Rethinking Generative Mode Coverage: A Pointwise Guaranteed Approach

Peilin Zhong* **Yuchen Mo*** **Chang Xiao*** **Pengyu Chen** **Changxi Zheng**

Columbia University
{peilin, chang, cxz}@cs.columbia.edu
{yuchen.mo, pengyu.chen}@columbia.edu

Abstract

Many generative models have to combat *missing modes*. The conventional wisdom to this end is by reducing through training a statistical distance (such as f-divergence) between the generated distribution and provided data distribution. But this is more of a heuristic than a guarantee. The statistical distance measures a *global*, but not *local*, similarity between two distributions. Even if it is small, it does not imply a plausible mode coverage. Rethinking this problem from a game-theoretic perspective, we show that a complete mode coverage is firmly attainable. If a generative model can approximate a data distribution moderately well under a global statistical distance measure, then we will be able to find a mixture of generators that collectively covers *every* data point and thus *every* mode, with a lower-bounded generation probability. Constructing the generator mixture has a connection to the multiplicative weights update rule, upon which we propose our algorithm. We prove that our algorithm guarantees complete mode coverage. And our experiments on real and synthetic datasets confirm better mode coverage over recent approaches, ones that also use generator mixtures but rely on global statistical distances.

1 Introduction

A major pillar of machine learning, the *generative* approach aims at learning a data distribution from a provided training dataset. While strikingly successful, many generative models suffer from *missing modes*. Even after a painstaking training process, the generated samples represent only a limited subset of the modes in the target data distribution, yielding a much lower entropy distribution.

Behind the missing mode problem is the conventional wisdom of training a generative model. Formulated as an optimization problem, the training process reduces a statistical distance between the generated distribution and the target data distribution. The statistical distance, such as f-divergence or Wasserstein distance, is often a *global* measure. It evaluates an integral of the discrepancy between two distributions over the data space (or a summation over a discrete dataset). In practice, reducing the global statistical distance to a perfect zero is virtually a mission impossible. Yet a small statistical distance does not certify the generator complete mode coverage. The generator may neglect underrepresented modes—ones that are less frequent in data space—in exchange for better matching the distribution of well represented modes, thereby lowering the statistical distance. In short, a global statistical distance is not ideal for promoting mode coverage (see Figure 1 for a 1D motivating example and later Figure 2 for examples of a few classic generative models).

This inherent limitation is evident in various types of generative models (see Appendix A for the analysis of a few classic generative models). Particularly in generative adversarial networks (GANs), mode collapse has been known as a prominent issue. Despite a number of recent improvements toward alleviating it [1, 2, 3, 4, 5, 6], none of them offers a complete mode coverage. In fact, even the

*equal contribution

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

fundamental question remains unanswered: *what precisely does a complete mode coverage mean?* After all, the definition of "modes" in a dataset is rather vague, depending on what specific distance metric is used for clustering data items (as discussed and illustrated in [4]).

We introduce an explicit notion of complete mode coverage, by switching from the global statistical distance to local *pointwise* coverage: provided a target data distribution P with a probability density $p(x)$ at each point x of the data space $\mathcal{X}$, we claim that a generator G has a *complete mode coverage* of P if the generator's probability $g(x)$ for generating x is pointwise lower bounded, that is,

$$g(x) \geq \psi \cdot p(x), \forall x \in \mathcal{X}, \tag{1}$$

for a reasonably large relaxation constant $\psi \in (0, 1)$. This notion of mode coverage ensures that *every point* x in the data space $\mathcal{X}$ will be generated by G with a finite and lower-bounded probability $g(x)$. Thereby, in contrast to the generator trained by reducing a global statistical distance (recall Figure 1), no mode will have an arbitrarily small generation probability, and thus no mode will be missed. Meanwhile, our mode coverage notion (1) stays compatible with the conventional heuristic toward reducing a global statistical distance, as the satisfaction of (1) implies that the total variation distance between P and G is upper bounded by $1 - \psi$ (see a proof in Appendix C).

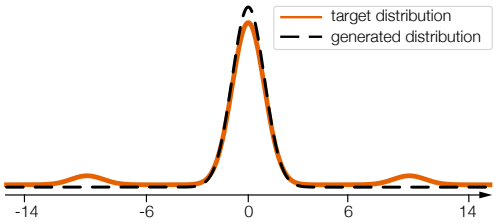

Figure 1: **Motivating example.** Consider a 1D target distribution P with three modes, i.e., a mixture of three Gaussians, $P = 0.9 \cdot \mathcal{N}(0, 1) + 0.05 \cdot \mathcal{N}(10, 1) + 0.05 \cdot \mathcal{N}(-10, 1)$ (solid orange curve). If we learn this distribution using a single Gaussian Q (black dashed curve). The statistical distance between the two is small: $D_{\text{TV}}(Q \parallel P) \leq 0.1$ and $D_{\text{KL}}(Q \parallel P) \leq 0.16$. The probability of drawing samples from the side modes (in $[-14, -6]$ and $[6, 14]$) of the target distribution P is $\Pr_{x\sim P}[6 \leq |x| \leq 14] \approx 0.1$, but the probability of generating samples from Q in the same intervals is $\Pr_{x\sim Q}[6 \leq |x| \leq 14] \approx 10^{-9}$. The side modes are missed!

At first sight, the pointwise condition (1) seems more stringent than reducing a global statistical distance, and pursuing it might require a new formulation of generative models. Perhaps somewhat surprisingly, a rethink from a game-theoretic perspective reveal that this notion of mode coverage is viable without formulating any new models. Indeed, a mixture of existing generative models (such as GANs) suffices. In this work, we provide an algorithm for constructing the generator mixture and a theoretical analysis showing the guarantee of our mode coverage notion (1).

1.1 A Game-Theoretic Analysis

Before delving into our algorithm, we offer an intuitive view of *why our mode coverage notion* (1) *is attainable* through a game-theoretic lens. Consider a two-player game between Alice and Bob: given a target data distribution P and a family $\mathcal{G}$ of generators[2], Alice chooses a generator $G \in \mathcal{G}$, and Bob chooses a data point $x \in \mathcal{X}$. If the probability density $g(x)$ of Alice's G generating Bob's choice of x satisfies $g(x) \geq \frac{1}{4}p(x)$, the game produces a value $v(G, x) = 1$, otherwise it produces $v(G, x) = 0$. Here $1/4$ is used purposely as an example to concretize our intuition. Alice's goal is to maximize the game value, while Bob's goal is to minimize the game value.

