

Our final step is to substitute $\theta = \theta^*$. Since $q(\cdot;\theta^*) = p(x)$, in the expression above, all terms except $2[\nabla_\theta q(x;\theta)][\nabla_\theta q(y;\theta)]^T$ cancel out and we obtain equation (3.9).

A.4 Proof of Proposition 3.5.3

This sections provides the proof of the Proposition 3.5.3. The proof relies on the following result by Mescheder et al. [168].

Theorem A.4.1 (Theorem A.3 of Mescheder et al. [168]). *Let $F(\alpha, \gamma)$ define a $\mathcal{C}^1$-mapping that maps some domain Ω to itself. Assume that there is a local neighborhood U of 0 such that $F(0, \gamma) = (0, \gamma)$ for $\gamma \in U$. Moreover, assume that all eigenvalues of $J := \nabla_\alpha F(\alpha, 0)\mid_{\alpha=0}$ have absolute value smaller than 1. Then the fixed point iteration defined by F is locally convergent to $\mathcal{M} := \{(0, \gamma) \mid \gamma \in U\}$ with linear convergence rate in a neighborhood of $(0, 0)$. Moreover, the convergence rate is $|\lambda_{\max}|$ with $\lambda_{\max}$ the eigenvalue of J with largest absolute value.*

Gradient descent update. We denote the gradient of the loss $L_\mathcal{G}$ w.r.t. θ as:

$$g(\theta;\psi) = \nabla_\theta L_\mathcal{G}(D_\psi, q(\cdot;\theta)).$$

We consider the update operator corresponding to the gradient descent for $L_\mathcal{G}$ w.r.t. θ:

$$F_h(\theta;\psi) = \theta - h \cdot g(\theta;\psi), \tag{A.1}$$

where $h > 0$ is the step size (learning rate). To understand the convergence of the gradient descent we examine the eigenvalues of the Jacobian $\nabla_\theta F_h(\theta;\psi)$ at θ^*. We notice that $\nabla_\theta F_h(\theta^*;\psi)$ is given by

$$\nabla_\theta F_h(\theta^*;\psi) = I - h \cdot H(\theta^*;\psi),$$

where $H(\theta^*;\psi)$ is the Hessian given by (3.9). From (3.9), we observe that $H(\theta^*;\psi)$ is a symmetric matrix and thus its eigenvalues are real numbers.

An eigenvalue λ of the Jacobian $\nabla_\theta F_h(\theta^*;\psi)$ is given by:

$$\lambda = 1 - h \cdot \mu, \tag{A.2}$$

where μ is the corresponding eigenvalue of the Hessian $H(\theta^*;\psi)$

Below we provide the proof for Proposition 3.5.3.

Proposition 3.5.3. Suppose that $\theta^* \in \mathcal{M}_G$ and a pair (ψ_0, θ^*) satisfies:

$$u^T[H(\theta^*;\psi_0)]u > 0 \quad \forall u \notin \mathcal{T}_{\theta^*}\mathcal{M}_G. \tag{A.3}$$

Then, with fixed $\psi = \psi_0$, gradient descent w.r.t. θ for (3.7b) converges to $\mathcal{M}_G$ in a neighborhood of θ^* provided a small enough learning rate. Moreover, the rate of convergence is at least linear.

Proof. Following Mescheder et al. [168], in order to apply Theorem A.4.1, we choose local coordinates α, γ for θ : $\theta(\alpha,\gamma)$. Without loss of generality (see Remark A.6 of

Mescheder et al. [168]), we can assume that

$$\theta^* = 0, \quad \mathcal{M}_G = \mathcal{T}_\theta^* \mathcal{M}_G = \{0\}^k \times \mathbb{R}^{n-k},$$
$$\theta(\alpha, \gamma) = [\alpha, \gamma]^T, \quad \alpha \in \mathbb{R}^k,\ \gamma \in \mathbb{R}^{n-k}.$$

In the local coordinates a vector $u \notin \mathcal{T}_{\theta^*}\mathcal{M}_G$ has the form $u = (\widetilde{u}, 0)$, where $\widetilde{u} \in \mathbb{R}^k$. Let $\widetilde{H}$ denote the sub-matrix of the Hessian $H(\theta^*(\alpha, \gamma); \psi_0)$ corresponding to the coordinates α. Then, condition (A.3) transforms into:

$$\widetilde{u}^T \widetilde{H} \widetilde{u} > 0 \quad \forall\, \widetilde{u},$$

which implies that $\widetilde{H}$ has only positive eigenvalues.

In order to apply Theorem A.4.1, we have to show that all eigenvalues λ of the Jacobian $\nabla_\alpha F_h(\theta(\alpha, \gamma); \psi_0)|_{\alpha=0}$ have absolute value smaller than 1. Given that $\widetilde{H}$ has only positive eigenvalues, the inequality $\lambda < 1$ is guaranteed by equation (A.2).Then it is sufficient for us to choose learning rate h that guarantees $\lambda > -1$. The inequality:

$$h < \frac{2}{\widetilde{\mu}_{\max}},$$

ensures that $\lambda > -1$.

By Theorem A.4.1 the fixed point iteration for F_h converges to $\mathcal{M}_G$.

$\square$

A.5 Poof of Proposition 3.5.6

This section provides a proof of Proposition 3.5.6.

We introduce function space operators:

$$\Gamma_1, \Gamma_2 : \mathbb{R}^n \to \mathcal{F}(\mathcal{X}),$$
$$\Gamma_1 : \quad (\Gamma_1[u])(x) = [g_1(x; \theta)]^T u = [\nabla_\theta q(x; \theta^*)]^T u,$$
$$\Gamma_2 : \quad (\Gamma_2[u])(x) = [g_2(x; \theta)]^T u = [\nabla_\theta \log q(x; \theta^*)]^T u.$$

Informally, Γ_1, Γ_2 are matrices of size $|\mathcal{X}| \times n$ where the first dimension can be infinite. Let us describe some properties of Γ_1 and Γ_2.

$$A_1^* = \Gamma_1 \Gamma_1^T, \qquad A_2^* = \Gamma_2 \Gamma_2^T,$$

$$\nabla_\theta \log q(x;\theta) = \frac{1}{q(x;\theta)} \nabla_\theta q(x;\theta) \implies \Gamma_2 = D_q \Gamma_1,$$

where D_q is a diagonal operator

$$D_q(x,y) = I[x=y] \frac{1}{q(x;\theta^*)},$$

with positive values on diagonal[2].

With Γ_1 we can represent the function-space perturbation $\varepsilon_u(x) = [\nabla_\theta q(x;\theta^*)]^T u$ as

$$\varepsilon_u = \Gamma_1 u$$

and re-write Definition 3.5.4 as

$$\Gamma_1 u \neq 0 \implies u^T \Gamma_1^T A \Gamma_1 u > 0. \tag{A.4}$$

By substituting $A = A_1^*$ in (A.4) we obtain:

$$\Gamma_1 u \neq 0 \implies \|\Gamma_1^T \Gamma_1 u\|^2 > 0.$$

This implication holds since $\mathrm{Ker}(\Gamma_1^T) \perp \mathrm{Im}(\Gamma_1)$.

By substituting $A = A_2^*$ in (A.4), we obtain:

$$\Gamma_1 u \neq 0 \implies \|\Gamma_1^T D_q \Gamma_1 u\|^2 > 0,$$

or equivalently:

$$\Gamma_1 u \neq 0 \implies \left\| \left(D_q^{\frac{1}{2}} \Gamma_1\right)^T \left(D_q^{\frac{1}{2}} \Gamma_1\right) u \right\|^2 > 0.$$

[2] q must be positive for $\log q$ to be defined.

This implication holds since $\Gamma_1 u \neq 0 \Rightarrow D_q^{\frac{1}{2}} \Gamma_1 u \neq 0$ and $\mathrm{Ker}(D_q^{\frac{1}{2}} \Gamma_1^T) \perp \mathrm{Im}(D_q^{\frac{1}{2}} \Gamma_1)$.

The minimality of the operators follows from the fact that:

$$\mathrm{rank}(A_1^*) = \mathrm{rank}(A_2^*) = \dim(W_q(\theta^*)). \tag{A.5}$$

Recall, that in Section 3.5.3 we denoted the components of the gradient $\nabla_\theta q(x; \theta^*)$ as function-space vectors $\alpha_1, \dots, \alpha_n \in \mathcal{F}(\mathcal{X})$:

$$\alpha_i(x) = \frac{\partial}{\partial \theta_i} q(x; \theta^*).$$

Next, we observe that

$$W_q(\theta^*) = \mathrm{span}(\alpha_1, \dots, \alpha_n),$$

$$A_1^* = \sum_{i=1}^{n} \alpha_i \alpha_i^T, \quad A_2^* = \sum_{i=1}^{n} (D_q \alpha_i)(D_q \alpha_i)^T.$$

Equation (A.5) follows from the above representation for A_1^*, A_2^* and $W_q(\theta^*)$.

Now, we derive the loss function (3.12). We substitute the discriminator-operator A_i^* into the loss (3.7b):

$$\begin{aligned} L_i^*(\theta) =& \big\langle\, p - q \,,\, A_i^*(p - q) \,\big\rangle = \big\langle\, p - q \,,\, \Gamma_i \Gamma_i^T (p - q) \,\big\rangle = \|\Gamma_i^T (p - q)\|^2 = \|\Gamma_i^T p - \Gamma_i^T q\|^2 \\ =& \Big\| \mathbb{E}_{p(x)} \Big[g_i(x; \theta) \Big] - \mathbb{E}_{q(x;\theta)} \Big[g_i(x; \theta) \Big] \Big\|^2 \end{aligned}$$

A.6 Proof of Proposition 3.5.7

This Section provides the proof of Proposition 3.5.7.

The gradient of the loss (3.14) w.r.t. p_i is given

$$\nabla_{p_i} \mathcal{L}(p_1, \dots, p_N | D) = 2(N-1) A_D^g p_i - \sum_{\substack{s=1 \\ s \neq i}}^{N} 2 A_D^g p_s.$$

For $i \leq k$ we split the sum into two:

$$\nabla_{p_i}\mathcal{L}(p_1,\dots,p_N|D) = 2(N-1)A_D^g p_i - 2\sum_{\substack{s=1\\ s\neq i}}^{k} A_D^g p_s - 2\sum_{s=k+1}^{N} A_D^g p_s.$$

Next, we use that for $s \leq k : p_s = p_i$, therefore:

$$\nabla_{p_i}\mathcal{L}(p_1,\dots,p_N|D) = 2(N-K)A_D^g p_i - 2\sum_{s=k+1}^{N} A_D^g p_s.$$

Finally, we observe that both terms above take the same value for all $1 \leq i \leq k$. This observation concludes the proof.

A.7 Toy Example Details

This section provides a detailed description of the toy examples shown in Section 3.1 (Figure 3-1), Section 3.5.2 (Figure 3-3), and Section A.6 (Figure 3-4).

Section A.7.1 describes the toy setup for GANs and the models (unary and pairwise) used to produce Figure 3-1 and Figure 3-3. Section A.7.2 describes the toy example for multiple distributions alignment (see Section A.6) and the models used to produce Figure 3-4.

The implementation of the described toy examples is provided in the codebase accompanying the Chapter.

A.7.1 DiracGAN & DiracPairGAN

Mescheder et al. [168] proposed DiracGAN a toy example of GAN, where both target distribution p and generative model q are defined by delta functions (i.e. each concentrated on a single point):

$$p(x) = \delta(x - x_{\text{real}}) \qquad q(x) = \delta(x - x_{\text{fake}}).$$

Here, $x_{\text{real}} = 0$ is a fixed real example, and x_{fake} is a free parameter of the generative model q. In this model, the distributions are aligned when $x_{\text{fake}} = x_{\text{real}}$.

Below we first consider the adversarial training objective for DiracGAN with a simple parameterization of the discriminator used in Mescheder et al. [168]. Then we introduce DiracPairGAN a modified formulation of DiracGAN with a pairwise discriminator and generator loss of the form (3.6).

DiracGAN [168]

In DiracGAN, the discriminator is defined as linear function $D_\psi(x) = \psi \cdot x$, parameterized by a single number ψ. D_ψ defines a linear classifier which estimates the probability of a given sample x being real/fake:

$$P_D(t = \text{real} \,|\, x, \psi) = \sigma(D_\psi(x)), \qquad P_D(t = \text{fake} \,|\, x, \psi) = \sigma(-D_\psi(x)),$$

where $t \in \{\text{real}, \text{fake}\}$ is a class label and $\sigma(\cdot)$ is the sigmoid function.

The discriminator is trained by maximizing log-likelihood:

$$\mathcal{L}(\psi, x_{\text{fake}}) = \log P_D(t = \text{real} \,|\, x_{\text{real}}, \psi) + \log P_D(t = \text{fake} \,|\, x_{\text{fake}}, \psi). \tag{A.6}$$

The generator x_{fake} and the discriminator ψ compete in a zero-sum game:

$$\min_{x_{\text{fake}}} \max_{\psi} \mathcal{L}(\psi, x_{\text{fake}}).$$

Note, that the first term in (A.6) is constant since $x_{\text{real}} = 0$ is constant. Therefore, $\mathcal{L}$ can be equivalently re-written as:

$$\widetilde{\mathcal{L}}(\psi, x_{\text{fake}}) = -\log(1 + \exp\{\psi \cdot x_{\text{fake}}\}).$$

It is easy to see that the alignment $x_{\text{fake}} = 0$ is not preserved in DiracGAN unless $\psi = 0$. To see that it is enough to check that

$$\psi \neq 0 \implies \frac{\partial}{\partial x} \widetilde{\mathcal{L}}(\psi, 0) \neq 0.$$

DiracPairGAN

In DiracPairGAN, we define a symmetric pairwise discriminator $D_\psi(x, y) = \psi \cdot |x - y|^\gamma$ where with a single parameter ψ and a hyperparameter $\gamma \geq 1$.

$D_\psi(x, y)$ denotes a probabilistic classifier which estimates the probability of a given pair of samples (x, y) coming from the same distribution rather than different distributions.

$$P_D(t = \text{same} \,|\, x, y, \psi) = \sigma(D_\psi(x, y)), \qquad P_D(t = \text{diff} \,|\, x, y, \psi) = \sigma(-D_\psi(x, y)),$$

where $t \in \{\text{same}, \text{diff}\}$ denotes the class label.

The negative log-likelihood loss for the pairwise discriminator is given by

$$\begin{aligned}\mathcal{L}_D(\psi, x_{\text{fake}}) = &- \log P_D(t = \text{same} \,|\, x_{\text{real}}, x_{\text{real}}, \psi) - \log P_D(t = \text{same} \,|\, x_{\text{fake}}, x_{\text{fake}}, \psi) \\ &- \log P_D(t = \text{diff} \,|\, x_{\text{real}}, x_{\text{fake}}, \psi) - \log P_D(t = \text{diff} \,|\, x_{\text{fake}}, x_{\text{real}}, \psi).\end{aligned} \tag{A.7}$$

We define an instance of PairGAN generator loss (3.7b):

$$\begin{aligned}\mathcal{L}_G(\psi, x_{\text{fake}}) = &- \log P_D(t = \text{diff} \,|\, x_{\text{real}}, x_{\text{real}}, \psi) - \log P_D(t = \text{diff} \,|\, x_{\text{fake}}, x_{\text{fake}}, \psi) \\ &+ \log P_D(t = \text{diff} \,|\, x_{\text{real}}, x_{\text{fake}}, \psi) + \log P_D(t = \text{diff} \,|\, x_{\text{fake}}, x_{\text{real}}, \psi).\end{aligned} \tag{A.8}$$

In general formulation of PairGAN the generator and the discriminator compete in a non-zero sum game:

$$\begin{aligned}&\min_{\psi} \mathcal{L}_D(\psi, x_{\text{fake}}) \\ &\min_{x_{\text{fake}}} \mathcal{L}_G(\psi, x_{\text{fake}}).\end{aligned}$$

We note, that our choice of parameterization allows us to re-write the game in a simplified form. Indeed, the first two terms in both (A.7) and (A.8) are constant and all equal to $-\log(\frac{1}{2})$ since $D_\psi(x, x) = 0$. Thus, the only difference in the losses (A.7) and (A.8) is in the signs of the third and the fourth terms. Observing this, we obtain an equivalent zero-sum game:

$$\min_{x_{\text{fake}}} \max_{\psi} \widetilde{\mathcal{L}}(\psi, x_{\text{fake}}),$$

where

$$\widetilde{\mathcal{L}}(\psi, x_{\text{fake}}) = -\log(1 + \exp\{\psi \cdot |x_{\text{fake}}|^{\gamma}\}).$$

In DiracPairGAN the alignment is preserved for any ψ since $\widetilde{\mathcal{L}}(\psi, x_{\text{fake}})$ is a function of absolute value of x_{fake} and, consequently,

$$\frac{\partial}{\partial x}\widetilde{\mathcal{L}}(\psi, 0) = 0 \quad \forall\, \psi.$$

A.7.2 Multiple Distributions

Below we consider the toy example demonstrating adversarial alignment of multiple distributions (see Section A.6).

Consider, three delta functions: p_1, p_2, p_3:

$$p_i(x) = \delta(x - x_i),$$

parameterized by real numbers x_1, x_2, and x_3 respectively. The goal of the toy models described below is to align the three distributions with one another, i.e. reach a situation where $x_1 = x_2 = x_3$.

Unary discriminator

For the three distributions problem we utilize a unary discriminator

$$D_{\boldsymbol{\psi}}(x) = [s_1(x, \boldsymbol{\psi}), s_2(x, \boldsymbol{\psi}), s_3(x, \boldsymbol{\psi})],$$

which defines a 3-class softmax classifier

$$P_D(t = i \,|\, x, \boldsymbol{\psi}) = \frac{\exp\{s_i(x, \boldsymbol{\psi})\}}{\sum\limits_{j=1}^{3} \exp\{s_j(x, \boldsymbol{\psi})\}},$$

where $t \in \{1, 2, 3\}$ is a class label and $P_D(t = i \,|\, x, \boldsymbol{\psi})$ is an estimate of the probability of a given sample x coming from $p_i(\cdot)$.

We define the logits s_i as quadratic parametric functions

$$s_i(x, \boldsymbol{\psi}) = a_i x^2 + b_i x + c_i, \quad i \in \{1, 2, 3\}$$

with $\boldsymbol{\psi}$ defined as a vector of all nine parameters

$$\boldsymbol{\psi} = [a_1, b_1, c_1, a_2, b_2, c_2, a_3, b_3, c_3]^T.$$

One can think of $D_\psi(x)$ as a linear classifier operating on a non-linear feature representation $\boldsymbol{\phi}(x) = [x^2, x, 1]^T$:

$$s_i(x, \boldsymbol{\psi}) = [a_i, b_i, c_i][x^2, x, 1]^T.$$

Note that with the described parameterization, the unary discriminator is powerful enough to represent a zero-error decision boundary for any location of the points x_1, x_2, x_3.