Now, consider two situations. In the first situation, Bob first chooses a mixed strategy, that is, a distribution Q over $\mathcal{X}$. Then, Alice chooses the best generator $G \in \mathcal{G}$ according to Bob's distribution Q. When the game starts, Bob samples a point x using his choice of distribution Q. Together with Alice's choice G, the game produces a value. Since x is now a random variable over Q, the expected game value is $\max_{G\in\mathcal{G}} \mathop{\mathbb{E}}_{x\sim Q}[v(G, x)]$. In the second situation, Alice first chooses a mixed strategy, that is, a distribution $R_{\mathcal{G}}$ of generators over $\mathcal{G}$. Then, given Alice's choice $R_{\mathcal{G}}$, Bob chooses the best data point $x \in \mathcal{X}$. When the game starts, Alice samples a generator G from the chosen distribution $R_{\mathcal{G}}$. Together with Bob's choice of x, the game produces a value, and the expected value is $\min_{x\in\mathcal{X}} \mathbb{E}_{G\sim R_{\mathcal{G}}}[v(G, x)]$.

According to von Neumann's minimax theorem [7, 8], Bob's optimal expected value in the first situation must be the same as Alice's optimal value in the second situation:

$$\min_{Q} \max_{G\in\mathcal{G}} \mathop{\mathbb{E}}_{x\sim Q}[v(G, x)] = \max_{R_{\mathcal{G}}} \min_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{G\sim R_{\mathcal{G}}}[v(G, x)]. \tag{2}$$

[2] An example of the generator family is the GANs. The definition will be made clear later in this paper.

With this equality realized, our agenda in the rest of the analysis is as follows. First, we show a lower bound of the left-hand side of (2), and then we use the right-hand side to reach the lower-bound of $g(x)$ as in (1), for Alice's generator G. To this end, we need to depart off from the current game-theoretic analysis and discuss the properties of existing generative models for a moment.

Existing generative models such as GANs [9, 1, 10] aim to reproduce arbitrary data distributions. While it remains intractable to have the generated distribution match *exactly* the data distribution, the approximations are often plausible. One reason behind the plausible performance is that the data space encountered in practice is "natural" and restricted—all English sentences or all natural object images or all images on a manifold—but not a space of arbitrary data. Therefore, it is reasonable to expect the generators in $\mathcal{G}$ (e.g., all GANs) to meet the following requirement[3] (without conflicting the no-free-lunch theorem [11]): for any distribution Q over a natural data space $\mathcal{X}$ encountered in practice, there exists a generator $G \in \mathcal{G}$ such that the total variation distance between G and Q is upper bounded by a constant γ, that is, $\frac{1}{2}\int_{\mathcal{X}} |q(x) - g(x)| \, \mathrm{d}x \leq \gamma$, where $q(\cdot)$ and $g(\cdot)$ are the probability densities on Q and the generated samples of G, respectively. Again as a concrete example, we use $\gamma = 0.1$. With this property in mind, we now go back to our game-theoretic analysis.

Back to the first situation described above. Once Bob's distribution Q (over $\mathcal{X}$) and Alice's generator G are identified, then given a target distribution P over $\mathcal{X}$ and an x drawn by Bob from Q, the probability of having Alice's G cover P (i.e., $g(x) \geq \frac{1}{4}p(x)$) at x is lower bounded. In our current example, we have the following lower bound:

$$\Pr_{x \sim Q}[g(x) \geq 1/4 \cdot p(x)] \geq 0.4. \tag{3}$$

Here 0.4 is related to the total variation distance bound (i.e., $\gamma = 0.1$) between G and Q, and this lower bound value is derived in Appendix D. Next, notice that on the left-hand side of (2), the expected value, $\mathbb{E}_{x \sim Q}[v(G, x)]$, is equivalent to the probability in (3). Thus, we have

$$\min_{Q} \max_{G \in \mathcal{G}} \mathop{\mathbb{E}}_{x \sim Q}[v(G, x)] \geq 0.4. \tag{4}$$

Because of the equality in (2), this is also the lower bound of its right-hand side, from which we know that there exists a distribution $R_{\mathcal{G}}$ of generators such that for any $x \in \mathcal{X}$, we have

$$\mathop{\mathbb{E}}_{G \sim R_{\mathcal{G}}}[v(G, x)] = \Pr_{G \sim R_{\mathcal{G}}}[g(x) \geq 1/4 \cdot p(x)] \geq 0.4. \tag{5}$$

This expression shows that for any $x \in \mathcal{X}$, if we draw a generator G from $R_{\mathcal{G}}$, then with a probability at least 0.4, G's generation probability density satisfies $g(x) \geq \frac{1}{4}p(x)$. Thus, we can think $R_{\mathcal{G}}$ as a "collective" generator $\boldsymbol{G}^*$, or a *mixture of generators*. When generating a sample x, we first choose a generator G according to $R_{\mathcal{G}}$ and then sample an x using G. The overall probability $g^*(x)$ of generating x satisfies $g^*(x) > 0.1p(x)$—precisely the pointwise lower bound that we pose in (1).

Takeaway from the analysis. This analysis reveals that a complete mode coverage is firmly viable. Yet it offers no recipe on *how* to construct the mixture of generators and their distribution $R_{\mathcal{G}}$ using existing generative models. Interestingly, as pointed out by Arora et al. [12], a constructive version of von Neumann's minimax theorem is related to the general idea of multiplicative weights update. Therefore, our key contributions in this work are **i)** the design of a multiplicative weights update algorithm (in Sec. 3) to construct a generator mixture, and **ii)** a theoretical analysis showing that our generator mixture indeed obtains the pointwise data coverage (1). In fact, we only need a small number of generators to construct the mixture (i.e., it is easy to train), and the distribution $R_{\mathcal{G}}$ for using the mixture is as simple as a uniform distribution (i.e., it is easy to use).