Similarly to the examples above, we train discriminator by maximizing log-likelihood

$$\mathcal{L}(\boldsymbol{\psi}, x_1, x_2, x_3) = \sum_{i=1}^{3} \log P_D(t = i \,|\, x_i, \boldsymbol{\psi}), \tag{A.9}$$

and define a zero-sum game between the points x_1, x_2, x_3 and discriminator $\boldsymbol{\psi}$:

$$\min_{x_1, x_2, x_3} \max_{\boldsymbol{\psi}} \mathcal{L}(\boldsymbol{\psi}, x_1, x_2, x_3).$$

Pairwise discriminator

Now, we define a multiple distributions model with a pairwise discriminator. Again, we utilize the same pairwise discriminator as in Section A.7.1: $D_\psi(x, y) = \psi \cdot |x - y|^\gamma$:

$$P_D(t = \text{same} \,|\, x, y, \psi) = \sigma(D_\psi(x, y)),$$
$$P_D(t = \text{diff} \,|\, x, y, \psi) = \sigma(-D_\psi(x, y)).$$

We use the following weighted negative log-likelihood objective for the discriminator:

$$\mathcal{L}_D(\psi, x_1, x_2, x_3) = -2\sum_{i=1}^{3} \log P_D(t = \text{same} \,|\, x_i, x_i, \psi) - \sum_{i \neq j} \log P_D(t = \text{diff} \,|\, x_i, x_j, \psi), \tag{A.10}$$

computed for same distribution pairs (x_i, x_i) and different distributions pairs $(x_i, x_j) : i \neq j$. In order to equalize the $3 : 6$ ratio of the number of $\{t = \text{same}\}$-pairs to the number of $\{t = \text{diff}\}$-pairs, we virtually augment the set of same distribution pair by using the weights $w_{\text{same}} = 2$, $w_{\text{diff}} = 1$.

Next, we define a non-zero sum game between ψ and x_1, x_2, x_3:

$$\min_{\psi} \mathcal{L}_D(\psi, x_1, x_2, x_3),$$
$$\min_{x_1, x_2, x_3} \mathcal{L}_G(\psi, x_1, x_2, x_3),$$

where the loss for x_1, x_2, x_3 is an instance of the adversarial loss (3.14) introduced in Section A.6:

$$\mathcal{L}_G(\psi, x_1, x_2, x_3) = -2\sum_{i=1}^{3} \log P_D(t = \text{diff} \,|\, x_i, x_i, \psi) + \sum_{i \neq j} \log P_D(t = \text{diff} \,|\, x_i, x_j, \psi). \tag{A.11}$$

Since $D_\psi(x, x) = 0$, the first terms in both (A.10) and (A.11) are constant. Therefore, the considered setup can be reduced to a zero-sum game:

$$\min_{x_1, x_2, x_3} \max_{\psi} \widetilde{\mathcal{L}}(\psi, x_1, x_2, x_3),$$

where the loss $\widetilde{\mathcal{L}}$ is given by

$$\widetilde{\mathcal{L}}(\psi, x_1, x_2, x_3) = -\sum_{i \neq j} \log(1 + \exp\{\psi \cdot |x_i - x_j|^\gamma\}).$$

Comments on domain-adversarial methods for domain-adaptation. The loss (A.9) used in the unary discriminator above is a simplified instance of the domain loss used in domain adversarial neural networks [DANN, 73]. In domain adversarial training notation, $\{p_i(x)\}_{i=1}^3$ represent the distribution of representations in different domains. The domain loss (A.9) is one of the terms in DANN objective. Optimization of the domain loss (A.9) w.r.t. parameters of distribution $\{p_i(x)\}_{i=1}^3$ can be interpreted as minimization of a divergence between the distributions. This regularization mechanism is expected to make the learned representation x invariant across the domains.

Note that this example represents only a part of the adversarial objective used in DANN. The full adversarial training procedure is defined as a three-player game between a feature extractor, a classifier and a domain discriminator. In contrast, here we only focus on one loss term which is responsible for the alignment of the distributions. Moreover, in DANN the distributions $\{p_i(x)\}_{i=1}^3$ are interconnected through the shared parameterization, while in the presented toy model we consider independently parameterized distributions. We believe that understanding the mechanics of the alignment with this toy example is important for the analysis and further development of domain-adversarial methods.

A.8 Experiment Details

A.8.1 Fixed Generator Matching Experiment

We provide additional results on fixed generator matching problem for WGANGP, LSGAN, and RaSGAN not shown in the main text. Figure A-1 shows trajectory of α over the course of training for all models.

For each model, we train all parameters of discriminator and the single parameter α of the re-parameterized generator with SGD for 50 000 steps. We use mini-batches of size 128. Table A.1 provides learning rates used for each model. We used the same learning rates for PairSGAN, SGAN, WGAN-GP, and XORGAN. We adjusted the learning rates for LSGAN, RSGAN, and RaSGAN to ensure comparable convergence rates relative to other models. For WGAN-GP we set gradient penalty regularizer parameter $\lambda = 10$ and perform $n_{\mathrm{D}} = 5$ discriminator updated per generator update.

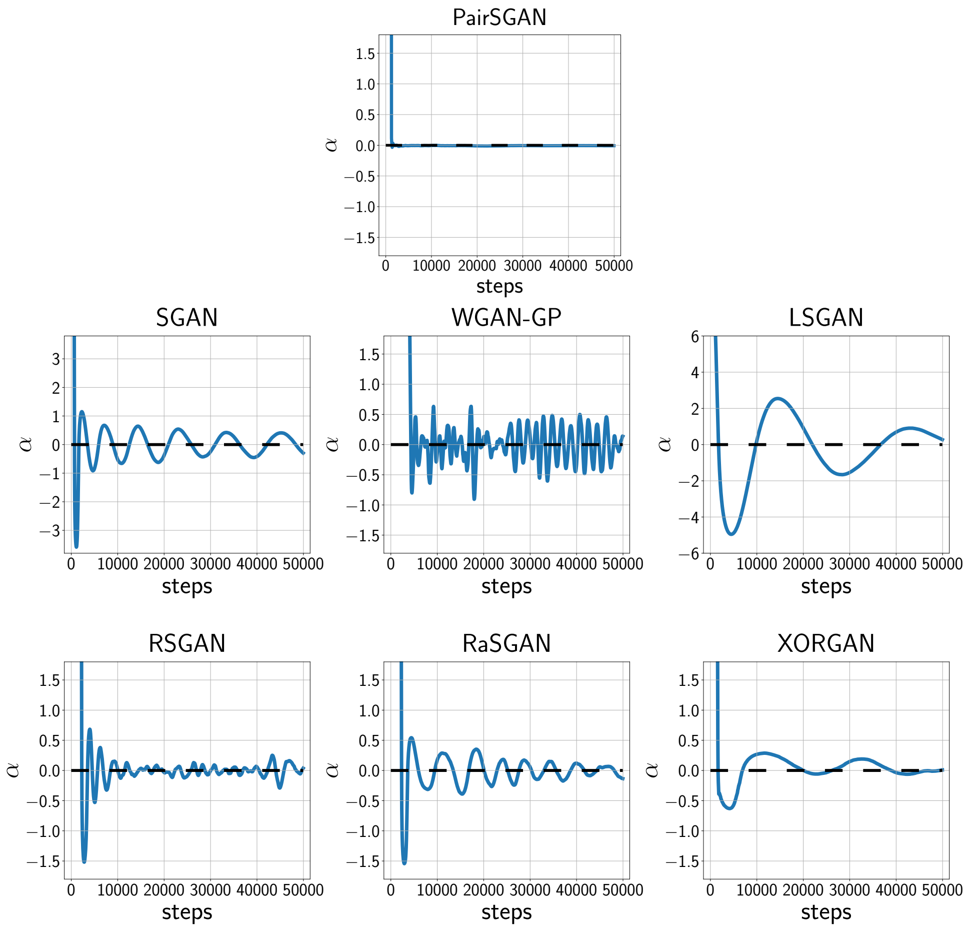


Figure A-1: Training curves for DCGAN generator re-parameterized with a single parameter α.

Table A.1: Learning rates for discriminator (η_D) and α (η_α) for different GAN models

Model	η_D	η_α
PairSGAN	0.004	0.32
SGAN	0.004	0.32
WGAN-GP	0.004	0.32
LSGAN	0.000032	0.16
RSGAN	0.001	0.1
RaSGAN	0.001	0.1
XORGAN	0.004	0.32

For all models we use the standard DCGAN architecture for the generator. We use the standard DCGAN discriminator architecture for all models relying on unary discriminators: unary GAN models (SGAN, WGAN-GP, LSGAN), relativistic GANs (RSGAN, RaSGAN), and XORGAN. For PairGAN we use the same pairwise discriminator architecture as in CIFAR-10 expereiments (see Section A.8.2).

A.8.2 Real World Datasets Experiment

In this section, we will describe in details two specific practical setups of PairGAN that we experiment with on the real world datasets, namely one for CIFAR-10 [130] and another for CAT [276].

CIFAR-10 One way to parameterize a pairwise discriminator is $D(x, y) = D_b([D_u(x), D_u(y)])$, where $D_u(\cdot)$ is the DCGAN unary discriminator that takes a single image as an input and returns a multi-dimensional output instead of just one-dimensional. $D_b([\cdot, \cdot])$ is a fully connected network that takes the concatenation of the two $D_u(\cdot)$ output as input and returns a single scalar.

With this setup, we train the discriminator and the generator with the following mini-batch estimators[3] of the PairSGAN loss:

$$\begin{aligned}\widetilde{\mathcal{L}}_D =& \frac{1}{N}\sum_{i=1}^{N}\Big[-\log D(u_i^1, u_i^2) - \log D(v_i^1, v_i^2) - \log(1-D(u_i^1, v_i^2)) - \log(1-D(v_i^1, u_i^2))\Big] \\ \widetilde{\mathcal{L}}_G =& \frac{1}{N}\sum_{i=1}^{N}\Big[\log D(v_i^1, v_i^2) - \log D(u_i^1, v_i^2) - \log D(v_i^1, u_i^2)\Big],\end{aligned} \tag{A.12}$$

where $\{u_i^1, u_i^2\}_{i=1}^N$ and $\{v_i^1, v_i^2\}_{i=1}^N$ denote real and generated samples respectively.

For both generator and D_u, we set the number of features to be 64. The output dimension of D_u is 128 and D_b is a 2-hidden-layer fully-connected network with residual connections [91] and LeakyReLU activation function.

CAT Note that the ideal pairwise discriminator for PairGAN should be symmetric, meaning that the order of the pair should not matter ($D(x, y) = D(y, x)$). From this observation, we parameterize the pairwise discriminator differently as $D(x, y) = D_b(D_u(x) + D_u(y))$ [269] where D_u is the same as the one we used in CIFAR-10, but D_b is now a binary network that takes in the addition of two D_u output and returns

[3]We omit $\log D(u_i^1, u_i^2)$ term from the estimator of the generator loss, since the generator does not receive gradient from this term as it involves only real samples.

a single scalar. With this setup, the discriminator is innately symmetric and the discriminator and generator's corresponding losses in this case will be:

$$\begin{aligned}\widetilde{\mathcal{L}}_D =& \frac{1}{N}\sum_{i=1}^{N}\Big[-\log D(u_i^1, u_i^2) - \log D(v_i^1, v_i^2) - 2\cdot\log(1-D(u_i^3, v_i^3))\Big]\\ \widetilde{\mathcal{L}}_G =& \frac{1}{N}\sum_{i=1}^{N}\Big[\log D(v_i^1, v_i^2) - 2\cdot\log D(u_i^3, v_i^3)\Big],\end{aligned} \tag{A.13}$$

However, with real world datasets (CAT in this case), we find the use of below techniques to be beneficial for training.

- **Averaging**: Inspired by averaging technique used in relativistic GANs [114], we introduce averaging-based version of pairwise discriminator. Given a single sample x and a mini-batch of N samples $Y = \{y_i\}_{i=1}^N$, the output of the discriminator is computed as

  $$\overline{D}(x, Y) = D_b\Big(D_u(x) + e\cdot\frac{1}{d\cdot N}\sum_{i=1}^{N}\sum_{j=1}^{d}[D_u(y_i)]_j\Big),$$

  where d denotes output dimension of $D_u(\cdot)$ and $e = [1, \ldots, 1]^T \in \mathbb{R}^d$.

  With the modified discriminator $\overline{D}$, we re-write the estimators (A.13) as

  $$\begin{aligned}\widetilde{\mathcal{L}}_D =& \frac{1}{N}\sum_{i=1}^{N}\Big[-\log\overline{D}(u_i^1, U^2) - \log\overline{D}(v_i^1, V^2) - 2\cdot\log(1-\overline{D}(u_i^3, V^3))\Big],\\ \widetilde{\mathcal{L}}_G =& \frac{1}{N}\sum_{i=1}^{N}\Big[\log\overline{D}(v_i^1, V^2) - 2\cdot\log\overline{D}(u_i^3, V^3)\Big],\end{aligned} \tag{A.14}$$

  where $U^j = \{u_i^j\}_{i=1}^N$, $V^j = \{v_i^j\}_{i=1}^N$, $j \in \{1, 2, 3\}$.

- **Annealing**: We start training with a different generator loss function for the generator and gradually anneal to the loss described above. The annealed loss function for the generator is given by:

  $$\mathcal{L}_G = \frac{1}{N}\sum_{i=1}^{N}\Big[\alpha\cdot\Big(-\log(1-\overline{D}(v_i^1, V^2))\Big)+(1-\alpha)\cdot\Big(\log\overline{D}(v_i^1, V^2)\Big)-2\cdot\log\overline{D}(u_i^3, V^3)\Big],$$

the annealing coefficient α changes from 1 to 0 as a function of the step counter $i = 1, 2, \ldots$

$$\alpha_i = \begin{cases} 1, & i < n_1 \\ 1 - \frac{i-n_1}{n_2}, & n_1 \leq i < n_1 + n_2 \\ 0, & i \geq n_1 + n_2 \end{cases}$$

For 64x64 and 128x128 resolutions, we set the number of features to be 64 and 32 for 256x256 resolution (the same as in the baselines provided by Jolicoeur-Martineau [114]) For all resolutions, the output dimension of D_u is 2 and D_b is a 1-hidden-layer fully-connected network with SELU activation function [125]: $D_b(x) = \texttt{FC}_{16\to1}(\texttt{SELU}(\texttt{FC}_{2\to16}(x)))$. Additionally, we use the annealing period of $n_1 + n_2 =$ 1000 steps with $n_1 = n_2 = 500$. Since there are only about 2000 training images for resolution 256x256, the mode collapse problem is severe with the vanilla versions of all models (our model and the baselines). Thus, for CAT 256$\times$256 we adopt PacGAN2 [147] architecture for the discriminator, which is the same modification done for all the corresponding baselines in Jolicoeur-Martineau [114].

For all real world datasets experiment, We train PairSGAN with Adam [121] using one step of discriminator per generator step. We use the same standard setting of hyperparameters (also the same in baseline models): learning rate 0.0002, $\beta_1 = 0.5$, $\beta_2 = 0.999$. We implement PairSGAN in PyTorch [189].

A.8.3 Examples

Figures A-2, A-3, and A-4 show samples generated by PairGAN trained on CAT dataset for resolutions 64x64, 128x128, and 256x256 respectively. While we resize the 128x128 and 256x256 samples in order to fit the figures in one page, we provide the original images in the code repository.

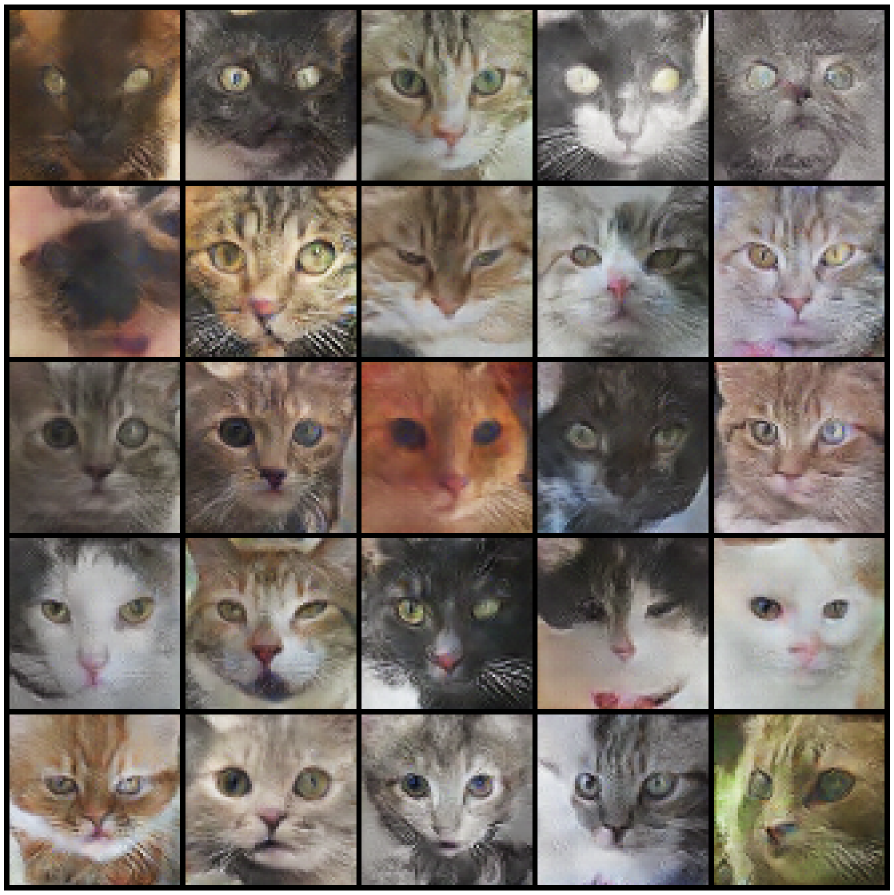


Figure A-2: Examples of 64×64 CAT images generated with PairGAN

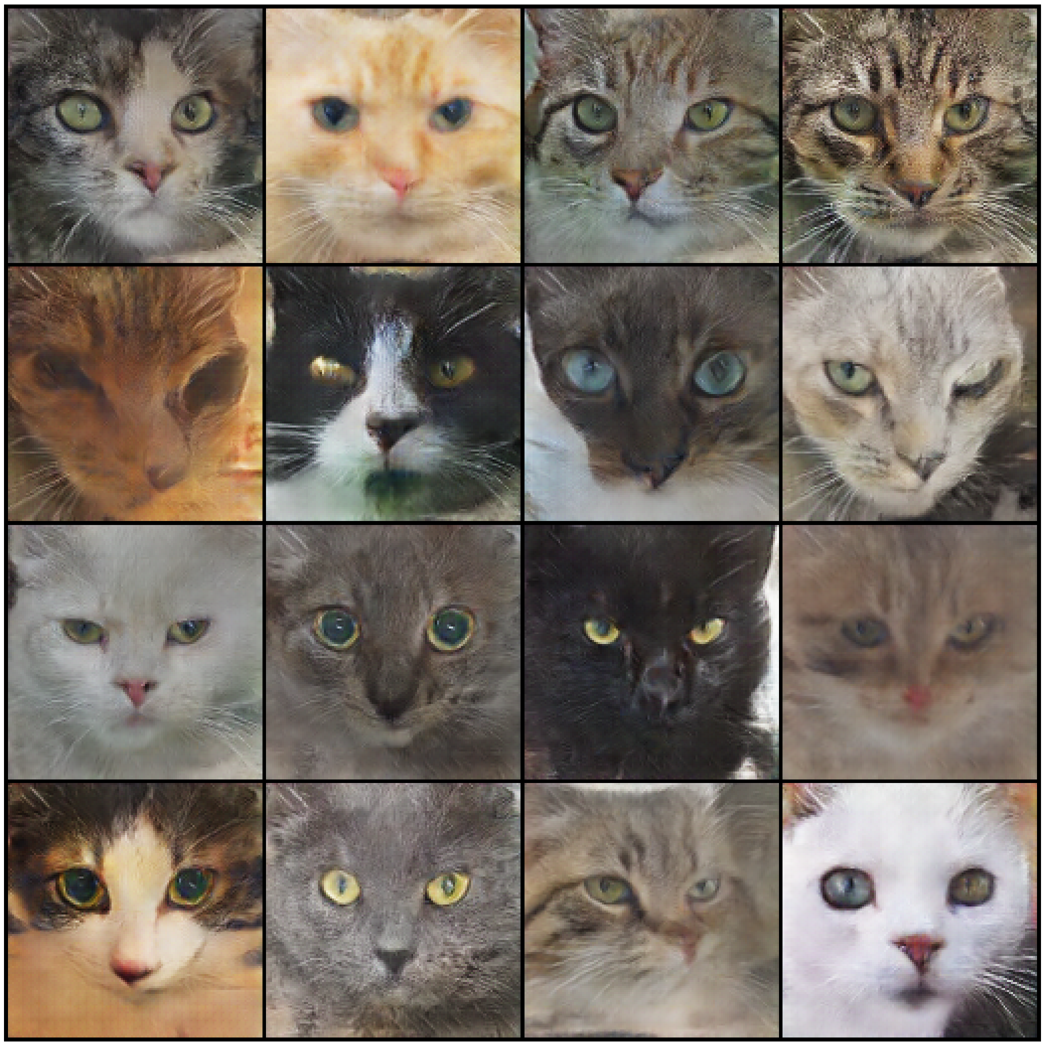


Figure A-3: Examples of 128 × 128 CAT images generated with PairGAN

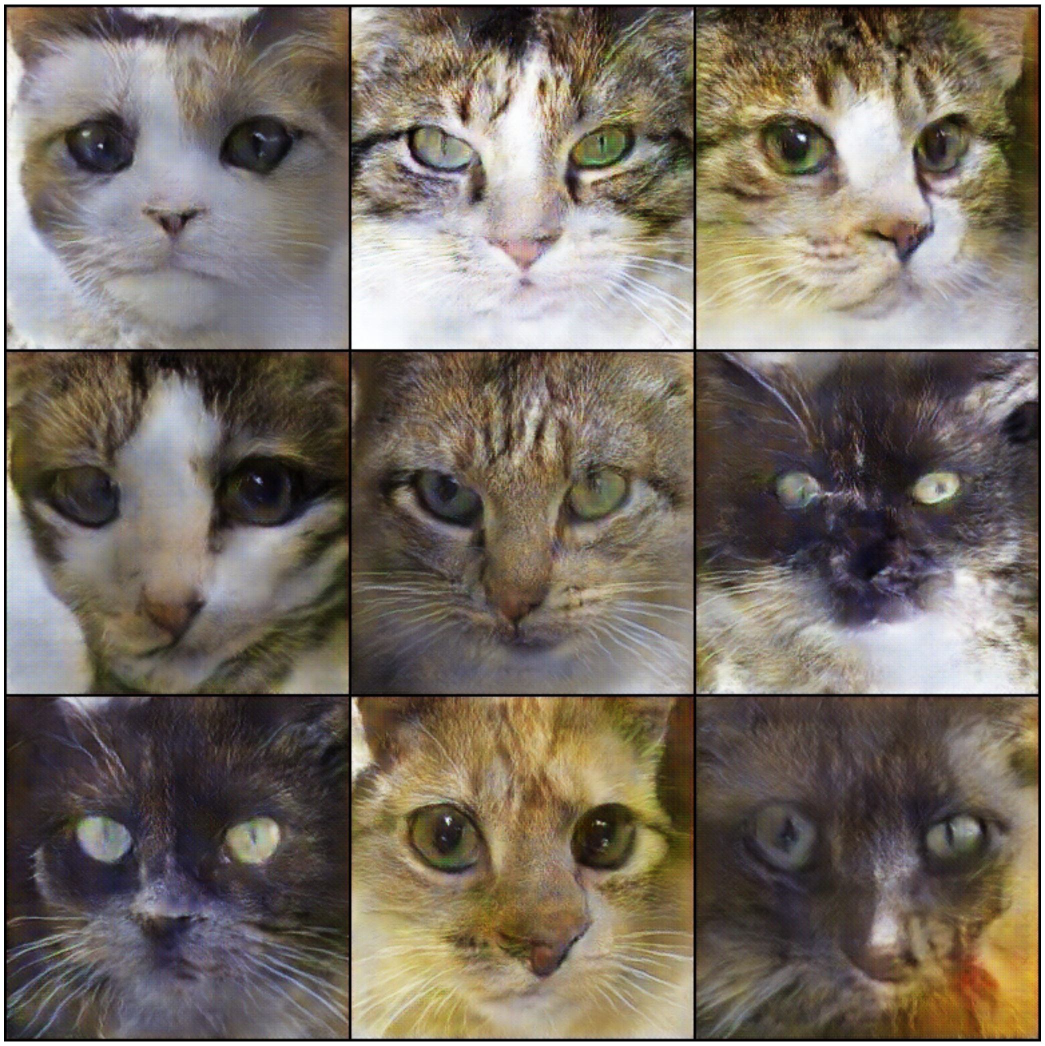


Figure A-4: Examples of 256×256 CAT images generated with PairGAN

Appendix B

Adversarial Support Alignment

B.1 Proofs of the Theoretical Results

B.1.1 Proof of Proposition 4.2.1

1) $\mathcal{D}_{\triangle}(p, q) \geq 0$ for all $p, q \in \mathcal{P}$:

Since $d(\cdot, \cdot) \geq 0$, for all p, q,

$$\mathrm{SD}(p, q) \coloneqq \mathbb{E}_{x \sim p}[d(x, \mathrm{supp}(q))] = \mathbb{E}_{x \sim p}\left[\inf_{y \in \mathrm{supp}(q)} d(x, y)\right] \geq 0, \tag{B.1}$$

which makes $\mathcal{D}_{\triangle}(p, q) = \mathrm{SD}(p, q) + \mathrm{SD}(q, p) \geq 0$.

---

2) $\mathcal{D}_{\triangle}(p, q) = 0$ if and only if $\mathrm{supp}(p) = \mathrm{supp}(q)$:

With statement 1, $\mathcal{D}_{\triangle}(p, q) = 0$ if and only if $\mathrm{SD}(p, q) = 0$ and $\mathrm{SD}(q, p) = 0$.

Then,

$$\mathrm{SD}(p, q) = 0 \implies \mathbb{E}_{x \sim p}[d(x, \mathrm{supp}(q))] = 0 \implies p\left(\{x | d(x, \mathrm{supp}(q)) > 0\}\right) = 0.$$

This is equivalent to

$$\forall x \in \mathrm{supp}(p),\ d(x, \mathrm{supp}(q)) = 0.$$

Thus, $\mathrm{supp}(p) \subseteq \mathrm{supp}(q)$, and similarly, $\mathrm{supp}(q) \subseteq \mathrm{supp}(p)$, which makes $\mathrm{supp}(p) = \mathrm{supp}(q)$.

B.1.2 Assumption and Proof of Theorem 4.2.2

Comments on Assumption (4.3)

Assumption (4.3) is not restrictive. Indeed, distributions satisfying Assumption (4.3) include:

- uniform $p(x) = U(x; [a, b])$;
- truncated normal;
- $p(x)$ of the form
$$p(x) = \begin{cases} \frac{1}{Z_p} e^{-E_p(x)}, & x \in \mathrm{supp}(p), \\ 0, & x \notin \mathrm{supp}(p), \end{cases}$$
with non-negative energy (unnormalized log-density) function $E_p : \mathcal{X} \to [0, \infty)$;
- mixture of any distributions satisfying Assumption (4.3), for instance the distributions shown in Figure 4-3 top-left are mixtures of truncated normal distributions on $[-2, 2]$.

Starting from arbitrary density $p_0(x)$ with bounded support we can derive a density $p(x)$ satisfying Assumption (4.3) via density clipping and re-normalization

$$p(x) \propto \mathrm{clip}\left(p_0(x), \left[\frac{1}{C'}, C'\right]\right),$$

for some $C' > 1$.

Proof of Theorem 4.2.2

First, we show that $\mathcal{D}_{\triangle}(p, q) = 0$ implies $\mathcal{D}_{\triangle}({f^*}_{\sharp}p, {f^*}_{\sharp}q) = 0$.

$\mathcal{D}_\triangle(p,q) = 0$ implies $\text{supp}(p) = \text{supp}(q)$. Then for any mapping $f : \mathcal{X} \to \mathbb{R}$, we have $\text{supp}(f_\sharp p) = \text{supp}(f_\sharp q)$, which implies $\text{supp}({f^*}_\sharp p) = \text{supp}({f^*}_\sharp q)$. Thus, $\mathcal{D}_\triangle({f^*}_\sharp p, {f^*}_\sharp q) = 0$.

Now, we prove that $\mathcal{D}_\triangle({f^*}_\sharp p, {f^*}_\sharp q) = 0$ implies $\mathcal{D}_\triangle(p,q) = 0$ by contradiction.

$\mathcal{D}_\triangle({f^*}_\sharp p, {f^*}_\sharp q) = 0$ implies the following:

$$\mathbb{E}_{t\sim {f^*}_\sharp p}\big[d(t, \text{supp}({f^*}_\sharp q)\big] = 0, \qquad \mathbb{E}_{t\sim {f^*}_\sharp q}\big[d(t, \text{supp}({f^*}_\sharp p)\big] = 0.$$

1) Suppose $\mathbb{E}_{x\sim p}[d(x, \text{supp}(q))] > 0$. This is only possible if $p(\{x \,|\, x \in \text{supp}(p) \setminus \text{supp}(q)\}) > 0$. Since $x \in \text{supp}(p) \setminus \text{supp}(q)$ implies $p(x) > 0, q(x) = 0$, and for any $x \in \text{supp}(p) \cup \text{supp}(q)$, $p(x) > 0, q(x) = 0$ if and only if $f^*(x) = \frac{p(x)}{p(x)+q(x)} = 1$, we have:

$$\mathbb{P}_{{f^*}_\sharp p}(\{1\}) = \mathbb{P}_p(\{x \,|\, x \in \text{supp}(p) \setminus \text{supp}(q)\}) > 0,$$

and therefore $1 \in \text{supp}({f^*}_\sharp p)$.

For a real number $\alpha : 0 < \alpha < \frac{1}{C^2+1}$, consider the probability of the event $(1-\alpha, 1] \subset [0,1]$ under distribution ${f^*}_\sharp q$:

$$\mathbb{P}_{{f^*}_\sharp q}((1-\alpha, 1]) = \mathbb{P}_q(\{x \,|\, f^*(x) \in (1-\alpha, 1]\}).$$

By assumption (4.3), $p(x) < C$ and $q(x) > 0$ implies $q(x) > \frac{1}{C}$, therefore for $x : q(x) > 0$ we have

$$f^*(x) = \frac{p(x)}{p(x)+q(x)} < \frac{p(x)}{p(x)+\frac{1}{C}} < \frac{C}{C+\frac{1}{C}} = 1 - \frac{1}{C^2+1} < 1-\alpha.$$

This means that $\mathbb{P}_{{f^*}_\sharp q}((1-\alpha, 1]) = 0$, i.e. $\text{supp}({f^*}_\sharp q) \cap (1-\alpha, 1] = \varnothing$.

To summarize, starting from the assumption that $\mathbb{E}_{x\sim p}[d(x, \text{supp}(q))] > 0$ we showed that

- $1 \in \text{supp}({f^*}_\sharp p)$, $\mathbb{P}_{{f^*}_\sharp p}(\{1\}) > 0$;
- $\text{supp}({f^*}_\sharp q) \cap (1-\alpha, 1] = \varnothing$.

Because $\mathcal{D}_\triangle(f^*{}_\sharp p, f^*{}_\sharp q) \geq \mathbb{E}_{t\sim f^*{}_\sharp p}\big[d(t, \mathrm{supp}(f^*{}_\sharp q))\big] \geq \mathbb{P}_{f^*{}_\sharp p}(\{1\}) \cdot d(1, \mathrm{supp}(f^*{}_\sharp q)) \geq \mathbb{P}_{f^*{}_\sharp p}(\{1\}) \cdot \alpha > 0$, which contradicts with the given $\mathcal{D}_\triangle(f^*{}_\sharp p, f^*{}_\sharp q) = 0$, we have $\mathbb{E}_{x\sim p}[d(x, \mathrm{supp}(q))] = 0$.

2) Similarly, it can be shown $\mathbb{E}_{x\sim q}[d(x, \mathrm{supp}(p))] = 0$.

Thus, $\mathcal{D}_\triangle(f^*{}_\sharp p, f^*{}_\sharp q) = 0$ implies $\mathcal{D}_\triangle(p, q) = 0$.

B.1.3 Proof of Proposition 4.2.4

Consider a 1-dimensional Euclidean space $\mathbb{R}$. Let $\mathrm{supp}(p) = [-\frac{1}{2}, \frac{1}{2}] \cup [1, 2]$ with $p([-\frac{1}{2}, \frac{1}{2}]) = \frac{3}{4}$ and $p([1, 2]) = \frac{1}{4}$. Let $\mathrm{supp}(q) = [-2, -1] \cup [-\frac{1}{2}, \frac{1}{2}] \cup [1, 2]$ with $q([-2, -1]) = \frac{1}{4}, q([-\frac{1}{2}, \frac{1}{2}]) = \frac{1}{4}$ and $q([1, 2]) = \frac{1}{2}$. The supports of p and q consist of disjoint closed intervals, and we assume uniform distribution within each of these intervals, i.e. p has density $p(x) = \frac{3}{4}, \forall x \in [-\frac{1}{2}, \frac{1}{2}]; p(x) = \frac{1}{4}, \forall x \in [1, 2]$ and q has densitiy $q(x) = \frac{1}{4}, \forall x \in [-2, -1]; q(x) = \frac{1}{4}, \forall x \in [-\frac{1}{2}, \frac{1}{2}]; q(x) = \frac{1}{2}, \forall x \in [1, 2]$. Clearly, $\mathrm{supp}(p) \neq \mathrm{supp}(q)$.

The optimal dual Wasserstein discriminator f_W^* is the maximizer of

$$\sup_{f:\mathrm{Lip}(f)\leq 1} \mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{y\sim q}[f(y)].$$

Thus, f_W^* is the maximizer of

$$\sup_{f:\mathrm{Lip}(f)\leq 1} \frac{1}{4}\left(3\int_{-\frac{1}{2}}^{\frac{1}{2}} f(x)dx + \int_1^2 f(x)dx - \int_{-2}^{-1} f(x)dx - \int_{-\frac{1}{2}}^{\frac{1}{2}} f(x)dx - 2\int_1^2 f(x)dx\right),$$

which simplifies to

$$\sup_{f:\mathrm{Lip}(f)\leq 1} \frac{1}{4}\left(-\int_{-2}^{-1} f(x)dx + 2\int_{-\frac{1}{2}}^{\frac{1}{2}} f(x)dx - \int_1^2 f(x)dx\right).$$

Since the optimization objective and the constraint are invariant to replacing the function $f(x)$ with its symmetric reflection $g(x) = f(-x)$, if f' is a optimal solution, then there exists a symmetric maximizer $f_W^*(x) = \frac{1}{2}f'(x) + \frac{1}{2}f'(-x)$, since $f_W^*(x) = f_W^*(-x)$ and $\mathrm{Lip}(f_W^*) \leq \mathrm{Lip}(f') \leq 1$. Thus, $\mathrm{supp}(f^\star_{W\sharp} p) = \mathrm{supp}(f^\star_{W\sharp} q)$ as

$f^*_W(x) = f^*_W(-x)$ for $x \in [1, 2]$.

Note that one can easily "extend" the above proof to discrete distributions, by replacing the disjoint segments $[-2, -1], [-\frac{1}{2}, \frac{1}{2}], [1, 2]$ with points $\{-1\}, \{0\}, \{1\}$.

B.1.4 Proof of Proposition 4.4.1

We assume that supports $\text{supp}(p)$ and $\text{supp}(q)$ are compact, p and q have continuous densities on $\mathcal{X} = \mathbb{R}^n$ and $d(x, y)$ is a continuous metric function.

From (4.9), we have

$$\mathcal{D}^\beta_W(p, q) = \inf_{\gamma \in \Gamma_\beta(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[d(x, y)],$$

where $\Gamma_\beta(p, q)$ is the set of all mesaures γ on $\mathcal{X} \times \mathcal{X}$ that satisfy

$$\int \gamma(x, y)\, dy = p(x), \quad \forall\, x, \tag{B.2}$$

$$\int \gamma(x, y)\, dx \le (1 + \beta)q(y), \quad \forall\, y. \tag{B.3}$$

Below we show that $\lim_{\beta\to\infty} \mathcal{D}^\beta_W(p, q) = \mathbb{E}_{x\sim p}[d(x, \text{supp}(q))] = SD(p, q)$ (B.1).

First observe that for any $x \in \text{supp}(p), y \in \text{supp}(q)$ we have $d(x, y) \ge d(x, \text{supp}(q)) = \inf_{y\in\text{supp}(q)} d(x, y)$. Therefore,

$$\begin{aligned} \mathcal{D}^\beta_W(p, q) &= \inf_{\gamma \in \Gamma_\beta(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[d(x, y)] && \text{(B.4)} \\ &\ge \inf_{\gamma \in \Gamma_\beta(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[d(x, \text{supp}(q))] = \mathbb{E}_{x\sim p}[d(x, \text{supp}(q))] = SD(p, q). && \text{(B.5)} \end{aligned}$$

Let $F(x)$ denote the function $F(x) = d(x, \text{supp}(q)) = \inf_{y\in\text{supp}(q)} d(x, y)$. Let $B_r(x)$ denote the ball centered at x with radius r: $B_r(x) = \{y \mid d(x, y) \le r\}$. For $\beta \ge 0$ let $s_\beta(x)$ be a function that for a given point x is defined through the constraint

$$s_\beta(x) : q(B_{F(x)+s_\beta(x)}(x)) = \frac{1}{1 + \beta}. \tag{B.6}$$

Since $\text{supp}(q)$ is bounded, by construction $s_\beta(x)$ is bounded for all x, $s_{\beta'}(x) < s_\beta(x)$

for $\beta' > \beta$, and $\lim_{\beta\to\infty} s_\beta(x) = 0$.

For $\beta \geq 0$ we construct distributions γ_β on $\mathcal{X} \times \mathcal{X}$ as

$$\gamma_\beta(x, y) = (1+\beta)p(x)q(y)\, I[y \in B_{F(x)+s_\beta(x)}(x)], \tag{B.7}$$

where $I[\cdot]$ is the indicator function.