2 Related Work

There exists a rich set of works improving classic generative models for alleviating missing modes, especially in the framework of GANs, by altering objective functions [13, 14, 15, 10, 16, 17], changing training methods [18, 19], modifying neural network architectures [2, 20, 21, 22, 23], or regularizing latent space distributions [4, 24]. The general philosophy behind these improvements is to reduce the statistical distance between the generated distribution and target distribution by making

[3] This requirement is weaker than the mainstream goal of generative models, which all aim to approximate a target data distribution as closely as possible. Here we only require the approximation error is upper bounded.

the models easier to train. Despite their technical differences, their optimization goals are all toward reducing a global statistical distance.

The idea of constructing a mixture of generators has been explored, with two ways of construction. In the first way, a set of generators are trained simultaneously. For example, Locatello et al. [25] used multiple generators, each responsible for sampling a subset of data points decided in a k-means clustering fashion. Other methods focus on the use of multiple GANs [26, 27, 28]. The theoretical intuition behind these approaches is by viewing a GAN as a two-player game and extending it to reach a Nash equilibrium with a mixture of generators [26]. In contrast, our method does not depend specifically on GANs, and our game-theoretic view is fundamentally different (recall Sec. 1.1).

Another way of training a mixture of generators takes a sequential approach. This is related to *boosting* algorithms in machine learning. Grnarova et al. [29] viewed the problem of training GANs as finding a mixed strategy in a zero-sum game, and used the Follow-the-Regularized-Leader algorithm [30] for training a mixture of generators iteratively. Inspired by AdaBoost [31], other approaches train a "weak" generator that fits a reweighted data distribution in each iteration, and all iterations together form an additive mixture of generators [32, 33] or a multiplicative mixture of generators [34].

Our method can be also viewed as a boosting strategy. From this perspective, the most related is AdaGAN [33], while significant differences exist. Theoretically, AdaGAN (and other boosting-like algorithms) is based on the assumption that the reweighted data distribution in each iteration becomes progressively easier to learn. It requires a generator in each iteration to have a statistical distance to the reweighted distribution smaller than the previous iteration. As we will discuss in Sec. 5, this assumption is not always feasible. We have no such assumption. Our method can use a weak generator in each iteration. If the generator is more expressive, the theoretical lower bound of our pointwise coverage becomes larger (i.e., a larger ψ in (1)). Algorithmically, our reweighting scheme is simple and different from AdaGAN, only doubling the weights or leaving them unchanged in each iteration. Also, in our mixture of generators, they are treated uniformly, and no mixture weights are needed, whereas AdaGAN needs a set of weights that are heuristically chosen.

To summarize, in stark contrast to all prior methods, our approach is rooted in a different philosophy of training generative models. Rather than striving for reducing a global statistical distance, our method revolves around an explicit notion of complete mode coverage as defined in (1). Unlike other boosting algorithms, our algorithm of constructing the mixture of generators guarantees complete mode coverage, and this guarantee is theoretically proved.

3 Algorithm

A mixture of generators. Provided a target distribution P on a data domain $\mathcal{X}$, we train a mixture of generators to pursue pointwise mode coverage (1). Let $\boldsymbol{G}^* = \{G_1, \ldots, G_T\}$ denote the resulting mixture of T generators. Each of them ($G_t, t = 1...T$) may use any existing generative model such as GANs. Existing methods that also rely on a mixture of generators associate each generator a nonuniform weight α_t and choose a generator for producing a sample randomly based on the weights. Often, these weights are chosen heuristically, e.g., in AdaGAN [33]. Our mixture is conceptually and computationally simpler. Each generator is treated equally. When using $\boldsymbol{G}^*$ to generate a sample, we first choose a generator G_i uniformly at random, and then use G_i to generate the sample.

Algorithm overview. Our algorithm of training $\boldsymbol{G}^*$ can be understood as a specific rule design in the framework of multiplicative weights update [12]. Outlined in Algorithm 1, it runs iteratively. In each iteration, a generator G_t is trained using an updated data distribution P_t (see Line 6-7 of Algorithm 1). The intuition here is simple: if in certain data domain regions the current generator fails to cover the target distribution sufficiently well, then we update the data distribution to emphasize those regions for the next round of generator training (see Line 9 of Algorithm 1). In this way, each generator can focus on the data distribution in individual data regions. Collectively, they are able to cover the distribution over the entire data domain, and thus guarantee pointwise data coverage.

Training. Each iteration of our algorithm trains an individual generator G_t, for which many existing generative models, such as GANs [9], can be used. The only prerequisite is that G_t needs to be trained to approximate the data distribution P_t moderately well. This requirement arises from our game-theoretic analysis (Sec. 1.1), wherein the total variation distance between G_t's distribution and P_t needs to be upper bounded. Later in our theoretical analysis (Sec. 4), we will formally state this requirement, which, in practice, is easily satisfied by most existing generative models.

Algorithm 1 Constructing a mixture of generators

1: **Parameters:** T, a positive integer number of generators, and $\delta \in (0, 1)$, a covering threshold.
2: **Input:** a target distribution P on a data domain $\mathcal{X}$.
3: For each $x \in \mathcal{X}$, initialize its weight $w_1(x) = p(x)$.
4: **for** $t = 1 \to T$ **do**
5: Construct a distribution P_t over $\mathcal{X}$ as follows:
6: For every $x \in \mathcal{X}$, normalize the probability density $p_t(x) = \frac{w_t(x)}{W_t}$, where $W_t = \int_{\mathcal{X}} w_t(x)\mathrm{d}x$.
7: Train a generative model G_t on the distribution P_t.
8: Estimate generated density $g_t(x)$ for every $x \in \mathcal{X}$.
9: For each $x \in \mathcal{X}$, if $g_t(x) < \delta \cdot p(x)$, set $w_{t+1}(x) = 2 \cdot w_t(x)$. Otherwise, set $w_{t+1}(x) = w_t(x)$.
10: **end for**
11: **Output:** a mixture of generators $\boldsymbol{G}^* = \{G_1, \ldots, G_T\}$.