For $\gamma_\beta(x, y)$ we have

$$\int \gamma_\beta(x, y)\, dy = (1+\beta)p(x)q(B_{F(x)+s_\beta(x)}(x)) = p(x), \tag{B.8}$$

$$\int \gamma_n(x, y)\, dx = (1+\beta)q(y) \int p(x) I[y \in B_{F(x)+s_\beta(x)}(x)]\, dx \leq (1+\beta)\, q(y). \tag{B.9}$$

Thus, $\gamma_\beta \in \Gamma_\beta(p, q)$. Moreover, for γ_β we have

$$\begin{aligned}
\mathbb{E}_{x,y\sim\gamma_\beta}[d(x, y)] &= \int (1+\beta)p(x)q(y)\, I[y \in B_{F(x)+s_\beta(x)}(x)] d(x, y)\, dx\, dy \\
&\leq \int (1+\beta)p(x)q(y) I[y \in B_{F(x)+s_\beta(x)}(x)](F(x) + s_\beta(x))\, dx\, dy \\
&= \int (1+\beta)p(x)(F(x) + s_\beta(x)) \left(\int q(y) I[y \in B_{F(x)+s_\beta(x)}(x)]\, dy \right) dx \\
&= \int p(x)(F(x) + s_\beta(x))\, dx \\
&= \mathbb{E}_{x\sim p}[d(x, \operatorname{supp} q)] + \int p(x) s_\beta(x)\, dx \\
&= SD(p, q) + \int p(x) s_\beta(x)\, dx.
\end{aligned} \tag{B.10}$$

Combining (B.5) and (B.10) we have

$$SD(p, q) \leq \lim_{\beta\to\infty} D_W^\beta(p, q) \tag{B.11}$$

$$\lim_{\beta\to\infty} D_W^\beta(p, q) \leq \lim_{\beta\to\infty} \mathbb{E}_{x,y\sim\gamma_\beta}[d(x, y)] \leq SD(p, q) + \lim_{\beta\to\infty} \int p(x) s_\beta(x)\, dx. \tag{B.12}$$

We can see that $\lim_{\beta\to\infty} \int p(x) s_\beta(x)\, dx = 0$. Indeed, since $\lim_{\beta\to\infty} s_\beta(x) = 0\ \forall\, x$,

$s_\beta(x) \to 0$ by measure. Then, for any $\varepsilon > 0$

$$\int p(x)s_\beta(x)\,dx = \int I[s_\beta(x) < \varepsilon]p(x)s_\beta(x)\,dx + \int I[s_\beta(x) \geq \varepsilon]p(x)s_\beta(x)\,dx \quad \text{(B.13)}$$

$$\leq \varepsilon \int p(x)\,dx + \left(\sup_{x\in\text{supp}(p)} s_\beta(x)\right) p(\{x : s_\beta(x) > \varepsilon\}) \quad \text{(B.14)}$$

$$\leq \varepsilon + \left(\sup_{x\in\text{supp}(p)} s_\beta(x)\right) p(\{x : s_\beta(x) > \varepsilon\}), \quad \text{(B.15)}$$

where the second term can be made smaller than ε by choosing large enough β since $\sup_{x\in\text{supp}(p)} s_\beta(x)$ is finite ($s_\beta(x)$ is bounded) and the measure of the set $p(\{x : s_\beta(x) > \varepsilon\})$ goes to 0 as $\beta \to \infty$ ($s_\beta(x)$ converges to zero by measure).

Now, we have shown that $SD(p, q) = \lim_{\beta\to\infty} D_W^\beta(p, q)$.

Then for $\mathcal{D}_W^{\infty,\infty}(p, q) := \lim_{\beta_1,\beta_2\to\infty} \mathcal{D}_W^{\beta_1,\beta_2}(p, q)$, we have

$$\mathcal{D}_W^{\infty,\infty}(p, q) = \mathcal{D}_W^{\infty}(p, q) + \mathcal{D}_W^{\infty}(q, p) = \text{SD}(p, q) + \text{SD}(q, p) = \mathcal{D}_\triangle(p, q),$$

which proves the proposition.

B.1.5 Proof of Proposition 4.4.2

1. $\mathcal{D}_W(p, q) = 0$ implies $p = q$, which is equivalent to

   $$\frac{p(x)}{q(x)} = 1, \qquad \forall x \in \text{supp}(p) \cup \text{supp}(q).$$

   Then clearly, for all finite $\beta_1, \beta_2 > 0$ it satisfies

   $$\frac{1}{1+\beta_2} \leq \frac{p(x)}{q(x)} \leq 1 + \beta_1, \qquad \forall x \in \text{supp}(p) \cup \text{supp}(q). \quad \text{(B.16)}$$

   Thus, $\mathcal{D}_W^{\beta_1,\beta_2}(p, q) = 0$ for all finite $\beta_1, \beta_2 > 0$.

2. $\mathcal{D}_W^{\beta_1,\beta_2}(p, q) = 0$ for some finite $\beta_1, \beta_2 > 0$ means that (B.16) is satisfied. This implies that $\forall x \in \text{supp}(p)$, $x \in \text{supp}(q)$ and $\forall x \in \text{supp}(q)$, $x \in \text{supp}(p)$, which makes $\text{supp}(p) = \text{supp}(q)$. Thus, $\mathcal{D}_\triangle(p, q) = 0$.

3. The converse of statements 1 and 2 are false:

   (a) For all finite $\beta_1, \beta_2 > 0$, let $\mathrm{supp}(p) = \mathrm{supp}(q) = \{x_1, x_2\}$. Let $p(x_1) = p(x_2) = 1/2$ and $q(x_1) = (1+\beta')/2$ and $q(x_2) = (1-\beta')/2$ where

$$\beta' = \min\left(\beta_2, 1 - \frac{1}{1+\beta_1}\right).$$

   Then, it can be easily checked that (B.16) is satisfied, which makes $\mathcal{D}_W^{\beta_1,\beta_2}(p,q) = 0$. However, since $\beta' \neq 0$, $p \neq q$ and thus $\mathcal{D}_W(p,q) \neq 0$.

   (b) Similar to (a), let $\mathrm{supp}(p) = \mathrm{supp}(q) = \{x_1, x_2\}$. Let $p(x_1) = q(x_2) = \varepsilon$ and $p(x_2) = q(x_1) = 1 - \varepsilon$ for some $\varepsilon > 0$. Since $\mathrm{supp}(p) = \mathrm{supp}(q)$, $\mathcal{D}_\triangle(p,q) = 0$. However,

$$\lim_{\varepsilon \downarrow 0} \frac{p(x_1)}{q(x_1)} = \lim_{\varepsilon \downarrow 0} \frac{\varepsilon}{1-\varepsilon} = 0,$$

   and, thus, for any finite $\beta_2 > 0$ we can choose $\varepsilon > 0$ such that

$$\frac{p(x_1)}{q(x_1)} = \frac{\varepsilon}{1-\varepsilon} < \frac{1}{1+\beta_2}.$$

   Therefore, (B.16) is not satisfied and $\mathcal{D}_W^{\beta_1,\beta_2}(p,q) \neq 0$.

B.1.6 Proof of Proposition 4.4.4

For any point $t \in \mathrm{supp}(f^*{}_\sharp p) \cup \mathrm{supp}(f^*{}_\sharp q)$, the values of the densities

$$[f^*{}_\sharp p](t) = \lim_{\varepsilon \downarrow 0} \frac{\mathbb{P}_p\left(\{x \mid t - \varepsilon < f^*(x) < t + \varepsilon\}\right)}{2\varepsilon} = \lim_{\varepsilon \downarrow 0} \frac{\int_{\{x \mid t-\varepsilon < f^*(x) < t+\varepsilon\}} p(x)\, dx}{2\varepsilon},$$
$$[f^*{}_\sharp q](t) = \lim_{\varepsilon \downarrow 0} \frac{\mathbb{P}_q\left(\{x \mid t - \varepsilon < f^*(x) < t + \varepsilon\}\right)}{2\varepsilon} = \lim_{\varepsilon \downarrow 0} \frac{\int_{\{x \mid t-\varepsilon < f^*(x) < t+\varepsilon\}} q(x)\, dx}{2\varepsilon}.$$

Note that for all $x : t - \varepsilon < f^*(x) < t + \varepsilon$ we have

$$t - \varepsilon < \frac{p(x)}{p(x) + q(x)} < t + \varepsilon,$$

which implies

$$(t-\varepsilon)(p(x)+q(x)) < p(x) < (t+\varepsilon)(p(x)+q(x)).$$

Since these inequalities hold for all $x : t-\varepsilon < f^*(x) < t+\varepsilon$, the similar relationship holds for the integrals:

$$\begin{aligned}(t-\varepsilon)\int_{\{x|t-\varepsilon<f^*(x)<t+\varepsilon\}}(p(x)+q(x))\,dx \\ < \int_{\{x|t-\varepsilon<f^*(x)<t+\varepsilon\}} p(x)\,dx < \\ (t+\varepsilon)\int_{\{x|t-\varepsilon<f^*(x)<t+\varepsilon\}}(p(x)+q(x))\,dx.\end{aligned}$$

The ratio $[{f^*}_\sharp p](t)/([{f^*}_\sharp p](t)+[{f^*}_\sharp q](t))$ can be expressed as

$$\frac{[{f^*}_\sharp p](t)}{[{f^*}_\sharp p](t)+[{f^*}_\sharp q](t)} = \lim_{\varepsilon\downarrow 0}\frac{\int_{\{x|t-\varepsilon<f^*(x)<t+\varepsilon\}} p(x)\,dx}{\int_{\{x|t-\varepsilon<f^*(x)<t+\varepsilon\}}(p(x)+q(x))\,dx}.$$

Using the inequality above we observe that

$$t-\varepsilon < \frac{\int_{\{x|t-\varepsilon<f^*(x)<t+\varepsilon\}} p(x)\,dx}{\int_{\{x|t-\varepsilon<f^*(x)<t+\varepsilon\}}(p(x)+q(x))\,dx} < t+\varepsilon,$$

for all $\varepsilon > 0$, and taking the limit $\varepsilon \downarrow 0$ we obtain

$$\frac{[{f^*}_\sharp p](t)}{[{f^*}_\sharp p](t)+[{f^*}_\sharp q](t)} = t.$$

B.1.7 Proof of Proposition 4.4.3

Proof of Proposition 4.4.3, statement #1.

$\Longrightarrow$: If $\mathcal{D}_W(p,q)=0$ then $p=q$, then ${f^*}_\sharp p = {f^*}_\sharp q$. Thus, $\mathcal{D}_W({f^*}_\sharp p, {f^*}_\sharp q)=0$.

$\Longleftarrow$: If $\mathcal{D}_W({f^*}_\sharp p, {f^*}_\sharp q)=0$, then ${f^*}_\sharp p = {f^*}_\sharp q$.

Consider probability of event $\left\{t \,\middle|\, t > \frac{1}{2}\right\}$ under distribution $f^*{}_\sharp p$.

$$\mathbb{P}_{f^*{}_\sharp p}\left(\left\{t \,\middle|\, t > \frac{1}{2}\right\}\right) = \int \mathbb{I}\left[f^*(x) > \frac{1}{2}\right] p(x)\,dx,$$

where $\mathbb{I}[\cdot]$ is the indicator function ($\mathbb{I}[c]$ equal to 1 when the condition c is satisfied, and equal to 0 otherwise). For all $x : p(x) > 0$, we have that $f^*(x) = \frac{p(x)}{p(x)+q(x)}$ and $p(x) + q(x) > 0$. Therefore, the expression above can be re-written as

$$\mathbb{P}_{f^*{}_\sharp p}\left(\left\{t \,\middle|\, t > \frac{1}{2}\right\}\right) = \int \mathbb{I}\left[\frac{p(x)}{p(x)+q(x)} > \frac{1}{2}\right] p(x)\,dx = \int \mathbb{I}[p(x) - q(x) > 0] p(x)\,dx.$$

Similarly, the probability of event $\left\{t \,\middle|\, t > \frac{1}{2}\right\}$ under distribution $f^*{}_\sharp q$ is

$$\mathbb{P}_{f^*{}_\sharp q}\left(\left\{t \,\middle|\, t > \frac{1}{2}\right\}\right) = \int \mathbb{I}[p(x) - q(x) > 0] q(x)\,dx.$$

$f^*{}_\sharp p = f^*{}_\sharp q$ implies that

$$\mathbb{P}_{f^*{}_\sharp p}\left(\left\{t \,\middle|\, t > \frac{1}{2}\right\}\right) = \mathbb{P}_{f^*{}_\sharp q}\left(\left\{t \,\middle|\, t > \frac{1}{2}\right\}\right),$$

or equivalently

$$\int \mathbb{I}[p(x) - q(x) > 0](p(x) - q(x))\,dx = 0.$$

Note, that the function $\mathbb{I}[p(x) - q(x) > 0](p(x) - q(x))$ is non-negative for any x. This means that the integral can be zero only if the function is zero everywhere implying that for any x either $\mathbb{I}[p(x) - q(x) > 0] = 0$ or $p(x) - q(x) = 0$. In other words,

$$p(x) \leq q(x), \quad \forall\, x.$$

Using the fact the both densities $p(x)$ and $q(x)$ must sum up to 1, we conclude that $p = q$ and $\mathcal{D}_W(p, q) = 0$.

Proof of Proposition 4.4.3, statement #2.

Note that by (4.2) and (4.11), we have

$$f^*(x) = \frac{p(x)}{p(x)+q(x)} = t = \frac{[f^*{}_\sharp p](t)}{[f^*{}_\sharp p](t) + [f^*{}_\sharp q](t)}, \qquad \forall x \in \mathrm{supp}(p) \cup \mathrm{supp}(q).$$

$\Longrightarrow$: Suppose $\mathcal{D}_W^{\beta_1,\beta_2}(p,q) = 0$. Then $\mathrm{supp}(p) = \mathrm{supp}(q) = S$ (by Proposition 4.4.2) and $\mathrm{supp}(f^*{}_\sharp p) = \mathrm{supp}(f^*{}_\sharp q) = T$ by (Theorem 4.2.2). Moreover, $\mathcal{D}_W^{\beta_1,\beta_2}(p,q) = 0$ implies

$$\frac{1}{1+\beta_2} \le \frac{p(x)}{q(x)} \le 1+\beta_1, \qquad \forall x \in S.$$

Since

$$f^*(x) = \frac{p(x)}{p(x)+q(x)} = \frac{\frac{p(x)}{q(x)}}{1+\frac{p(x)}{q(x)}}, \qquad \forall x \in S,$$

the inequalities above are equivalent to

$$\frac{1}{2+\beta_2} \le f^*(x) \le \frac{1+\beta_1}{2+\beta_1}, \qquad \forall x \in S.$$

Combined with Proposition 4.4.4, the above implies that

$$\frac{1}{2+\beta_2} \le \frac{[f^*{}_\sharp p](t)}{[f^*{}_\sharp p](t) + [f^*{}_\sharp q](t)} \le \frac{1+\beta_1}{2+\beta_1}, \qquad \forall t \in T,$$

or equivalently

$$\frac{1}{1+\beta_2} \le \frac{[f^*{}_\sharp p](t)}{[f^*{}_\sharp q](t)} \le 1+\beta_1, \qquad \forall t \in T.$$

Therefore, $\mathcal{D}_W^{\beta_1,\beta_2}(f^*{}_\sharp p, f^*{}_\sharp q) = 0$.

$\Longleftarrow$: similarly, when $\mathcal{D}_W^{\beta_1,\beta_2}(f^*{}_\sharp p, f^*{}_\sharp q) = 0$,

$$\mathrm{supp}(f^*{}_\sharp p) = \mathrm{supp}(f^*{}_\sharp q) = T \qquad \Longrightarrow \qquad \mathrm{supp}(p) = \mathrm{supp}(q) = S.$$

Moreover,

$$\frac{1}{1+\beta_2} \leq \frac{[f^*{}_\sharp p](t)}{[f^*{}_\sharp q](t)} \leq 1+\beta_1, \qquad \forall t \in T,$$

$$\Downarrow$$

$$\frac{1}{2+\beta_2} \leq \frac{[f^*{}_\sharp p](t)}{[f^*{}_\sharp p](t) + [f^*{}_\sharp q](t)} \leq \frac{1+\beta_1}{2+\beta_2}, \qquad \forall t \in T$$

$$\Downarrow$$

$$\frac{1}{2+\beta_2} \leq f^*(x) \leq \frac{1+\beta_1}{2+\beta_1}, \qquad \forall x \in S,$$

$$\Downarrow$$

$$\frac{1}{1+\beta_2} \leq \frac{p(x)}{q(x)} \leq 1+\beta_1, \qquad \forall x \in S.$$

Therefore, $\mathcal{D}_W^{\beta_1,\beta_2}(p,q) = 0$.

B.2 Discussion of "Soft" and "Hard" Assignments with 1D Discrete Distributions

In Section 4.4.2 we considered the "soft-assignment" relaxed OT problem (4.13) and claimed that for integer β, the set of minimizers of (4.13) must contain a "hard-assignment" transportation plan, meaning $\gamma_{ij} \in \{0,1\}, \forall i,j$. Below we justify this claim.

Note that for $\beta = 0$ the OT problem (4.13) is the standard OT problem for Wasserstein-1 distance (4.12), since the inequality constraints $\sum_{i=1}^{m} \gamma_{ij} \leq 1$, $\forall j$ can only be satisfied as equalities. For this problem, it is known (e.g. see [197] Proposition 2.1) that the set of optimal "soft-assignment" contains a "hard-assignment" represented by a normalized permutation matrix. This fact can be proven using the Birkhoff–von Neumann theorem. The Birkhoff–von Neumann theorem states that the set of doubly stochastic matrices

$$P \in \mathbb{R}^{n\times n}: \qquad P_{ij} \geq 0, \forall\, i, j, \qquad \sum_{j=1}^{n} P_{ij} = 1, \forall\, i, \qquad \sum_{i=1}^{n} P_{ij} = 1, \forall\, j$$

is exactly the set of all finite convex combinations of permutation matrices. In the context of the linear program (4.13) with $\beta = 0$, the Birkhoff–von Neumann theorem means that all extreme points of the polyhedron $\Gamma_\beta(o^p, o^q)$ are hard-assignment matrices. Therefore, by the fundamental theorem of linear programming [20], the minimum of the objective is reached at a "hard-assignment" matrix.

We argue that a similar result holds for the case of integer $\beta > 0$. In this case, the matrices in $\Gamma_\beta(o^p, o^q)$ can not be associated with the doubly stochastic matrices, since constraints on of the marginals of γ are relaxed to inequality constraints. Because of that, the Birkhoff–von Neumann theorem can not be applied. However, Budish et al. [34] provide a generalization of the Birkhoff–von Neumann theorem (Theorem 1 in [34]) which applies to the cases where the equality constraints are replaced with integer-valued inequality constraints (recall that we consider integer β). Using this generalized result, our claim can be proven by performing the following steps.

Clearly, the polyhedron $\Gamma_\beta(o^p, o^q)$ contains all "hard-assignment" matrices and all their finite convex combinations. The result proven in [34] implies that each element of $\Gamma_\beta(o^p, o^q)$ can be represented as a finite convex combination of "hard-assignment" matrices. Thus, the polyhedron $\Gamma_\beta(o^p, o^q)$ is exactly the set of all finite convex combinations of "hard-assignment" matrices and all extreme points of the polyhedron are "hard-assignment" matrices. Finally, by analogy with the case of $\beta = 0$, we invoke the fundamental theorem of the linear programming and conclude that the minimum of the objective (4.13) is reached at γ corresponding to a "hard-assignment" matrix.

B.3 Experiment Details

B.3.1 USPS to MNIST experiment specifications

We use USPS [104] and MNIST [140] datasets for this adaptation problem.

Following Tachet des Combes et al. [238] we use LeNet-like [140] architecture for the feature extractor with the 500-dimensional feature representation. The classifier consists of a single linear layer. The discriminator is implemented by a 3-layer MLP with 512 hidden units and leaky-ReLU activation.

We train all methods for 65 000 steps with batch size 64. We train the feature extractor, the classifier, and the discriminator with SGD (learning rate 0.02, momentum 0.9, weight decay $5 \cdot 10^{-4}$). We perform a single discriminator update per 1 update of the feature extractor and the classifier. After the first 30 000 steps we linearly anneal the feature extractor's and classifier's learning rates for 30 000 steps to the final value $2 \cdot 10^{-5}$.

The feature extractor's loss is given by a weighted combination of the cross-entropy classification loss on the labeled source example and a domain alignment loss computed from the discriminator's signal (recall that different method use different forms of the alignment loss). The weight for the classification term is constant and set to $\lambda_{\text{cls}} = 1$. We introduce schedule for the alignment weight λ_{align}. For all alignment methods we linearly increase λ_{align} from 0 to 1.0 during the first 10000 steps.

For ASA we use history buffers of size 1000.

B.3.2 STL to CIFAR experiment specifications

We use STL [46] and CIFAR-10 [130] for this adaptation task. STL and CIFAR-10 are both 10-class classification problems. There are 9 common classes between the two datasets. Following Shu et al. [225] we create a 9-class classification problem by selecting the subsets of examples of the 9 common classes.

For the feature extractor, we adapt the deep CNN architecture of Shu et al. [225]. The feature representation is a 192-dimensional vector. The classifier consists of a single linear layer. The discriminator is implemented by a 3-layer MLP with 512 hidden units and leaky-ReLU activation.

We train all methods for 40 000 steps with batch size 64. We train the feature extractor, the classifier, and the discriminator with ADAM [120] (learning rate 0.001, $\beta_1 = 0.5$, $\beta_2 = 0.999$, no weight decay). We perform a single discriminator update per

1 update of the feature extractor and the classifier.

The weight for the classification loss term is constant and set to $\lambda_{\text{cls}} = 1$. For all alignment methods we use constant alignment weight $\lambda_{\text{align}} = 0.1$.

For ASA we use history buffers of size 1000.

Conditional entropy loss. Following [225] we use auxiliary conditional entropy loss on target examples for domain adaptation methods. For a classifier $C^\phi : \mathcal{Z} \to \mathcal{Y}$ and a feature extractor $F^\theta : \mathcal{X} \to \mathcal{Z}$ where classifier outputs the distribution over class labels $\{1, \dots, K\}$

$$C^\phi(z) \in \mathbb{R}^K : \quad \left[C^\phi(z)\right]_k \geq 0, \quad \sum_{k=1}^{K} \left[C^\phi(z)\right]_k = 1,$$

the conditional entropy loss on target examples $\{x_i^q\}_{i=1}^{N_q}$ is given by

$$\mathcal{L}_{\text{ent}} = \lambda_{\text{ent}} \cdot \frac{1}{N^q} \sum_{i=1}^{N_q} \left(- \sum_{k=1}^{K} \left[C^\phi(F^\theta(x_i^q))\right]_k \log \left[C^\phi(F^\theta(x_i^q))\right]_k \right). \tag{B.17}$$

λ_{ent} is the weight of the conditional entropy loss in the total training objective. This loss acts as an additional regularization of the embeddings of the unlabeled target examples: minimization of the conditional entropy pushes target embeddings away from the classifier's decision boundary.

For all domain adapation methods we use the conditional entropy loss (B.17) on target examples with the weight $\lambda_{\text{ent}} = 0.1$.

B.3.3 VisDA-17 experiment specifications

We use train and validation sets of the VisDA-17 challenge [193].

For the feature extractor we use ResNet-50 [92] architecture with modified output size of the final linear layer. The feature representation is 256-dimensional vector. We use the weights from pre-trained ResNet-50 model (`torchvision` model hub) for all layers except the final linear layer. The classifier consists of a single linear layer. The discriminator is implemented by a 3-layer MLP with 1024 hidden units and leaky-ReLU activation.

We train all methods for 50 000 steps with batch size 36. We train the feature extractor, the classifier, and the discriminator with SGD. For the feature extractor we use learning rate 0.001, momentum 0.9, weight decay 0.001. For the classifier we use learning rate 0.01, momentum 0.9, weight decay 0.001. For the discriminator we use learning rate 0.005, momentum 0.9, weight decay 0.001. We perform a single discriminator update per 1 update of the feature extractor and the classifier. We linearly anneal the feature extractor's and classifier's learning rate throughout the training (50 000) steps. By the end of the training the learning rates of the feature extractor and the classifier are decreased by a factor of 0.05.

The weight for the classification term is constant and set to $\lambda_{\text{cls}} = 1$. We introduce schedule for the alignment weight λ_{align}. For all alignment methods we linearly increase λ_{align} from 0 to 0.1 during the first 10000 steps. For all methods we use auxiliary conditional entropy loss on target examples with the weight $\lambda_{\text{ent}} = 0.1$.

For ASA we use history buffers of size 1000.

Appendix C

Compositional Sculpting of Iterative Generative Processes

C.1 Classifier Guidance for Parameterized Operations

This section covers the details of classifier guidance and classifier training for the parameterized operations (Section 5.4.1).

While in the probabilistic model (5.7) all observations y_i are exchangeable, in the parameterized model (5.12) y_1 and y_2 are not symmetric. This difference requires changes in the classifier training algorithm for the parameterized operations.

We develop the method for the parameterized operations based on two observations:

- y_1 appears in (5.12) in the same way as in (5.16)-(5.18);
- the likelihood $\widetilde{p}(y_2|x;\alpha)$ of y_2 given x can be expressed as the function of $\widetilde{p}(y_1|x)$ and α:

$$\begin{aligned}\widetilde{p}(y_2\!=\!1|x;\alpha) &= \frac{\alpha p_1(x)}{\alpha p_1(x)+(1-\alpha)p_2(x)} = \frac{\alpha\frac{p_1(x)}{p_1(x)+p_2(x)}}{\alpha\frac{p_1(x)}{p_1(x)+p_2(x)}+(1-\alpha)\frac{p_2(x)}{p_1(x)+p_2(x)}}\\ &= \frac{\alpha\widetilde{p}(y_1\!=\!1|x)}{\alpha\widetilde{p}(y_1\!=\!1|x)+(1-\alpha)\widetilde{p}(y_1\!=\!2|x)},\end{aligned} \tag{C.1a}$$

$$\widetilde{p}(y_2\!=\!2|x;\alpha) = \frac{(1-\alpha)p_2(x)}{\alpha p_1(x) + (1-\alpha)p_2(x)} = \frac{(1-\alpha)\frac{p_2(x)}{p_1(x)+p_2(x)}}{\alpha\frac{p_1(x)}{p_1(x)+p_2(x)} + (1-\alpha)\frac{p_2(x)}{p_1(x)+p_2(x)}}$$
$$= \frac{(1-\alpha)\widetilde{p}(y_1\!=\!2|x)}{\alpha\widetilde{p}(y_1\!=\!1|x) + (1-\alpha)\widetilde{p}(y_1\!=\!2|x)}. \tag{C.1b}$$

These two observations combined suggest the training procedure where 1) the terminal state classifier is trained to approximate $\widetilde{p}(y_1\!=\!i|x)$ in the same way as in Section 5.5.2; 2) the probability estimates $w_i(\widehat{x}, \widehat{\alpha}; \phi) \approx \widetilde{p}(y_2 = i; \widehat{\alpha})$ are expressed through the learned terminal state classifier $\widetilde{p}(y_1\!=\!i|x)$ via (C.1). Below we provide details of this procedure for the case of GFlowNet composition.

Learning the terminal state classifier. The marginal y_1 classifier $\widetilde{Q}_\phi(y_1|x)$ is learned by minimizing the cross-entropy loss

$$\mathcal{L}_T(\phi) = \mathop{\mathbb{E}}_{(\widehat{x},\widehat{y}_1)\sim\widetilde{p}(x,y_1)}\left[-\log\widetilde{Q}_\phi(y_1\!=\!\widehat{y}_1|x=\widehat{x})\right]. \tag{C.2}$$

Then, the joint classifier $\widetilde{Q}_\phi(y_1, y_2|x;\alpha)$ is constructed as

$$\widetilde{Q}_\phi(y_1, y_2|x;\alpha) = \widetilde{Q}_\phi(y_1|x)\widetilde{Q}_\phi(y_2|x;\alpha), \tag{C.3}$$

where $\widetilde{Q}_\phi(y_2|x;\alpha)$ can be expressed through the marginal $\widetilde{Q}_\phi(y_1|x)$ via (C.1).

Learning the non-terminal state classifier. The non-terminal state classifier $\widetilde{Q}(y_1, y_2|s;\alpha)$ models y_1 and y_2 jointly. Note that α is one of the inputs to the classifier model. Given a sampled trajectory $\widehat{\tau}$, labels $\widehat{y}_1, \widehat{y}_2$, and $\widehat{\alpha}$, the total cross-entropy loss of all non-terminal states in $\hat{\tau}$ is

$$\ell(\widehat{\tau}, \widehat{y}_1, \widehat{y}_2, \widehat{\alpha}; \phi) = \sum_{t=0}^{|\tau|-1}\left[-\log\widetilde{Q}_\phi(y_1\!=\!\widehat{y}_1, y_2\!=\!\widehat{y}_2|s\!=\!\widehat{s}_t;\widehat{\alpha})\right]. \tag{C.4}$$

The pairs $(\widehat{\tau}, \widehat{y}_1)$ can be generated via a sampling scheme similar to the one used for the terminal state classifier loss above: 1) $\widehat{y}_1 \sim \widetilde{p}(y_1)$ and 2) $\widehat{\tau} \sim p_{\widehat{y}_1}(\tau)$. An approximation of the distribution of $\widehat{y}_2$ given $\widehat{\tau}$ is constructed using (C.1):

$$w_1(\widehat{x}, \widehat{\alpha}; \phi) = \frac{\widehat{\alpha}\widetilde{Q}_\phi(y_1 = 1 | x = \widehat{x})}{\widehat{\alpha}\widetilde{Q}_\phi(y_1 = 1 | x = \widehat{x}) + (1 - \widehat{\alpha})\widetilde{Q}_\phi(y_1 = 2 | x = \widehat{x})} \approx \widetilde{p}(y_2 = 1 | x = \widehat{x}; \widehat{\alpha}), \tag{C.5a}$$

$$w_2(\hat{x}, \widehat{\alpha}; \phi) = \frac{(1 - \widehat{\alpha})\widetilde{Q}_\phi(y_1 = 2 | x = \widehat{x})}{\widehat{\alpha}\widetilde{Q}_\phi(y_1 = 1 | x = \widehat{x}) + (1 - \widehat{\alpha})\widetilde{Q}_\phi(y_1 = 2 | x = \widehat{x})} \approx \widetilde{p}(y_2 = 2 | x = \widehat{x}; \widehat{\alpha}). \tag{C.5b}$$

Since these expressions involve outputs of the terminal state classifier which is being trained simultaneously, we again (see Section 5.5.2) introduce the target network parameters $\overline{\phi}$ that are used to compute the probability estimates (C.5).

The training loss for the non-terminal state classifier is

$$\mathcal{L}_N(\phi, \overline{\phi}) = \mathop{\mathbb{E}}_{\widehat{\alpha} \sim p(\alpha)} \mathop{\mathbb{E}}_{(\widehat{\tau}, \widehat{y}_1) \sim \widetilde{p}(\tau, y_1)} \left[\sum_{\widehat{y}_2 = 1}^{2} w_{\widehat{y}_2}(\widehat{x}, \widehat{\alpha}; \overline{\phi}) \ell(\widehat{\tau}, \widehat{y}_1, \widehat{y}_2, \widehat{\alpha}; \phi) \right], \tag{C.6}$$

where $p(\alpha)$ is sampling distribution over $\alpha \in (0, 1)$. In our experiments, we used the following sampling scheme for α:

$$\widehat{z} \sim U[-B, B], \qquad \widehat{\alpha} = \frac{1}{1 + \exp(-\widehat{z})}. \tag{C.7}$$

C.2 Proof of Proposition 5.4.1

Without loss of generality, we prove the claims of the proposition assuming absolutely continuous distributions p_1, p_2 with probability density functions $p_1(\cdot)$, $p_2(\cdot)$.

Claim 1. For any two distributions p_1, p_2 we have $p_1(x) \geq 0$, $p_2(x) \geq 0$, $\int_\mathcal{X} p_1(x)\, dx = \int_\mathcal{X} p_2(x)\, dx = 1 < \infty$. Then, the RHS of the expression for the parameterized contrast operation $p_1 \,◑_{(1-\alpha)}\, p_2$ (5.13) satisfies

$$\frac{p_1(x)^2}{\alpha p_1(x) + (1 - \alpha) p_2(x)} = \frac{p_1(x)}{\alpha} \cdot \underbrace{\frac{p_1(x)}{p_1(x) + \frac{(1-\alpha)}{\alpha} p_2(x)}}_{\leq 1} \leq \frac{p_1(x)}{\alpha}, \tag{C.8}$$

$\forall\, x \in \operatorname{supp}(p_1) \cup \operatorname{supp}(p_2)$. For points $x \notin \operatorname{supp}(p_1) \cup \operatorname{supp}(p_2)$, we set $\frac{p_1(x)^2}{\alpha p_1(x) + (1-\alpha) p_2(x)} =$ 0 since by construction the composite distributions do not have probability mass outside of the union of the supports of the original distributions. The above implies that

$$\int_{\mathcal{X}} \frac{p_1^2(x)}{\alpha p_1(x) + (1-\alpha) p_2(x)}\, dx \leq \frac{1}{\alpha} \int_{\mathcal{X}} p_1(x)\, dx = \frac{1}{\alpha} < \infty. \tag{C.9}$$

Therefore, the RHS of the expression for the parameterized contrast operation $p_1 \,◑_{(1-\alpha)}\, p_2$ (5.13) can be normalized, and the distribution $p_1 \,◑_{(1-\alpha)}\, p_2$ is well-defined.

Claim 2. For any $\gamma \in (0, 1)$ we provide an infinite collection of distribution pairs p_1 and p_2 such that negation $p_1 \operatorname{neg}_\gamma p_2$ results in a non-normalizable distribution.

For the given $\gamma \in (0, 1)$ we select four numbers $\mu_1 \in \mathbb{R}$, $\mu_2 \in \mathbb{R}$, $\sigma_1 > 0$, $\sigma_2 > 0$ such that

$$\sigma_1^2 \geq \frac{1}{\gamma} \sigma_2^2, \tag{C.10}$$

Consider univariate normal distributions $p_1(x) = \mathcal{N}(x; \mu_1, \sigma_1^2)$, $p_2(x) = \mathcal{N}(x; \mu_2, \sigma_2^2)$ with density functions

$$p_i(x) = \mathcal{N}(x; \mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left\{-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right\}, \qquad i \in \{1, 2\}. \tag{C.11}$$

For such p_1 and p_2, the RHS of (5.3) is

$$\frac{p_1(x)}{(p_2(x))^\gamma} = \frac{1}{\left(\sqrt{2\pi}\right)^{1-\gamma}} \frac{\sigma_2^\gamma}{\sigma_1} \exp\left\{x^2 \left(\frac{\gamma\sigma_1^2 - \sigma_2^2}{2\sigma_1^2\sigma_2^2}\right) + x\left(\frac{\mu_1}{\sigma_1^2} - \gamma\frac{\mu_2}{\sigma_2^2}\right) + \gamma\frac{\mu_2^2}{2\sigma_2^2} - \frac{\mu_1^2}{2\sigma_1^2}\right\}. \tag{C.12}$$

Conditions (C.10) imply that the quadratic function under the exponent above has a non-negative coefficient for x^2. Therefore this function either grows unbounded as $x \to \infty$ (if the coefficients for the quadratic and linear terms are not zero), or constant (if the coefficients for quadratic and linear terms are zero). In either case, $\int_{\mathbb{R}} p_1(x)/(p_2(x))^\gamma\, dx = \infty$.

C.3 Proofs and Derivations

C.3.1 Proof of Proposition 5.5.1

Our goal is to show that the policy (5.21) induces the mixture distribution $p_M(x) = \sum_{i=1}^m \omega_i p_i(x)$.

Preliminaries. In our proof below we use the notion of "the probability of observing a state $s \in \mathcal{S}$ on a GFlowNet trajectory". Following Bengio et al. [19], we abuse the notation and denote this probability by

$$p_i(s) \triangleq p_i(\{\tau : s \in \tau\}) = \sum_{\tau \in \mathcal{T}_{s_0,s}} \prod_{t=0}^{|\tau|-1} p_{i,F}(s_t|s_{t-1}), \tag{C.13}$$

where $\mathcal{T}_{s_0,s}$ is the set of all (sub)trajectories starting at s_0 and ending at s. The probabilities induced by the policy (5.21) are denoted by $p_M(s)$. Note that $p_i(s)$ and $p_M(s)$ should not be interpreted as probability mass functions over the set of states $\mathcal{S}$. In particular $p_i(s_0) = p_M(s_0) = 1$ and sums $\sum_{s\in\mathcal{S}} p_i(s)$, $\sum_{s\in\mathcal{S}} p_M(s)$ are not equal to 1 (unless $\mathcal{S} = \{s_0\}$). However, the functions $p_i(\cdot)$, $p_M(\cdot)$ restricted to the set of terminal states $\mathcal{X}$ give valid probability distributions over $\mathcal{X}$: $\sum_{x\in\mathcal{X}} p_i(x) = \sum_{x\in\mathcal{X}} p_M(x) = 1$.

By definition $p_i(\cdot)$ and $p_M(\cdot)$ satisfy the recurrent relationship

$$p_i(s) = \sum_{s_*:(s_*\to s)\in\mathcal{A}} p_i(s_*)p_{i,F}(s|s_*), \qquad p_M(s) = \sum_{s_*:(s_*\to s)\in\mathcal{A}} p_M(s_*)p_{M,F}(s|s_*). \tag{C.14}$$

The joint distribution of y and τ described in the statement of Proposition 5.5.1 is $p(\tau, y = i) = w_i p_i(\tau)$. This joint distribution over y and trajectories implies the following expressions for the distributions involving intermediate states s.

$$p(y = i) = \omega_i, \tag{C.15}$$

$$p(\tau|y = i) = p_i(\tau) = \prod_{t=0}^{|\tau|-1} p_{i,F}(s_t|s_{t-1}), \tag{C.16}$$

$$p(\tau) = \sum_{i=1}^{m} p(\tau|y{=}i)p(y{=}i) = \sum_{i=1}^{m} \omega_i p_i(\tau), \tag{C.17}$$

$$p(s|y{=}i) = p_i(s), \tag{C.18}$$

$$p(s) = \sum_{i=1}^{m} p(s|y{=}i)p(y{=}i) = \sum_{i=1}^{m} \omega_i p_i(s). \tag{C.19}$$

Proof. Using the notation introduced above, we can formally state our goal. We need to show that $p_M(x)$ induced by $p_{M,F}$ gives the mixture distribution

$$p_M(x) = \sum_{i=1}^{m} \omega_i p_i(x). \tag{C.20}$$

We prove a more general equation for all states $s \in \mathcal{S}$

$$p_M(s) = \sum_{i=1}^{m} \omega_i p_i(s) \tag{C.21}$$

by induction over the DAG $(\mathcal{S}, \mathcal{A})$.