Estimation of generated probability density. In Line 8 of Algorithm 1, we need to estimate the probability $g_t(x)$ of the current generator sampling a data point x. Our estimation follows the idea of adversarial training, similar to AdaGAN [33]. First, we train a discriminator D_t to distinguish between samples from P_t and samples from G_t. The optimization objective of D_t is defined as

$$\max_{D_t} \mathop{\mathbb{E}}_{x \sim P_t} [\log D_t(x)] + \mathop{\mathbb{E}}_{x \sim G_t} [\log(1 - D_t(x))].$$

Unlike AdaGAN [33], here P_t is the currently updated data distribution, not the original target distribution, and G_t is the generator trained in the current round, not a mixture of generators in all past rounds. As pointed out previously [35, 33], once D_t is optimized, we have $D_t(x) = \frac{p_t(x)}{p_t(x)+g_t(x)}$ for all $x \in \mathcal{X}$, and equivalently $\frac{g_t(x)}{p_t(x)} = \frac{1}{D_t(x)} - 1$. Using this property in Line 9 of Algorithm 1 (for testing the data coverage), we rewrite the condition $g_t(x) < \delta \cdot p(x)$ as

$$\frac{g_t(x)}{p(x)} = \frac{g_t(x)}{p_t(x)} \frac{p_t(x)}{p(x)} = \left(\frac{1}{D_t(x)} - 1 \right) \frac{w_t(x)}{p(x) W_t} < \delta,$$

where the second equality utilize the evaluation of $p_t(x)$ in Line 6 (i.e., $p_t(x) = {w_t(x)}/{W_t}$).

Note that if the generators G_t are GANs, then the discriminator of each G_t can be reused as D_t here. Reusing D_t introduces no additional computation. In contrast, AdaGAN [33] always has to train an additional discriminator D_t in each round using the mixture of generators of all past rounds.

Working with empirical dataset. In practice, the true data distribution P is often unknown when an empirical dataset $\mathbb{X} = \{x_i\}_{i=1}^n$ is given. Instead, the empirical dataset is considered as n i.i.d. samples drawn from P. According to the Glivenko-Cantelli theorem [36], the uniform distribution over n i.i.d. samples from P will converge to P as n approaches to infinity. Therefore, provided the empirical dataset, we do not need to know the probability density $p(x)$ of P, as every sample $x_i \in \mathbb{X}$ is considered to have a finite and uniform probability measure. An empirical version of Algorithm 1 and more explanation are presented in the supplementary document (Algorithm 2 and Appendix B).

4 Theoretical Analysis

We now provide a theoretical understanding of our algorithm, showing that the pointwise data coverage (1) is indeed obtained. Our analysis also sheds some light on how to choose the parameters of Algorithm 1.

4.1 Preliminaries

We first clarify a few notational conventions and introduce two new theoretical notions for our subsequent analysis. Our analysis is in continuous setting; results on discrete datasets follow directly.

Notation. Formally, we consider a d-dimensional measurable space $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$, where $\mathcal{X}$ is the d-dimensional data space, and $\mathcal{B}(\mathcal{X})$ is the Borel σ-algebra over $\mathcal{X}$ to enable probability measure. We use a capital letter (e.g., P) to denote a probability measure on this space. When there is no ambiguity, we also refer them as probability distributions (or distributions). For any subset $\mathcal{S} \in \mathcal{B}(\mathcal{X})$, the probability of $\mathcal{S}$ under P is $P(\mathcal{S}) \coloneqq \Pr_{x \sim P}[x \in \mathcal{S}]$. We use G to denote a generator. When there is

no ambiguity, G also denotes the distribution of its generated samples. All distributions are assumed absolutely continuous. Their probability density functions (i.e., the derivative with respect to the Lebesgue measure) are referred by their corresponding lowercase letters (e.g., $p(\cdot)$, $q(\cdot)$, and $g(\cdot)$).

Moreover, we use $[n]$ to denote the set $\{1, 2, ..., n\}$, $\mathbb{N}_{>0}$ for the set of all positive integers, and $\mathbb{1}(\mathcal{E})$ for the indicator function whose value is 1 if the event $\mathcal{E}$ happens, and 0 otherwise.

f-divergence. Widely used in objective functions of training generative models, f-divergence is a statistical distance between two distributions. Let P and Q be two distributions over $\mathcal{X}$. Provided a convex function f on $(0, \infty)$ such that $f(1) = 0$, f-divergence of Q from P is defined as $D_f(Q \parallel P) \coloneqq \int_{\mathcal{X}} f\left(\frac{q(x)}{p(x)}\right) p(x) \mathrm{d}x$. Various choices of f lead to some commonly used f-divergence metrics such as total variation distance D_{TV}, Kullback-Leibler divergence D_{KL}, Hellinger distance D_{H}, and Jensen-Shannon divergence D_{JS} [35, 37]. Among them, total variation distance is upper bounded by many other f-divergences. For instance, $D_{\mathrm{TV}}(Q \parallel P)$ is upper bounded by $\sqrt{\frac{1}{2} D_{\mathrm{KL}}(Q \parallel P)}$, $\sqrt{2 D_{\mathrm{H}}(Q \parallel P)}$, and $\sqrt{2 D_{\mathrm{JS}}(Q \parallel P)}$, respectively. Thus, if two distributions are close under those f-divergence measures, so are they under total variation distance. For this reason, our theoretical analysis is based on the total variation distance.

δ-cover and (δ, β)-cover. We introduce two new notions for analyzing our algorithm. The first is the notion of *δ-cover*. Given a data distribution P over $\mathcal{X}$ and a value $\delta \in (0, 1]$, if a generator G satisfies $g(x) \geq \delta \cdot p(x)$ at a data point $x \in \mathcal{X}$, we say that x is δ-covered by G under distribution P. Using this notion, the pointwise mode coverage (1) states that x is ψ-covered by G under distribution P for all $x \in \mathcal{X}$. We also extend this notion to a measurable subset $\mathcal{S} \in \mathcal{B}(\mathcal{X})$: we say that $\mathcal{S}$ is δ-covered by G under distribution P if $G(\mathcal{S}) \geq \delta \cdot P(\mathcal{S})$ is satisfied.

Next, consider another distribution Q over $\mathcal{X}$. We say that G can *(δ, β)-cover* (P, Q), if the following condition holds:

$$\Pr_{x \sim Q}[x \text{ is } \delta\text{-covered by } G \text{ under distribution} P] \geq \beta. \tag{6}$$

For instance, using this notation, Equation (3) in our game-theoretic analysis states that G can $(0.25, 0.4)$-cover (P, Q).

4.2 Guarantee of Pointwise Data Coverage

In each iteration of Algorithm 1, we expect the generator G_t to approximate the given data distribution P_t sufficiently well. We now formalize this expectation and understand its implication. Our intuition is that by finding a property similar to (3), we should be able to establish a pointwise coverage lower bound in a way similar to our analysis in Sec. 1.1. Such a property is given by the following lemma (and proved in Appendix E.1).

Lemma 1. *Consider two distributions, P and Q, over the data space $\mathcal{X}$, and a generator G producing samples in $\mathcal{X}$. For any $\delta, \gamma \in (0, 1]$, if $D_{TV}(G \parallel Q) \leq \gamma$, then G can $(\delta, 1 - 2\delta - \gamma)$-cover (P, Q).*

Intuitively, when G and Q are identified, γ is set. If δ is reduced, then more data points in $\mathcal{X}$ can be δ-covered by G under P. Thus, the probability defined in (6) becomes larger, as reflected by the increasing $1 - 2\delta - \gamma$. On the other hand, consider a fixed δ. As the discrepancy between G and Q becomes larger, γ increases. Then, sampling an x according to Q will have a smaller chance to land at a point that is δ-covered by G under P, as reflected by the decreasing $1 - 2\delta - \gamma$.

Next, we consider Algorithm 1 and identify a sufficient condition under which the output mixture of generators $\boldsymbol{G}^*$ covers every data point with a lower-bounded guarantee (i.e., our goal (1)). Simply speaking, this sufficient condition is as follows: in each round t, the generator G_t is trained such that given an x drawn from distribution P_t, the probability of x being δ-covered by G_t under P is also lower bounded. A formal statement is given in the next lemma (proved in Appendix E.2).

Lemma 2. *Recall that $T \in \mathbb{N}_{>0}$ and $\delta \in (0, 1)$ are the input parameters of Algorithm 1. For any $\varepsilon \in [0, 1)$ and any measurable subset $\mathcal{S} \in \mathcal{B}(\mathcal{X})$ whose probability measure satisfies $P(\mathcal{S}) \geq 1/2^{\eta T}$ with some $\eta \in (0, 1)$, if in every round $t \in [T]$, G_t can $(\delta, 1 - \varepsilon)$-cover (P, P_t), then the resulting mixture of generators $\boldsymbol{G}^*$ can $(1 - \varepsilon/\ln 2 - \eta)\delta$-cover $\mathcal{S}$ under distribution P.*

This lemma is about lower-bounded coverage of a measurable subset $\mathcal{S}$, not a point $x \in \mathcal{X}$. At first sight, it is not of the exact form in (1) (i.e., pointwise δ-coverage). This is because formally speaking it makes no sense to talk about covering probability at a single point (whose measure is zero). But as

T approaches to ∞, $\mathcal{S}$ that satisfies $P(\mathcal{S}) \geq 1/2^{\eta T}$ can also approach to a point (and η approaches to zero). Thus, Lemma 2 provides a condition for pointwise lower-bounded coverage in the limiting sense. In practice, the provided dataset is always discrete, and the probability measure at each discrete data point is finite. Then, Lemma 2 is indeed a sufficient condition for pointwise lower-bounded coverage.

From Lemma 1, we see that the condition posed by Lemma 2 is indeed satisfied by our algorithm, and combing both lemmas yields our final theorem (proved in Appendix E.3).

Theorem 1. *Recall that $T \in \mathbb{N}_{>0}$ and $\delta \in (0,1)$ are the input parameters of Algorithm 1. For any measurable subset $\mathcal{S} \in \mathcal{B}(\mathcal{X})$ whose probability measure satisfies $P(\mathcal{S}) \geq 1/2^{\eta T}$ with some $\eta \in (0,1)$, if in every round $t \in [T]$, $D_{TV}(G_t \parallel P_t) \leq \gamma$, then the resulting mixture of generators $\boldsymbol{G}^*$ can $(1-(\gamma+2\delta)/\ln 2-\eta)\delta$-cover $\mathcal{S}$ under distribution P.*

In practice, existing generative models (such as GANs) can approximate P_t sufficiently well, and thus $D_{\text{TV}}(G_t \parallel P_t) \leq \gamma$ is always satisfied for some γ. According to Theorem 1, a pointwise lower-bounded coverage can be obtained by our Algorithm 1. If we choose to use a more expressive generative model (e.g., a GAN with a stronger network architecture), then G_t can better fit P_t in each round, yielding a smaller γ used in Theorem 1. Consequently, the pointwise lower bound of the data coverage becomes larger, and effectively the coefficient ψ in (1) becomes larger.

4.3 Insights from the Analysis

γ, η, δ, and T in Theorem 1. In Theorem 1, γ depends on the expressive power of the generators being used. It is therefore determined once the generator class $\mathcal{G}$ is chosen. But η can be directly set by the user and a smaller η demands a larger T to ensure $P(\mathcal{S}) \geq 1/2^{\eta T}$ is satisfied. Once γ and η is determined, we can choose the best δ by maximizing the coverage bound (i.e., $(1-(\gamma+2\delta)/\ln 2-\eta)\delta$) in Theorem 1. For example, if $\gamma \leq 0.1, \eta \leq 0.01$, then $\delta \approx 1/4$ would optimize the coverage bound (see Appendix E.4 for more details), and in this case the coefficient ψ in (1) is at least $1/30$.

Theorem 1 also sets the tone for the training cost. As explained in Appendix E.4, given a training dataset of size n, the size of the generator mixture, T, needs to be at most $O(\log n)$. This theoretical bound is consistent with our experimental results presented in Sec. 5. In practice, only a small number of generators are needed.

Estimated density function g_t. The analysis in Sec. 4.2 assumes that the generated probability density g_t of the generator G_t in each round is known, while in practice we have to estimate g_t by training a discriminator D_t (recall Section 3). Fortunately, only mild assumptions in terms of the quality of D_t are needed to retain the pointwise lower-bounded coverage. Roughly speaking, D_t needs to meet two conditions: 1) In each round t, only a fraction of the covered data points (i.e., those with $g_t(x) \geq \delta \cdot p(x)$) is falsely classified by D_t and doubled their weights. 2) In each round t, if the weight of a data point x is not doubled based on the estimation of $D_t(x)$, then there is a good chance that x is truly covered by G_t (i.e., $g_t(x) \geq \delta \cdot p(x)$). A detailed and formal discussion is presented in Appendix E.5. In short, our estimation of g_t would not deteriorate the efficacy of the algorithm, as also confirmed in our experiments.