Base case. Consider the initial state $s_0 \in \mathcal{S}$. By definition $p_i(s_0) = p_M(s_0) = 1$ which implies

$$p_M(s_0) = \sum_{i=1}^{m} \omega_i p_i(s_0). \tag{C.22}$$

Inductive step. Consider a state s such that (C.21) holds for all predecessor states $s_* : (s_* \to s) \in \mathcal{A}$. For such a state we have

$$p_M(s) = \sum_{s_*:(s_*\to s)\in\mathcal{A}} p_M(s_*)p_{M,F}(s \,|\, s_*) \quad \{\text{used (C.14)}\} \tag{C.23}$$

$$= \sum_{s_*:(s_*\to s)\in\mathcal{A}} p_M(s_*) \left(\sum_{i=1}^{m} p(y = i|s_*)p_{i,F}(s|s_*) \right) \quad \{\text{used definition of } p_{M,F}\} \tag{C.24}$$

$$= \sum_{s_*:(s_*\to s)\in\mathcal{A}} \frac{p_M(s_*)}{p(s_*)} \left(\sum_{i=1}^{m} p(s_*|y{=}i)p(y{=}i)p_{i,F}(s|s_*) \right) \quad \{\text{used Bayes' theorem}\} \tag{C.25}$$

$$= \sum_{s_*:(s_* \to s) \in \mathcal{A}} \frac{p_M(s_*)}{\sum_{i=1}^m \omega_i p_i(s_*)} \left(\sum_{i=1}^m \omega_i p_i(s_*) p_{i,F}(s|s_*) \right) \quad \{\text{used (C.15), (C.18), (C.19)}\} \tag{C.26}$$

$$= \sum_{s_*:(s_* \to s) \in \mathcal{A}} \left(\sum_{i=1}^m \omega_i p_i(s_*) p_{i,F}(s|s_*) \right) \quad \{\text{used induction hypothesis}\} \tag{C.27}$$

$$= \sum_{i=1}^m \omega_i \left(\sum_{s_*:(s_* \to s) \in \mathcal{A}} p_i(s_*) p_{i,F}(s|s_*) \right) \quad \{\text{changed summation order}\} \tag{C.28}$$

$$= \sum_{i=1}^m \omega_i p_i(s), \qquad \{\text{used (C.14)}\} \tag{C.29}$$

which proves (C.21) for s.

C.3.2 Proof of Proposition 5.5.2

Claim 1. Our goal is to prove the relationship (5.22) for all non-terminal states $s \in \mathcal{S} \setminus \mathcal{X}$. To prove this relationship, we invoke several important properties of Markovian probability flows on DAGs [19].

By Proposition 16 of Bengio et al. [19] for the given GFlowNet forward policy $p_F(\cdot|\cdot)$ there exists a unique backward policy $p_B(\cdot|\cdot)$ such that the probability of any complete trajectory $\tau = (s_0 \to \ldots \to s_{|\tau|} = x)$ in DAG $(\mathcal{S}, \mathcal{A})$ can be expressed as

$$p(\tau) = p(x) \prod_{t=1}^{|\tau|} p_B(s_{t-1}|s_t), \tag{C.30}$$

and the probability of observing a state $s \in \mathcal{S}$ on a trajectory can be expressed as

$$p(s) = \sum_{x \in \mathcal{X}} p(x) \sum_{\tau \in \mathcal{T}_{s,x}} \prod_{t=1}^{|\tau|} p_B(s_{t-1}|s_t), \tag{C.31}$$

where $\mathcal{T}_{s,x}$ is the set of all (sub)trajectories starting at s and ending at x. Moreover, $p_F(\cdot|\cdot)$ and $p_B(\cdot|\cdot)$ are related through the "detailed balance condition" [19, Proposition 21]

$$p(s) p_F(s'|s) = p(s') p_B(s|s'), \qquad \forall\, (s \to s') \in \mathcal{A}. \tag{C.32}$$

By the statement of Proposition 5.5.2, in the probabilistic model $p(x, y)$, the marginal distribution $p(x)$ is realized by the GFlowNet forward policy $p_F(\cdot|\cdot)$ and y is independent of intermediate states s. The joint distribution $p(s, y)$ is given by

$$\begin{aligned}
p(s,y) &= \sum_{x\in\mathcal{X}} p(x,y) \sum_{\tau\in\mathcal{T}_{s,x}} \prod_{t=1}^{|\tau|} p_B(s_{t-1}|s_t) && \text{(C.33)}\\
&= \sum_{x\in\mathcal{X}} p(x,y) \sum_{s':(s\to s')\in\mathcal{A}} p_B(s|s') \sum_{\tau\in\mathcal{T}_{s',x}} \prod_{t=1}^{|\tau|} p_B(s_{t-1}|s_t) && \text{(C.34)}\\
&= \sum_{s':(s\to s')\in\mathcal{A}} p_B(s|s') \underbrace{\sum_{x\in\mathcal{X}} p(x,y) \sum_{\tau\in\mathcal{T}_{s',x}} \prod_{t=1}^{|\tau|} p_B(s_{t-1}|s_t)}_{p(s',y)} && \text{(C.35)}\\
&= \sum_{s':(s\to s')\in\mathcal{A}} p_B(s|s') p(s',y). && \text{(C.36)}
\end{aligned}$$

Expressing the conditional probability $p(y|s)$ through the joint $p(s, y)$ we obtain

$$\begin{aligned}
p(y|s) &= \frac{p(s,y)}{p(s)} \quad \{\text{used definition of conditional probability}\} && \text{(C.37)}\\
&= \frac{1}{p(s)} \sum_{s':(s\to s')\in\mathcal{A}} p_B(s|s')p(s',y) \quad \{\text{used (C.36)}\} && \text{(C.38)}\\
&= \sum_{s':(s\to s')\in\mathcal{A}} p_B(s|s')\frac{p(s')}{p(s)}p(y|s') \quad \{\text{decomposed } p(s',y) = p(s')p(y|s')\} && \text{(C.39)}\\
&= \sum_{s':(s\to s')\in\mathcal{A}} p_F(s'|s)p(y|s'), \quad \{\text{used (C.32)}\} && \text{(C.40)}
\end{aligned}$$

which proves (5.22).

Claim 2. Our goal is to show that the classifier-guided policy (5.23) induces the conditional distribution $p(y|x)$.

We know (Section C.3.1) that the state probabilities induced by the marginal GFlowNet policy $p_F(\cdot|\cdot)$ satisfy the recurrence

$$p(s) = \sum_{s_*:(s_*\to s)\in\mathcal{A}} p(s_*)p_F(s|s_*). \qquad \text{(C.41)}$$

Let $p_y(\cdot)$ denote the state probabilities induced by the classifier-guided policy (5.23). These probabilities by definition (Section C.3.1) satisfy the recurrence

$$p_y(s) = \sum_{s_*:(s_* \to s) \in \mathcal{A}} p_y(s_*) p_F(s|s_*, y). \tag{C.42}$$

We show that

$$p_y(s) = p(s|y), \tag{C.43}$$

by induction over DAG $(\mathcal{S}, \mathcal{A})$.

Base case. Consider the initial state s_0. By definition $p_y(s_0) = 1$. At the same time $p(s_0|y) = p(\{\tau : s_0 \in \tau\}|y) = 1$. Therefore $p_y(s_0) = p(s_0|y)$.

Inductive step. Consider a state s such that (C.43) holds for all predecessor states $s_* : (s_* \to s) \in \mathcal{A}$. For such a state we have

$$\begin{aligned}
p_y(s) &= \sum_{s_*:(s_* \to s) \in \mathcal{A}} p_y(s_*) p_F(s|s_*, y) \quad \{\text{used (C.42)}\} && \text{(C.44)}\\
&= \sum_{s_*:(s_* \to s) \in \mathcal{A}} p_y(s_*) p_F(s|s_*) \frac{p(y|s)}{p(y|s_*)} \quad \{\text{used (5.23)}\} && \text{(C.45)}\\
&= \sum_{s_*:(s_* \to s) \in \mathcal{A}} p(s_*|y) p_F(s|s_*) \frac{p(y|s)}{p(y|s_*)} \quad \{\text{used induction hypothesis}\} && \text{(C.46)}\\
&= \sum_{s_*:(s_* \to s) \in \mathcal{A}} \frac{p(y|s_*)p(s_*)}{p(y)} p_F(s|s_*) \frac{p(y|s)}{p(y|s_*)} \quad \{\text{used Bayes' theorem}\} && \text{(C.47)}\\
&= \frac{p(y|s)}{p(y)} \sum_{s_*:(s_* \to s) \in \mathcal{A}} p(s_*) p_F(s|s_*) \quad \{\text{rearranged terms}\} && \text{(C.48)}\\
&= \frac{p(y|s)}{p(y)} p(s) \quad \{\text{used (C.41)}\} && \text{(C.49)}\\
&= p(s|y), \quad \{\text{used Bayes' theorem}\} && \text{(C.50)}
\end{aligned}$$

which proves (C.43) for state s.

C.3.3 Proof of Theorem 5.5.3

By Proposition 5.5.1 we have that the policy

$$p_{M,F}(s'|s) = \sum_{i=1}^{m} p_{i,F}(s'|s)\widetilde{p}(y{=}i|s), \tag{C.51}$$

generates the mixture distribution $p_M(x) = \frac{1}{m}\sum_{i=1}^{m} p_i(x)$.

In the probabilistic model $\widetilde{p}(x, y_1 \ldots, y_n)$ the marginal distribution $\widetilde{p}(x) = p_M(x)$ is realized by the mixture policy $p_{M,F}$. Therefore, $\widetilde{p}(x, y_1, \ldots, y_n)$ satisfies the conditions of Proposition 5.5.2 which states that the conditional distribution $\widetilde{p}(x|y_1, \ldots, y_n)$ is realized by the classifier-guided policy

$$\begin{aligned} p_F(s'|s, y_1, \ldots, y_n) &= p_{M,F}(s'|s)\frac{\widetilde{p}(y_1, \ldots, y_n|s')}{\widetilde{p}(y_1, \ldots, y_n|s)} \\ &= \frac{\widetilde{p}(y_1, \ldots y_n \mid s')}{\widetilde{p}(y_1, \ldots y_n \mid s)} \sum_{i=1}^{m} p_{i,F}(s'|s)\widetilde{p}(y{=}i|s). \end{aligned} \tag{C.52}$$

C.3.4 Detailed Derivation of Classifier Training Objective

This section provides a more detailed step-by-step derivation of the non-terminal state classifier training objective (5.27).

Step 1. Our goal is to train a classifier $\widetilde{Q}(y_1, \ldots, y_n|s)$. This classifier can be obtained as the optimal solution of

$$\min_{\phi} \mathbb{E}_{\widehat{\tau}, \widehat{y}_1, \ldots, \widehat{y}_n \sim \widetilde{p}(\tau, y_1, \ldots, y_n)} \left[\ell(\widehat{\tau}, \widehat{y}_1, \ldots, \widehat{y}_n; \phi)\right], \tag{C.53}$$

where $\ell(\cdot)$ is defined in equation (5.26). An unbiased estimate of the loss (and its gradient) can be obtained by sampling $(\widehat{\tau}, \widehat{y}_1, \ldots, \widehat{y}_n)$ and evaluating (5.26) directly. However sampling tuples $(\tau, y_1, \ldots, y_n)$ is not straightforward. The following steps describe our proposed approach to the estimation of expectation in (C.53).

Step 2. The expectation in (C.53) can be expressed as

$$\mathbb{E}_{\widehat{\tau},\widehat{y}_1\sim\widetilde{p}(\tau,y_1)}\left[\sum_{\widehat{y}_2=1}^{m}\cdots\sum_{\widehat{y}_n=1}^{m}\left(\prod_{i=2}^{n}\widetilde{p}(y_i\!=\!\widehat{y}_i|x\!=\!\widehat{x})\right)\ell(\widehat{\tau},\widehat{y}_1,\ldots,\widehat{y}_n;\phi)\right], \tag{C.54}$$

where we re-wrote the expectation over $(y_2,\ldots,y_n)|\tau$ as the explicit sum of the form $\mathbb{E}_{q(z)}[g(z)] = \sum_{z\in\mathcal{Z}} q(z)g(z)$. The expectation over (τ, y_1) can be estimated by sampling pairs $(\widehat{\tau},\widehat{y}_1)$ as described in the paragraph after equation (5.26): 1) $\widehat{y}_1 \sim \widetilde{p}(y_1)$ and 2) $\widehat{\tau} \sim p_{\widehat{y}_1}(\tau)$. The only missing part is the probabilities $\widetilde{p}(y_i\!=\!\widehat{y}_i|x\!=\!\widehat{x})$ which are not directly available.

Step 3. Our proposal is to approximate these probabilities as $\widetilde{p}(y_1 = j|x = \widehat{x}) \approx w_j(\widehat{x};\phi) = \widetilde{Q}_\phi(y_1\!=\!j|x\!=\!\widehat{x})$. The idea here is that the terminal state classifier $\widetilde{Q}_\phi(y_1|x)$, when trained to optimality, produces outputs exactly equal to the probabilities $\widetilde{p}(y_1|x)$, and the more the classifier is trained the better is the approximation of the probabilities.

Step 4. Steps 1-3, give a procedure where the computation of the non-terminal state classification loss requires access to the terminal state classifier. As we described in the paragraph preceding equation (5.27), we propose to train non-terminal and terminal classifiers simultaneously and introduce "target network" parameters. The weights w are computed by the target network $\widetilde{Q}_{\overline{\phi}}$.

Combining all the steps above, we arrive at objective (5.27) which we use to estimate the expectation in (C.53).

C.3.5 Assumptions and Proof of Proposition 5.5.4

This subsection provides a formal statement of the assumptions and a more detailed formulation of Proposition 5.5.4.

The assumptions, the formulation of the result, and the proof below closely follow those of Theorem 1 of Peluchetti [192]. Theorem 1 in [192] generalizes the result of Brigo [31] (Corollary 1.3), which derives the SDE for mixtures of 1D diffusion processes.

We found an error in the statement and the proof of Theorem 1 (Appendix A.2 of [192]). The error makes the result of [192] for D-dimensional diffusion processes disagree with the result of [31] for 1-dimensional diffusion processes.

Here we provide a corrected version of Theorem 1 of [192] in a modified notation and a simplified setting (mixture of finite rather than infinite number of diffusion processes). Most of the content is directly adapted from [192].

Notation:

- for a vector-valued $f : \mathbb{R}^D \to \mathbb{R}^D$, the divergence of f is denoted as $\nabla \cdot (f(x)) = \sum_{d=1}^{D} \frac{\partial}{\partial x_d} f_d(x)$,
- for a scalar-values $a : \mathbb{R}^D \to \mathbb{R}$, the divergence of the gradient of a (the Laplace operator) is denoted by $\Delta(a(x)) = \nabla \cdot (\nabla a(x)) = \sum_{d=1}^{D} \frac{\partial^2}{\partial x_d^2} a(x)$.

Assumption 1 (SDE solution). A given D-dimensional SDE(f, g):

$$dx_t = f_t(x_t)dt + g_t dw_t, \tag{C.55}$$

with associated initial distribution $p_0(x)$ and integration interval $[0, T]$ admits a unique strong solution on $[0, T]$.

Assumption 2 (SDE density). A given D-dimensional SDE(f, g) with associated initial distribution $p_0(x)$ and integration interval $[0, T]$ admits a marginal density on $(0, T)$ with respect to the D-dimensional Lebesgue measure that uniquely satisfies the Fokker-Plank (Kolmogorov-forward) partial differential equation (PDE):

$$\frac{\partial p_t(x)}{\partial t} = -\nabla \cdot (f_t(x) p_t(x)) + \frac{1}{2}\Delta(g_t^2 p_t(x)). \tag{C.56}$$

Assumption 3 (positivity). For a given stochastic process, all finite-dimensional densities, conditional or not, are strictly positive.

Theorem C.3.1 (Diffusion mixture representation)**.** *Consider the family of D-dimensional SDEs on $t \in [0, T]$ indexed by $i \in \{1, \ldots, m\}$,*

$$dx_{i,t} = f_{i,t}(x_{i,t})dt + g_{i,t} dw_{i,t}, \qquad x_{i,0} \sim p_{i,0}, \tag{C.57}$$

where the initial distributions $p_{i,0}$ and the Wiener processes $w_{i,t}$ are all independent. Let $p_{i,t}$, $t \in (0, T)$ denote the marginal density of $x_{i,t}$. For mixing weights $\{\omega_i\}_{i=1}^m$, $\omega_i \geq 0$, $\sum_{i=1}^m \omega_i = 1$, define the mixture marginal density $p_{M,t}$ for $t \in (0, T)$ and the mixture initial distribution $p_{M,0}$ by

$$p_{M,t}(x) = \sum_{i=1}^m \omega_i p_{i,t}(x) \quad p_{M,0}(x) = \sum_{i=1}^m \omega_i p_{M,0}(x). \tag{C.58}$$

Consider the D-dimensional SDE on $t \in [0, T]$ defined by

$$f_{M,t}(x) = \frac{\sum_{i=1}^m \omega_i p_{i,t}(x) f_{i,t}(x)}{p_{M,t}(x)}, \qquad g_{M,t}(x) = \sqrt{\frac{\sum_{i=1}^m \omega_i p_{i,t}(x) g_{i,t}^2}{p_{M,t}(x)}}, \tag{C.59}$$

$$dx_t = f_{M,t}(x_t)dt + g_{M,t}(x_t)dw_t, \qquad x_{M,0} \sim p_{M,0}. \tag{C.60}$$

It is assumed that all diffusion processes $x_{i,t}$ and the diffusion process $x_{M,t}$ satisfy the regularity Assumptions 1, 2, and 3. Then the marginal distribution of the diffusion $x_{M,t}$ is $p_{M,t}$.

Proof. For $0 < t < T$ we have that

$$\frac{\partial p_{M,t}(x)}{\partial t} = \frac{\partial}{\partial t}\left(\sum_{i=1}^m \omega_i p_{i,t}(x)\right) \tag{C.61}$$

$$= \sum_{i=1}^m \omega_i \frac{\partial p_{i,t}(x)}{\partial t} \tag{C.62}$$

$$= \sum_{i=1}^m \omega_i \left(-\nabla \cdot (f_{i,t}(x)p_{i,t}(x)) + \frac{1}{2}\Delta(g_{i,t}^2 p_{i,t}(x))\right) \tag{C.63}$$

$$= \sum_{i=1}^m \omega_i \left(-\nabla \cdot \left(\frac{p_{i,t}(x)f_{i,t}(x)}{p_{M,t}(x)} p_{M,t}(x)\right) + \frac{1}{2}\Delta\left(\frac{p_{i,t}(x)g_{i,t}^2}{p_{M,t}(x)} p_{M,t}(x)\right)\right) \tag{C.64}$$

$$= -\nabla \cdot \left(\sum_{i=1}^m \frac{\omega_i p_{i,t}(x)f_{i,t}(x)}{p_{M,t}(x)} p_{M,t}(x)\right) + \frac{1}{2}\Delta\left(\sum_{i=1}^m \frac{\omega_i p_{i,t}(x)g_{i,t}^2}{p_{M,t}(x)} p_{M,t}(x)\right) \tag{C.65}$$

$$= -\nabla \cdot (f_{M,t}(x)p_{M,t}(x)) + \frac{1}{2}\Delta(g_{M,t}^2 p_{M,t}(x)). \tag{C.66}$$

The second is an exchange of the order of summation and differentiation, the third line is the application of the Fokker-Planck PDEs for processes $x_{i,t}$, the fourth line is a rewriting in terms of $p_{M,t}$, the fifth line is another exchange of the order of summation and differentiation. The result follows by noticing that $p_{M,t}(x)$ satisfies the Fokker-Planck equation of SDE(f_M, g_M). $\square$

Proof of Proposition 5.5.4. Below, we show that the result of Proposition 5.5.4 follows from Theorem C.3.1.