Generalization. An intriguing question for all generative models is their *generalization* performance: how well can a generator trained on an empirical distribution (with a finite number of data samples) generate samples that follow the true data distribution? While the generalization performance has been long studied for supervised classification, generalization of generative models remains a widely open theoretical question. We propose a notion of generalization for our method, and provide a preliminary theoretical analysis. All the details are presented in Appendix E.6.

5 Experiments

We now present our major experimental results, while referring to Appendix F for network details and more results. We show that our mixture of generators is able to cover all the modes in various synthetic and real datasets, while existing methods always have some modes missed.

Previous works on generative models used the Inception Score [1] or the Fréchet Inception Distance [18] as their evaluation metric. But we do not use them, because they are both global measures, not reflecting mode coverage in local regions [38]. Moreover, these metrics are designed to measure the quality of generated images, which is orthogonal to our goal. For example, one can always use a more expressive GAN in each iteration of our algorithm to obtain better image quality and thus better inception scores.

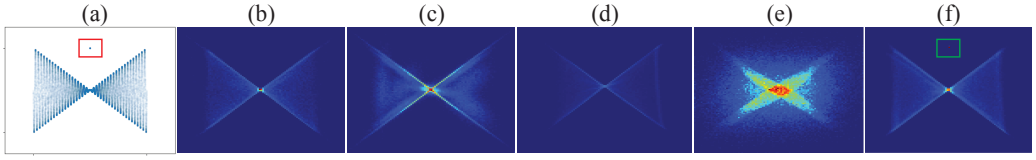

Figure 2: **Generative models on synthetic dataset.** (a) The dataset consists of two modes: one major mode as an expanding sine curve ($y = x \sin \frac{4x}{\pi}$) and a minor mode as a Gaussian located at $(10, 0)$ (highlighted in the reb box). (b-f) We show color-coded distributions of generated samples from (b) EM, (c) GAN, (d) VAE, (e) AdaGAN, and (f) our method (i.e., a mixture of GANs). Only our method is able to cover the second mode (highlighted in the green box; zoomin to view).

Since the phenomenon of missing modes is particularly prominent in GANs, our experiments emphasize on the mode coverage performance of GANs and compare our method (using a mixture of GANs) with DCGAN [39], MGAN [27], and AdaGAN. The latter two also use multiple GANs to improve mode coverage, although they do *not* aim for the same mode coverage notion as ours.

Overview. We first outline all our experiments, including those presented in Appendix F. **i)** We compare our method with a number of classic generative models on a synthetic dataset. **ii)** In Appendix F.3, we also compare our method with AdaGAN [33] on other synthetic datasets as well as stacked MNIST dataset, because both are boosting algorithms aiming at improving mode coverage. **iii)** We further compare our method with a single large DCGAN, AdaGAN, and MGAN on the Fashion-MNIST dataset [40] mixed with a very small portion of MNIST dataset [41].

Various generative models on synthetic dataset. As we show in Appendix A, many generative models, such as expectation-maximization (EM) methods, VAEs, and GANs, all rely on a global statistical distance in their training. We therefore test their mode coverage and compare with ours. We construct on $\mathbb{R}^2$ a synthetic dataset with two modes. The first mode consists of data points whose x-coordinate is uniformly sampled by $x_i \sim [-10, 10]$ and the y-coordinate is $y_i = x_i \sin \frac{4x_i}{\pi}$. The second mode has data points forming a Gaussian at $(0, 10)$. The total number of data points in the first mode is 400$\times$ of the second. As shown in Figure 2, generative models include EM, GAN, VAE, and AdaGAN [33] all fail to cover the second mode. Our method, in contrast, captures both modes. We run KDE to estimate the likelihood of our generated samples on our synthetic data experiments (using KDE bandwidth=0.1). We compute $L = 1/N \sum_i P_{model}(x_i)$, where x_i is a sample in the minor mode. For the minor mode, our method has a mean log likelihood of -1.28, while AdaGAN has only -967.64 (almost no samples from AdaGAN).

Fashion-MNIST and partial MNIST. Our next experiment is to challenge different GAN models with a real dataset that has separated and unbalanced modes. This dataset consists of the entire training dataset of Fashion-MNIST (with 60k images) mixed with randomly sampled 100 MNIST images labeled as "1". The size of generator mixture is always set to be 30 for AdaGAN, MGAN and our method, and all generators share the same network structure. Additionally, when comparing with a *single* DCGAN, we ensure that the DCGAN's total number of parameters is comparable to the total number of parameters of the 30 generators in AdaGAN, MGAN, and ours.

	"1"s	Frequency	Avg Prob.
DCGAN	13	0.14×10^{-4}	0.49
MGAN	collapsed	-	-
AdaGAN	60	0.67×10^{-4}	0.45
Our method	289	3.2×10^{-4}	0.68

Table 1: **Ratios of generated images classified as "1".** We generate 9×10^5 images from each method. The second column indicates the numbers of samples being classified as "1", and the third column indicates the ratio. In the fourth column, we average the prediction probabilities over all generated images that are classified as "1".

To evaluate the results, we train an 11-class classifier to distinguish the 10 classes in Fashion-MNIST and one class in MNIST (i.e., "1"). First, we check how many samples from each method are classified as "1". The test setup and results are shown in Table 1 and its caption. The results suggest that our method can generate more "1" samples with higher prediction confidence. Note that MGAN has a strong mode collapse and fails to produce "1" samples. While DCGAN and AdaGAN generate some samples that are classified as "1", inspecting the generated images reveals that those samples are all visually far from "1"s, but incorrectly classified by the pre-trained classifier (see Figure 3). In contrast, our method is able to generate samples close to "1". We also note that our method can produce higher-quality images if the underlying generative models in each round become stronger.

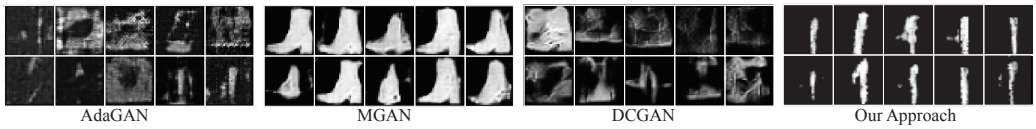

Figure 3: **Most confident "1" samples.** Here we show samples that are generated by each tested methods and also classified by the pre-trained classifier most confidently as "1" images (i.e., top 10 in terms of the classified probability). Samples of our method are visually much closer to "1".

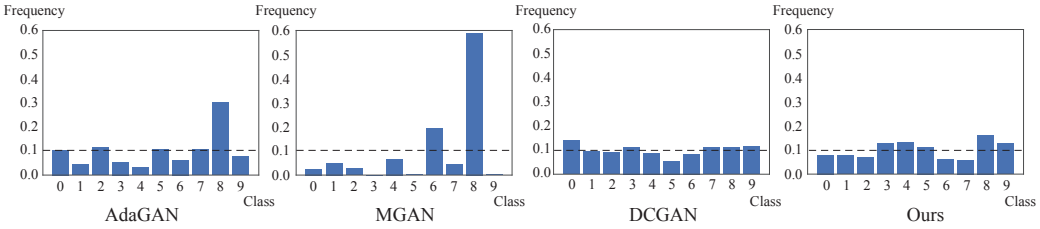

Figure 5: **Distribution of generated samples.** Training samples are drawn uniformly from each class. But generated samples by AdaGAN and MGAN are considerably nonuniform, while those from DCGAN and our method are more uniform. This experiment suggests that the conventional heuristic of reducing a statistical distance might not merit its use in training generative models.

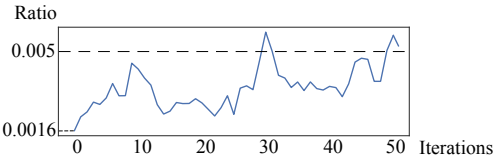

Figure 4: **Weight ratio of "1"s.** We calculate the ratio of the total weights of training images labeled by "1" to the total weights of all training images in each round, and plot here how the ratio changes with respect to the iterations in our algorithm.

Another remarkable feature is observed in our algorithm. In each round of our training algorithm, we calculate the total weight $\bar{w}_t$ of provided training samples classified as "1" as well as the total weight W_t of all training samples. When plotting the ratio $\bar{w}_t/W_t$ changing with respect to the number of rounds (Figure 4), interestingly, we found that this ratio has a maximum value at around 0.005 in this example. We conjecture that in the training dataset if the ratio of "1" images among all training images is around $1/200$, then a single generator may learn and generate "1" images (the minority mode). To verify this conjecture, we trained a GAN (with the same network structure) on another training dataset with 60k training images from Fashion-MNIST mixed with 300 MNIST "1" images. We then use the trained generator to sample 100k images. As a result, In a fraction of 4.2×10^{-4}, those images are classified as "1". Figure 8 in Appendix F shows some of those images. This result confirms our conjecture and suggests that $\bar{w}_t/W_t$ may be used as a measure of *mode bias* in a dataset.

Lastly, in Figure 5, we show the generated distribution over the 10 Fashion-MNIST classes from each tested method. We neglect the class "1", as MGAN fails to generate them. The generated samples of AdaGAN and MGAN is highly nonuniform, though in the training dataset, the 10 classes of images are uniformly distributed. Our method and DCGAN produce more uniform samples. This suggests that although other generative models (such as AdaGAN and MGAN) aim to reduce a global statistical distance, the generated samples may not easily match the empirical distribution—in this case, a uniform distribution. Our method, while not aiming for reducing the statistical distance in the first place, matches the target empirical distribution plausibly, as a byproduct.

6 Conclusion

We have presented an algorithm that iteratively trains a mixture of generators, driven by an explicit notion of complete mode coverage. With this notion for designing generative models, our work poses an alternative goal, one that differs from the conventional training philosophy: instead of reducing a global statistical distance between the target distribution and generated distribution, one only needs to make the distance mildly small but not have to reduce it toward a perfect zero, and our method is able to boost the generative model with theoretically guaranteed mode coverage.

Acknowledgments. This work was supported in part by the National Science Foundation (CAREER-1453101, 1816041, 1910839, 1703925, 1421161, 1714818, 1617955, 1740833), Simons Foundation (#491119 to Alexandr Andoni), Google Research Award, a Google PhD Fellowship, a Snap Research Fellowship, a Columbia SEAS CKGSB Fellowship, and SoftBank Group.

References

[1] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[2] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

[3] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[4] Chang Xiao, Peilin Zhong, and Changxi Zheng. Bourgan: Generative networks with metric embeddings. In *Advances in Neural Information Processing Systems*, pages 2269–2280, 2018.

[5] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[6] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, 2016.

[7] J v Neumann. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320, 1928.

[8] Ding-Zhu Du and Panos M Pardalos. *Minimax and applications*, volume 4. Springer Science & Business Media, 2013.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[10] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[11] David H Wolpert, William G Macready, et al. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[12] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[13] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016.

[14] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[15] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2813–2821. IEEE, 2017.

[16] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[17] Yunus Saatci and Andrew G Wilson. Bayesian gan. In *Advances in neural information processing systems*, pages 3622–3631, 2017.

[18] 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.

[19] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[20] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[21] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. *arXiv preprint arXiv:1712.04086*, 2017.

[22] Akash Srivastava, Lazar Valkoz, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3310–3320, 2017.

[23] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[24] Chongxuan Li, Max Welling, Jun Zhu, and Bo Zhang. Graphical generative adversarial networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6072–6083. Curran Associates, Inc., 2018.

[25] Francesco Locatello, Damien Vincent, Ilya Tolstikhin, Gunnar Rätsch, Sylvain Gelly, and Bernhard Schölkopf. Clustering meets implicit generative models. *arXiv preprint arXiv:1804.11130*, 2018.

[26] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). *arXiv preprint arXiv:1703.00573*, 2017.

[27] 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.

[28] David Keetae Park, Seungjoo Yoo, Hyojin Bahng, Jaegul Choo, and Noseong Park. Megan: Mixture of experts of generative adversarial networks for multimodal image generation. *arXiv preprint arXiv:1805.02481*, 2018.

[29] Paulina Grnarova, Kfir Y Levy, Aurelien Lucchi, Thomas Hofmann, and Andreas Krause. An online learning approach to generative adversarial networks. *arXiv preprint arXiv:1706.03269*, 2017.

[30] Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[31] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[32] Yaxing Wang, Lichao Zhang, and Joost van de Weijer. Ensembles of generative adversarial networks. *arXiv preprint arXiv:1612.00991*, 2016.

[33] Ilya O Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. Adagan: Boosting generative models. In *Advances in Neural Information Processing Systems*, pages 5430–5439, 2017.