First, we rewrite $f_{M,t}(x_t)$ and $g_{M,t}(x_t)$ in (C.59) in terms of the classifier probabilities (5.30):

$$f_{M,t}(x_t) = \frac{\sum_{i=1}^m \omega_i p_{i,t}(x_t) f_{i,t}(x_t)}{p_{M,t}(x_t)} = \sum_{i=1}^m p(y\!=\!i|x_t) f_{i,t}(x_t), \tag{C.67}$$

$$g_{M,t}(x_t) = \sqrt{\frac{\sum_{i=1}^m \omega_i p_{i,t}(x_t) g_{i,t}^2}{p_{M,t}(x_t)}} = \sqrt{\sum_{i=1}^m p(y\!=\!i|x_t) g_{i,t}^2}. \tag{C.68}$$

With these expressions, we apply the result of Theorem C.3.1 to the base forward processes $dx_{i,t} = f_{i,t}(x_{i,t})\,dt + g_{i,t}\,dw_{i,t}$ and obtain the mixture forward process in equation (5.28).

From the forward process, we derive the backward process following Song et al. [231]. Using the result of Anderson [7], the backward process for (5.28) is given by

$$dx_t = \left[f_{M,t}(x_t) - \nabla_{x_t}(g_{M,t}^2(x_t)) - g_{M,t}^2(x_t)\nabla_{x_t}\log p_{M,t}(x_t)\right]dt + g_{M,t}(x_t)\,d\overline{w}_t. \tag{C.69}$$

Note that the term $\nabla_{x_t}(g_{M,t}^2(x_t))$ is due to the fact that the diffusion coefficient $g_{M,t}(x_t)$ in (5.28) is a function of x (cf., equation (16), Appendix A in [231]). This term can be transformed as follows

$$\nabla_{x_t}(g_{M,t}^2(x_t)) = \nabla_{x_t}\left(\sum_{i=1}^m p(y\!=\!i|x_t) g_{i,t}^2\right) \tag{C.70}$$

$$= \sum_{i=1}^m g_{i,t}^2 \nabla_{x_t} p(y\!=\!i|x_t) \tag{C.71}$$

$$= \sum_{i=1}^m p(y\!=\!i|x_t) g_{i,t}^2 \nabla_{x_t}\log p(y\!=\!i|x_t) \tag{C.72}$$

$$= \sum_{i=1}^{m} p(y\!=\!i|x_t) g_{i,t}^2 \nabla_{x_t} \Big(\log \omega_i + \log p_{i,t}(x_t) - \log p_{M,t}(x_t) \Big) \tag{C.73}$$

$$= \sum_{i=1}^{m} p(y\!=\!i|x_t) g_{i,t}^2 \Big(\nabla_{x_t} \log p_{i,t}(x_t) - \nabla_{x_t} \log p_{M,t}(x_t) \Big) \tag{C.74}$$

$$= \left(\sum_{i=1}^{m} p(y\!=\!i|x_t) g_{i,t}^2 s_{i,t}(x_t) \right) - g_{M,t}^2(x_t) \nabla_{x_t} \log p_{M,t}(x_t). \tag{C.75}$$

Substituting the last expression in (C.69), we notice that the term $g_{M,t}^2(x_t)\nabla_{x_t} \log p_{M,t}(x_t)$ cancels out, and, after simple algebraic manipulations, we arrive at (5.29).

C.3.6 Proof of Theorem 5.5.5

We proove Theorem 5.5.5 in the assumptions of Section C.3.5. In Section C.3.5 we established that the mixture diffusion process has the forward SDE

$$dx_t = f_{M,t}(x_t)dt + g_{M,t}(x_t)\, dw_t. \tag{C.76}$$

and the backward SDE

$$dx_t = \big[f_{M,t}(x_t) - \nabla_{x_t}(g_{M,t}^2(x_t)) - g_{M,t}^2(x_t) \nabla_{x_t} \log p_{M,t}(x_t) \big]\, dt + g_{M,t}(x_t)\, d\overline{w}_t. \tag{C.77}$$

We apply classifier guidance with classifier $\widetilde{p}(y_1, \dots, y_n|x_t)$ to the mixture diffusion process, following Song et al. [231] (see equations (48)-(49) in [231]). The backward SDE of the classifier-guided mixture diffusion is

$$\begin{aligned} dx_t = &\Big[f_{M,t}(x_t) - \nabla_{x_t}(g_{M,t}^2(x_t)) \\ &- g_{M,t}^2(x_t) \Big(\nabla_{x_t} \log p_{M,t}(x_t) + \nabla_{x_t} \log \widetilde{p}(y_1, \dots, y_n|x_t) \Big) \Big] dt + g_{M,t}(x_t)\, d\overline{w}_t. \end{aligned} \tag{C.78}$$

Finally, we arrive at (5.32) by substituting (C.75) in the above, canceling out the term $g_{M,t}^2(x_t)\nabla_{x_t} \log p_{M,t}(x_t)$, and applying simple algebraic manipulations.

C.4 Implementation Details

C.4.1 Classifier Guidance in GFlowNets

Classifier guidance in GFlowNets (5.23) is realized through modification of the base forward policy via the multiplication by the ratio of the classifier outputs ${}^{p(y|s')}/_{p(y|s)}$. The ground truth (theoretically optimal) non-terminal state classifier $p(y|s)$ by Proposition 5.5.2 satisfies (5.22) which ensures that the guided policy (5.23) is valid, i.e. for any state $s \in \mathcal{S}$

$$\sum_{s':(s\to s')\in\mathcal{A}} p_F(s'|s,y) = \sum_{s':(s\to s')\in\mathcal{A}} p_F(s'|s)\frac{p(y|s')}{p(y|s)} \tag{C.79}$$

$$= \frac{1}{p(y|s)} \underbrace{\sum_{s':(s\to s')\in\mathcal{A}} p_F(s'|s)p(y|s')}_{=p(y|s) \text{ by Proposition 5.5.2}} = 1. \tag{C.80}$$

In practice, the ground truth values of $p(y|s)$ are unavailable. Instead, an approximation $Q_\phi(y|s) \approx p(y|s)$ is learned. Equation (5.22) might not hold for the learned classifier Q_ϕ, but we still wish to use $\widetilde{Q}_\phi$ for classifier guidance in practice. In order to ensure that the classifier-guided policy is valid in practice even when the approximation $Q_\phi(y|s)$ of the classifier $p(y|s)$ is used, we implement guidance as described below.

First, we express the guided policy (5.23) in terms of log-probabilities:

$$\log p_F(s'|s,y) = \log p_F(s'|s) + \log p(y|s') - \log p(y|s). \tag{C.81}$$

Parameterizing distributions through log-probabilities is common practice in probabilistic modeling: GFlowNet forward policies [16, 158] and probabilistic classifiers are typically parameterized by deep neural networks that output logits (unnormalized log-probabilities).

Second, in the log-probability parameterization the guided policy (5.23) can be equivalently expressed as

$$p_F(s'|s,y) = \Big[\text{softmax}\Big(\log p_F(\cdot|s) + \log p(y|\cdot) - \log p(y|s)\Big)\Big]_{s'} \tag{C.82}$$

$$= \frac{\exp\Big(\log p_F(s'|s) + \log p(y|s') - \log p(y|s)\Big)}{\sum_{s'':(s\to s'')\in\mathcal{A}} \exp\Big(\log p_F(s''|s) + \log p(y|s'') - \log p(y|s)\Big)}. \tag{C.83}$$

In theory, the softmax operation can be replaced with simple exponentiation, i.e. the numerator in (C.83) is sufficient on its own since Proposition 5.5.2 ensures that the sum in the denominator equals to 1. However, using the softmax is beneficial in practice when we substitute learned classifier $Q_\phi(y|s)$ instead of the ground truth classifier $p(y|s)$. Indeed when $Q_\phi(y|s)$ does not satisfy (5.22), the softmax operation ensures that the guided policy

$$p_F(s'|s,y) = \Big[\text{softmax}\Big(\log p_F(\cdot|s) + \log Q_\phi(y|\cdot) - \log Q_\phi(y|s)\Big)\Big]_{s'} \tag{C.84}$$

is valid (i.e. probabilities sum up to 1 over s'). The fact the softmax expression is valid in theory ensures that policy (C.84) guided by $Q_\phi(y|s)$ approaches the ground truth policy (guided by $p(y|s)$) as $Q_\phi(y|s)$ approaches $p(y|s)$ throughout training.

C.5 Experiment details

C.5.1 2D Distributions with GFlowNets

The base GFlowNet forward policies $p_{i,F}(s'|s;\theta)$ were parameterized as MLPs with 2 hidden layers and 256 units in each hidden layer. The cell coordinates of a state s on the 2D 32×32 grid were one-hot encoded. The dimensionality of the input was $2 \cdot 32$. The outputs of the forward policy network were the logits of the softmax distribution over 3 action choices: 1) move down; 2) move right; 3) stop.

We trained base GFlowNets with the trajectory balance loss [158]. We fixed the uniform backward policy in the trajectory balance objective. We used Adam optimizer [120] with learning rate 0.001, and pre-train the base models for 20 000 steps with batch size 16 (16 trajectories per batch). The log of the total flow $\log Z_\theta$ was optimized with Adam with a learning rate 0.1. In order to promote exploration in trajectory sampling for the trajectory balance objective, we used the sampling policy which takes random actions (uniformly) with probability 0.05 but, otherwise, follows the current learned forward policy.

The classifier was parameterized as MLP with 2 hidden layers and 256 units in each hidden layer. The inputs to the classifier were one-hot encoded cell coordinates, terminal state flag ($\{0, 1\}$), and $\log(\alpha/(1-\alpha))$ (in the case of parameterized operations). The classifier outputs were the logits of the joint label distribution $\widetilde{Q}_\phi(y_1, \ldots, y_n|s)$ for non-terminal states s and the logits of the marginal label distribution $\widetilde{Q}_\phi(y_1|x)$ for terminal states x.

We trained the classifier with the loss described in Section 5.5.2. We used Adam with learning rate 0.001. We performed 15 000 training steps with batch size 64 (64 trajectories sampled from each of the base models per training step). We updated the target network parameters $\overline{\phi}$ as the exponential moving average (EMA) of ϕ with the smoothing factor 0.995. We linearly increased the weight $\gamma(\text{step})$ of the non-terminal state loss from 0 to 1 throughout the first 3 000 steps and kept constant $\gamma = 1$ afterward. For the α-parameterized version of the classifier (Section C.1), we used the following sampling scheme for α: $\widehat{z} \sim U[-3.5, 3.5]$, $\widehat{\alpha} = \frac{1}{1+\exp(-\widehat{z})}$.

Quantitative evaluation. For each of the composite distributions shown in Figure 5-3 we evaluated the L1-distance $L_1(p_{\text{method}}, p_{\text{GT}}) = \sum_{x\in\mathcal{X}} |p_{\text{method}}(x) - p_{\text{GT}}(x)|$ between the distribution p_{method} induced by the classifier-guided policy and the ground-truth composition distribution p_{GT} computed from the known base model probabilities p_i. The evaluation results are presented below.

Figure 5-3 **top row**. $p_1 \otimes p_2$: $L_1 = 0.071$; $p_1 \,◑\, p_2$: $L_1 = 0.086$; $p_1 \,◑_{0.95}\, p_2$: $L_1 = 0.167$.

Figure 5-3 **bottom row**. $\widetilde{p}(x|y_1{=}1, y_2{=}2)$: $L_1 = 0.076$; $\widetilde{p}(x|y_1{=}1, y_2{=}2, y_3{=}3)$: $L_1 = 0.087$; $\widetilde{p}(x|y_1{=}2, y_2{=}2)$: $L_1 = 0.112$; $\widetilde{p}(x|y_1{=}2, y_2{=}2, y_3{=}2)$: $L_1 = 0.122$.

Figure 5-7 shows the distance between the composition and the ground truth as a function of the number of training steps for the classifier as well as the terminal and non-terminal classifier learning curves.

C.5.2 Molecule Generation

Reward normalization. We used the following normalization rules for SEH, SA, and QED rewards in the molecule domain.

- $\text{SEH} = {}^{\text{SEH}_{\text{raw}}}/_{8}$;
- $\text{SA} = \frac{10-\text{SA}_{\text{raw}}}{9}$;
- $\text{QED} = \text{QED}_{\text{raw}}$.

Training details and hyperparameters. The base GFlowNet policies were parameterized as graph neural networks with Graph Transformer architecture [268]. We used 6 transformer layers with an embedding of size 128. The input to the Graph Transformer was the graph of fragments with node attributes describing fragments and edge attributes describing attachment points of the edges.

The base GFlowNets were trained with trajectory balance loss. We used Adam optimizer. For the policy network $p_F(s'|s;\theta)$, we set the initial learning rate 0.0005 and exponentially decayed with the factor $2^{\text{step}/20\,0000}$. For the log of the total flow $\log Z_\theta$ we set the initial learning rate 0.0005 and exponentially decayed with the factor $2^{\text{step}/50\,000}$. We trained the base GFlowNets for 15 000 steps with batch size 64 (64 trajectories per batch). In order to promote exploration in trajectory sampling for the trajectory balance objective, we used the sampling policy which takes random actions (uniformly) with probability 0.1 but, otherwise, follows the current learned forward policy.

The classifier was parameterized as a graph neural network with Graph Transformer architecture. We used 4 transformer layers with embedding size 128. The inputs to the classifier were the fragment graph, terminal state flag ($\{0, 1\}$), and $\log({}^{\alpha}/_{(1-\alpha)})$ (in case of parameterized operations). The classifier outputs were the logits of the joint label distribution $\widetilde{Q}_\phi(y_1, \ldots, y_n|s)$ for non-terminal states s and the logits of the marginal label distribution $\widetilde{Q}_\phi(y_1|x)$ for terminal states x.

We trained the classifier with the loss described in Section 5.5.2. We used Adam with learning rate 0.001. We performed 15 000 training steps with batch size 8 (8 trajectories sampled from each of the base models per training step). We updated the target network parameters $\overline{\phi}$ as the exponential moving average (EMA) of ϕ with the smoothing factor 0.995. We linearly increased the weight $\gamma(\text{step})$ of the non-terminal state loss from 0 to 1 throughout the first 4 000 steps and kept constant $\gamma = 1$ afterward. For the α-parameterized version of the classifier (Section C.1), we used the following sampling scheme for α: $\widehat{z} \sim U[-5.5, 5.5]$, $\widehat{\alpha} = \frac{1}{1+\exp(-\widehat{z})}$.

C.5.3 Colored MNIST Generation via Diffusion Models

The colored MNIST experiment in Section 5.6.3 follows the method for composing diffusion models introduced in Section 5.5.3. The three base diffusion models were trained on colored MNIST digits generated from the original MNIST dataset. These colored digits were created by mapping MNIST images from their grayscale representation to either the red or green channel, leaving the other channels set to 0. For Figure 5-6 we post-processed the red and green images generated by the base models and compositions into beige and cyan respectively, which are more accessible colors for colorblind people.

Models, training details, and hyperparameters. The base diffusion models were defined as VE SDEs [231]. Their score models were U-Net [213] networks consisting of 4 convolutional layers with 64, 128, 256, and 256 channels and 4 matching transposed convolutional layers. Time was encoded using 256-dimensional Gaussian random features [240]. The score model was trained using Adam optimizer [120] with a learning rate decreasing exponentially from 10^{-2} to 10^{-4}. We performed 200 training steps with batch size 32.

The first classifier $\widetilde{Q}(y_1, y_2|x_t)$ was a convolutional network consisting of 2 convolutional layers with 64 and 96 channels and three hidden layers with 512, 256 and 256 units. This classifier is time-dependent and used 128-dimensional Gaussian random features to embed the time. The output was a 3x3 matrix encoding the predicted log-probabilities. The classifier was trained on trajectories sampled from the reverse SDE of the base diffusion models using the AdaDelta optimizer [270] with a learning rate of 1.0. We performed 700 training steps with batch size 128. For the first 100 training steps the classifier was only trained on terminal samples.

The second conditional classifier $\widetilde{Q}(y_3|y_1, y_2, x_t)$ was a similar convolutional network with 2 convolutional layers with 64 channels and two hidden layers with 256 units. This classifier is conditioned both on time and on (y_1, y_2). The time variable was embedded used 128-dimensional Gaussian random features. The (y_1, y_2) variables were encoded using a 1-hot encoding scheme. The output of the classifier was the three predicted log-probabilities for y_3. Contrary to the first classifier, this one was not trained on the base diffusion models but rather on samples from the posterior $\widetilde{p}(x|y_1, y_2)$. It's loss function was:

$$\mathcal{L}_c(\phi) = \mathop{\mathbb{E}}_{(\widehat{x}_0, \widehat{x}_t, \widehat{y}_2, \widehat{y}_1, t) \sim \widetilde{p}(x_0, x_t|y_1, y_2)\widetilde{p}(y_1)\widetilde{p}(y_2)p(t)} \left[\sum_{\widehat{y}_3=1}^{m} -w_{\widehat{y}_3}(\widehat{x}_0) \log \widetilde{Q}_\phi(\widehat{y}_3|\widehat{y}_1, \widehat{y}_2, \widehat{x}_t) \right] \tag{C.85}$$

where $w_{\widehat{y}_3}(\widehat{x}_0)$ is estimated using the first classifier. The classifier was trained using the AdaDelta optimizer [270] with a learning rate of 0.1. We performed 200 training steps with batch size 128.

Sampling. Sampling from both the individual base models and the composition was done using the Predictor-Corrector sampler [231]. We performed sampling over 500 time steps to generate the samples shown in Figure 5-6. The samples used to train the classifier were generated using the same method.

When sampling from the composition we found that using scaling for the classifier guidance was generally necessary to achieve high-quality results. Without scaling, the norm of the gradient over the first and second classifier was too small relative to the gradient predicted by the score function, and hence did not sufficiently steer the mixture towards samples from the posterior. Experimentally, we found that scaling factor 10 for the first classifier and scaling factor 75 for the second produced high quality results.

C.6 Additional Results

C.6.1 Analysis of Sample Diversity of Base GFlowNets in Molecule Generation Domain

In order to assess the effect of the reward exponent β on mode coverage and sample diversity, we evaluated samples generated from GFlowNets pre-trained with different values of β. The results are in Tables C.1 and C.2. The details of the evaluation and the reported metrics are described in the table captions. As expected, larger reward exponents shift the learned distributions towards high-scoring molecules (the total number of molecules with scores above the threshold increases). For 'SA' and 'QED' models we don't observe negative effects of large β on sample diversity and mode coverage: the average pairwise similarity of top 1 000 molecules doesn't grow as β increases and the ratio of Tanimoto-separated modes remains high. For 'SEH' models we observe a gradual increase in the average pairwise similarity of top 1 000 molecules and a gradual decrease in the ratio of Tanimoto-separated modes. However, the total number of separated modes grows as β increases, which indicates that larger reward exponents do not lead to mode dropping.

C.6.2 Binary Operations for MNIST Digit Generation via Diffusion Models

Here we present a variant of the colored digit generation experiment from Section 5.6.3 using 2 diffusion models. This allows us to better illustrate the harmonic mean and contrast operations on this image domain. In a similar fashion to the experiment in Section 5.6.3, we trained two diffusion models to generate colored MNIST digits. p_1 was trained to generate red and green 0 digits and p_2 was trained to generate green 0 and 1 digits. As before, we used post-processing to map green to cyan and red to beige.

Table C.1: Average pairwise similarity [16] of molecules generated by GFlowNets trained on 'SEH', 'SA', 'QED' rewards at different values of β. For each combination (reward, β) a GFlowNet was trained with the corresponding reward $R(x)^\beta$. Then, 5 000 molecules were generated. The numbers in the table reflect the average pairwise Tanimoto similarity of top 1 000 molecules (selected according to the target reward function).

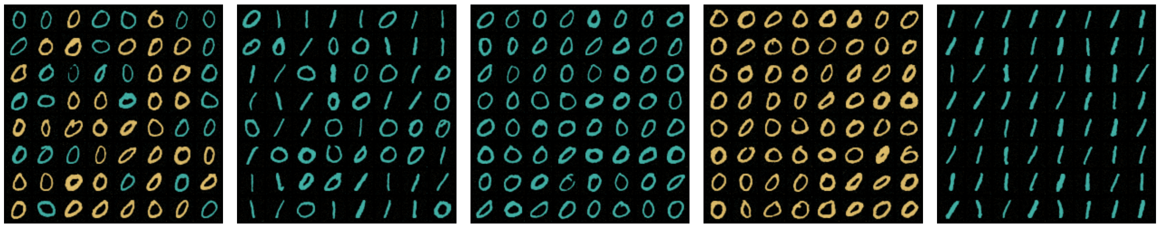
	SEH	SA	QED
$\beta = 1$	0.527	0.539	0.480
$\beta = 4$	0.529	0.527	0.464
$\beta = 10$	0.535	0.500	0.438
$\beta = 16$	0.548	0.465	0.422
$\beta = 32$	0.585	0.411	0.398
$\beta = 96$	0.618	0.358	0.404

Table C.2: Number of Tanimoto-separated modes found above reward threshold. For each combination (reward, β) a GFlowNet was trained with the corresponding reward $R(x)^\beta$, and then 5 000 molecules were generated. Cell format is "A/B", where A is the number of Tanimoto-separated modes found above the reward threshold, and B is the total number of generated molecules above the threshold. Analogously to Figure 14 in [16], we consider having found a new mode representative when a new molecule has Tanimoto similarity smaller than 0.7 to every previously found mode's representative molecule. Reward thresholds (in $[0, 1]$, normalized values) are 'SEH': 0.875, 'SA': 0.75, 'QED': 0.75. Note that the normalized threshold of 0.875 for 'SEH' corresponds to the unnormalized threshold of 7 used in [16].

	SEH	SA	QED
$\beta = 1$	15 / 17	37 / 37	0 / 0
$\beta = 4$	12 / 17	82 / 82	0 / 0
$\beta = 10$	85 / 109	332 / 337	18 / 18
$\beta = 16$	190 / 280	886 / 910	253 / 253
$\beta = 32$	992 / 1821	2859 / 3080	3067 / 3124
$\beta = 96$	1619 / 4609	4268 / 4983	4470 / 4980

Implementation details. The diffusion models used in this experiment and their training procedure were exactly the same as in Section C.5.3. The sampling method used to obtain samples from the base models and their compositions was also the same. We found that scaling the classifier guidance was generally required for high-quality results, and used a scaling factor of 20 in this experiment.

The classifier was a convolutional network with 2 convolutional layers consisting of 32 and 64 channels and two hidden layers with 512 and 128 units. The classifier's time input was embedded using 128-dimensional Gaussian random features. The output was a 2x2 matrix encoding the predicted log-probabilities $\widetilde{Q}(y_1, y_2 \mid x_t)$. The classifier was trained on trajectories, sampled from the reverse SDE of the base diffusion models, using the AdaDelta optimizer [270] with a learning rate of 0.1 and a decay rate of 0.97. We performed 200 training steps with batch size 128. For the first 100 training steps, the classifier was only trained on terminal samples.



(a) p_1 (b) p_2 (c) $p_1 \otimes p_2$ (d) p_1 ◑ p_2 (e) p_1 ◐ p_2

Figure C-1: **Diffusion model composition on colored MNIST.** (a,b) samples from base diffusion models. (c-e) samples from the resulting harmonic mean and contrast compositions.

Results. Figure C-1 shows samples obtained from the two trained diffusion models p_1, p_2 and from the harmonic mean and contrast compositions of these models. We observe that the harmonic mean generates only cyan zero digits, because this is the only type of digit on which both p_1 and p_2 have high density. The contrast p_1 ◑ p_2 generates beige zero digits from p_1. However, unlike p_1, it does not generate cyan zero digits, as p_2 has high density there. The story is similar for p_1 ◐ p_2, which generates cyan one digits from p_2, but not zero digits due to p_1 having high density over those.

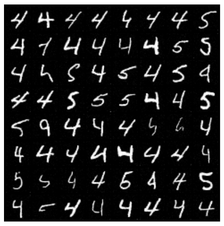
C.6.3 MNIST Subset Generation via Diffusion Models

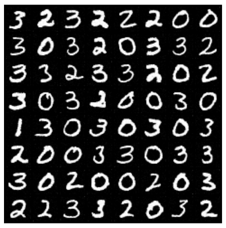
We report in this section on additional results for composing diffusion models on the standard MNIST dataset. We trained two diffusion models: $p_{\{0,\ldots,5\}}$ was trained to generate MNIST digits 0 through 5, and $p_{\{4,\ldots,9\}}$ to generate MNIST digits 4 through 9. The training procedure and models used in this experiment were the same as in Section C.6.2.

Figure C-2 shows samples obtained from the two diffusion models, from the harmonic mean, and from the contrast compositions of these models. We observe that the harmonic mean correctly generates mostly images on which both diffusion models' sampling distributions have high probability, i.e. digits 4 and 5. For the contrasts we see that in both cases digits are generated that have high probability under one model but low probability under the other. We observe some errors, namely some 9's being generated by the harmonic mean and some 4's being generated by the contrast $p_{\{0,\ldots,5\}}$ ◐ $p_{\{4,\ldots,9\}}$. This is likely because 4 and 9 are visually similar,

(a) $p_{\{0,\ldots,5\}}$

(b) $p_{\{4,\ldots,9\}}$


(c)
$p_{\{0,\ldots,5\}} \otimes p_{\{4,\ldots,9\}}$


(d)
$p_{\{0,\ldots,5\}} \mathbin{◑} p_{\{4,\ldots,9\}}$

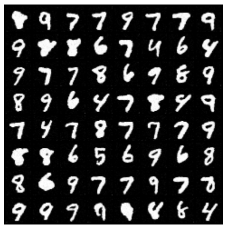
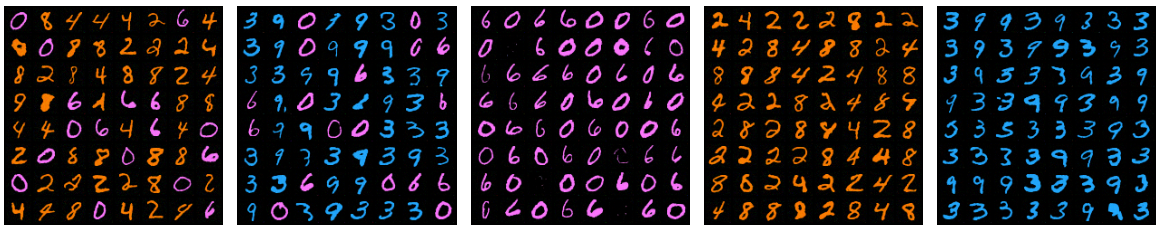

(e)
$p_{\{0,\ldots,5\}} \mathbin{◐} p_{\{4,\ldots,9\}}$

Figure C-2: **Diffusion model composition on MNIST.** (a,b) samples from base diffusion models. (c-e) samples from the resulting harmonic mean and contrast compositions.

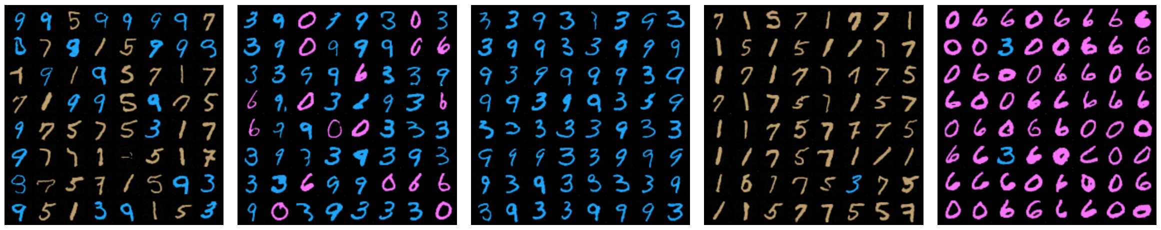
causing the guiding classifier to misclassify them, and generate them under the wrong composition.

We also present binary operations between three distributions. In Figure C-3 and C-4, p_0 models even digits, p_1 models odd digits, and p_2 models digits that are divisible by 3. We color digits $\{0, 6\}$ purple, $\{3, 9\}$ blue, $\{4, 8\}$ orange, and $\{1, 5, 7\}$ beige. In Figure C-3, harmonic mean of p_0 and p_2 generates the digit 0 and 6, whereas the contrast of p_0 with p_2 shows even digits non-divisible by 3 ($p_0 \mathbin{◑} p_2 = \{4, 8\}$), and odd numbers that are divisible by 3 ($p_0 \mathbin{◐} p_2 = \{3, 9\}$). We observe that the samples from $p_0 \otimes p_2$ inherit artifacts from the base generator for p_2 (the thin digit 0), which shows the impact that the base models have on the composite distribution. In Figure C-4 we present similar results between odd digits ($\{5, 3, 5, 7, 9\}$. We noticed that samples from both $p_1 \mathbin{◑} p_2$ and $p_1 \mathbin{◐} p_2$ includes a small number of the digit 3.



(a) p_0:Even Digits (b) p_2: $\{0, 3, 6, 9\}$ (c) $p_0 \otimes p_2$ (d) p_0 ◑ p_2 (e) p_0 ◐ p_2

Figure C-3: **Composing even digits and multiples of three on Colored MNIST.** (a,b) samples from base diffusion models. (c-e) samples from the resulting harmonic mean and contrast compositions.



(a) p_1: Odd Digits (b) p_2: $\{0, 3, 6, 9\}$ (c) $p_1 \otimes p_2$ (d) p_1 ◑ p_2 (e) p_1 ◐ p_2

Figure C-4: **Composing odd digits and multiples of three on Colored MNIST.** (a,b) samples from base diffusion models. (c-e) samples from the resulting harmonic mean and contrast compositions.

C.6.4 Chaining: Sequential Composition of Multiple Distributions

We present results on chaining binary composition operations sequentially on a custom colored MNIST dataset.

Setup. We start with three base generative models that are trained to generate p_1, p_2 and p_3 in Figure C-5. Specifically, p_1 is a uniform distribution over digits $\{0, 1, 2, 3, 4, 5\}$, p_2 is a uniform distribution over even digits $\{0, 2, 4, 6, 8\}$, and p_3 is a uniform distribution over digits divisible by 3: $\{0, 3, 6, 9\}$. Note that we use a different color for each digit consistent across p_1, p_2, p_3. Our goal is to produce combinations of chained binary operations involving all three distributions, where two of them were combined first, then in a second step, combined with the third distribution through either harmonic mean $\otimes$ or contrast ◑.

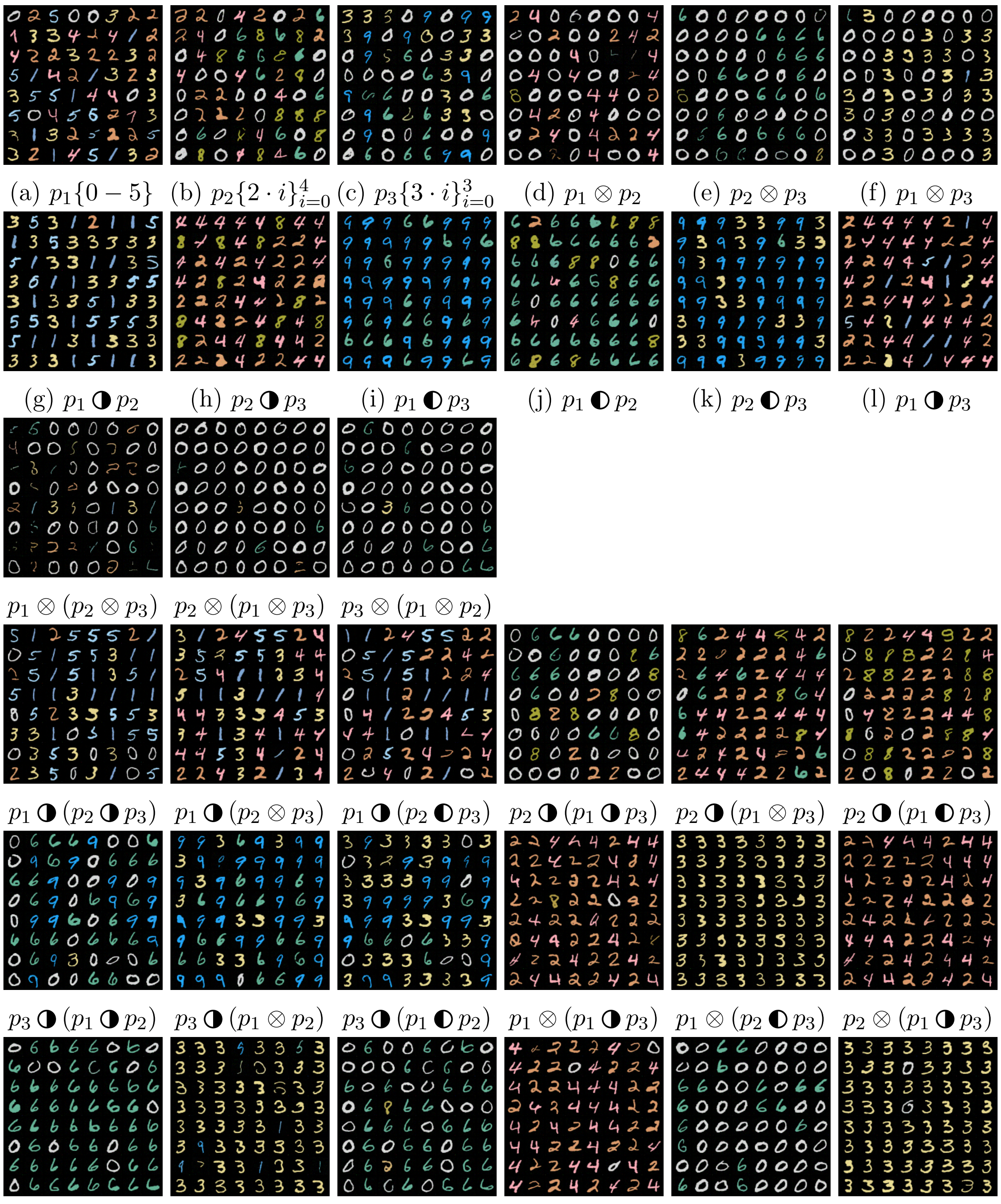
Binary Classifier Training. Consider, for example, the operation $(p_1 \otimes p_2) \mathbin{◑} p_3$. We use the same classifier training procedure for p_1 versus p_2, as well as the composite model $(p_1 \otimes p_2)$ versus p_3, except that in the later case we sample from composite model as a whole. Our classifier training simultaneously optimizes the terminal classifier and the intermediate state classifier.

Implementation Detains. We adapted diffusion model training code for the base distributions from [229]. Our diffusion model used a UNet backbone with four latent layers on both the contracting and the expansive path. The contracting channel dimensions were $[64, 128, 256, 256]$ with the kernel size 3, and strides $[1, 2, 2, 2]$. The time embedding used a mixture of 128 sine and cosine feature pairs, with a total of 256 dimensions. These features were passed through a sigmoid activation and then expanded using a different linear head for each layer. The activations were then added to the 2D feature maps at each layer channel-wise. We used a fixed learning rate of 0.01 with Adam optimizer [120].

We adapted the classifier training code from the MNIST example in Pytorch [189]. Our binary classifier has two latent layers with channel dimensions $[32, 64]$, stride 1 and kernel width of 3. We use dropout on both layers: $p_1 = 25\%, p_2 = 50\%$. We train the network for 200 epochs on data sampled online in batches of 256 from each source model, and treat the composite model in the second step the same way. We use the Adadelta [270] optimizer with the default setting of learning rate 1.0.

Sampling. We generate samples according to Sec 5.5.3 and multiply the gradient by $\alpha = 20$.

Results. Row 3 from Figure C-5 contains mostly zeros with a few exceptions. This is in agreement with harmonic mean being symmetric. In $p_1 \otimes (p_2 \otimes p_3)$, digits that are outside of the intersection still appear with thin strokes.



(a) $p_1\{0-5\}$ (b) $p_2\{2 \cdot i\}_{i=0}^{4}$ (c) $p_3\{3 \cdot i\}_{i=0}^{3}$ (d) $p_1 \otimes p_2$ (e) $p_2 \otimes p_3$ (f) $p_1 \otimes p_3$

(g) $p_1 \text{◑} p_2$ (h) $p_2 \text{◑} p_3$ (i) $p_1 \text{◐} p_3$ (j) $p_1 \text{◐} p_2$ (k) $p_2 \text{◐} p_3$ (l) $p_1 \text{◑} p_3$

$p_1 \otimes (p_2 \otimes p_3)$ $p_2 \otimes (p_1 \otimes p_3)$ $p_3 \otimes (p_1 \otimes p_2)$

$p_1 \text{◑} (p_2 \text{◑} p_3)$ $p_1 \text{◑} (p_2 \otimes p_3)$ $p_1 \text{◑} (p_2 \text{◐} p_3)$ $p_2 \text{◑} (p_1 \text{◑} p_3)$ $p_2 \text{◑} (p_1 \otimes p_3)$ $p_2 \text{◑} (p_1 \text{◐} p_3)$

$p_3 \text{◑} (p_1 \text{◑} p_2)$ $p_3 \text{◑} (p_1 \otimes p_2)$ $p_3 \text{◑} (p_1 \text{◐} p_2)$ $p_1 \otimes (p_1 \text{◑} p_3)$ $p_1 \otimes (p_2 \text{◐} p_3)$ $p_2 \otimes (p_1 \text{◑} p_3)$

$p_2 \otimes (p_1 \text{◐} p_3)$ $p_3 \otimes (p_1 \text{◑} p_2)$ $p_3 \otimes (p_1 \text{◐} p_2)$ $p_1 \otimes (p_2 \text{◑} p_3)$ $p_1 \text{◐} (p_2 \otimes p_3)$ $p_2 \text{◐} (p_1 \otimes p_3)$

Figure C-5: **Chaining Binary Operations on Colored MNIST.** (a-c) Samples from 3 pre-trained diffusion models. (d-l) Samples from binary compositions. (row 3) The harmonic mean between all three. (row 4 and beyond) various ways to chain the operations. Parentheses indicate the order of composition.