[34] Aditya Grover and Stefano Ermon. Boosted generative models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[35] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.

[36] Francesco Paolo Cantelli. Sulla determinazione empirica delle leggi di probabilita. *Giorn. Ist. Ital. Attuari*, 4(421-424), 1933.

[37] Shun-ichi Amari. *Information geometry and its applications*. Springer, 2016.

[38] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[39] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[40] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[41] 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.

[42] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[43] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Supplementary Document
Rethinking Generative Mode Coverage: A Pointwise Guaranteed Approach

A Global Statistic Distance Based Generative Approaches

In this section, we analyze a few classic generative models to show their connections to the reduction of a certain global statistical distance. The reliance on global statistical distances explains why they suffer from missing modes, as empirically confirmed in Figure 2 of the main text.

Maximum Likelihood Estimation. Consider a target distribution P with density function $p(\cdot)$. Suppose we are provided with n i.i.d. samples $\{x_1, x_2, \cdots, x_n\}$ drawn from P. The goal of training a generator through maximum likelihood estimation (MLE) is to find from a predefined generator family $\mathcal{G}$ the generator G that maximize

$$L(G) = \frac{1}{n}\sum_i \log g(x_i),$$

where $g(\cdot)$ is the probability density function of the distribution generated by G. When n approaches ∞, the MLE objective amount to

$$\lim_{n\to\infty}\left(\max_{G\in\mathcal{G}} L(G)\right) = \max_{G\in\mathcal{G}} \mathop{\mathbb{E}}_{x\sim P}[\log g(x)] = \max_{G\in\mathcal{G}} \int p(x)\log g(x)\mathrm{d}x = \min_{G\in\mathcal{G}}\left(-\int p(x)\log g(x)\mathrm{d}x\right),$$

which is further equivalent to solve the following optimization problem:

$$\int p(x)\log p(x)\mathrm{d}x + \min_{G\in\mathcal{G}}\left(-\int p(x)\log g(x)\mathrm{d}x\right) = \min_{G\in\mathcal{G}} D_{\mathrm{KL}}(P \parallel G).$$

This is because the first term on the LHS is irrelevant from G and thus is a constant. From this expression, it is evident that the goal of MLE is to minimize a global statistical distance, namely, KL-divergence.

Figure 6 illustrates an 1D example wherein the MLE fails to achieve pointwise coverage. Although Figure 6, for pedagogical purpose, involves a generator family $\mathcal{G}$ consisting of only two generators, it is by no means a pathological case, since in practice generators always have limited expressive power, limited by a number of factors. For GANs, it is limited by the structure of generators. For VAEs, it is the structure of encoders and decoders. For Gaussian Mixture models, it is the dimension of the space and the number of mixture components. Given a $\mathcal{G}$ with limited expressive power, MLE cannot guarantee complete mode coverage.

Variational Autoencoders (VAEs). A VAE has a encoder $\theta \in \Theta$ and a decoder $\phi \in \Phi$ chosen from an encoder and decoder families, Θ and Φ. It also needs a known prior distribution Q (whose probability density is $q(\cdot)$) of latent variable z. Provided a decoder ϕ and the prior distribution Q, we can construct a generator G: to generate an x, we firstly sample a latent variable $z \sim Z$ and then sample an x according to the (approximated) likelihood function $p_\phi(x|z)$. To train a VAE, a target distribution P is provided and the training objective is

$$\max_{\theta\in\Theta,\phi\in\Phi} \int_x p(x)\cdot \mathrm{ELBO}_{\theta,\phi}(x)\mathrm{d}x, \tag{7}$$

where $\mathrm{ELBO}_{\theta,\phi}(x)$ is called the evidence lower bound, defined as

$$\mathrm{ELBO}_{\theta,\phi}(x) = \int_z p_\theta(z|x)\log p_\phi(x|z)\mathrm{d}z - \int_z p_\theta(z|x)\log\left(\frac{p_\theta(z|x)}{q(z)}\right)\mathrm{d}z, \tag{8}$$

Here $p_\theta(z|x)$ is the (approximated) posterior function.

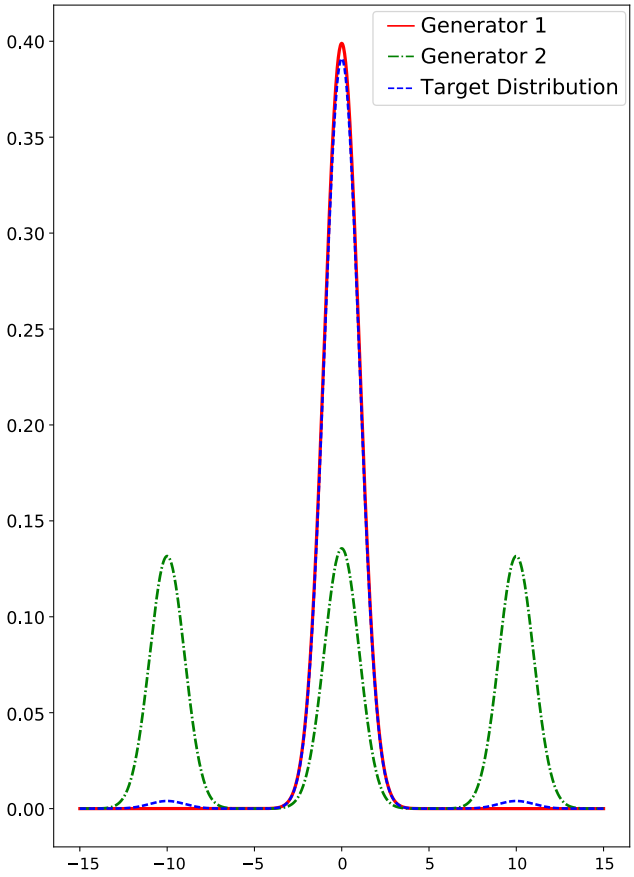

Figure 6: Consider a 1D target distribution P with three modes, i.e., a mixture of three Gaussians, $P = 0.98 \cdot \mathcal{N}(0,1) + 0.01 \cdot \mathcal{N}(10,1) + 0.01 \cdot \mathcal{N}(-10,1)$. In this example, the generator class $\mathcal{G}$ only contains two generators. The generated distribution of the first generator G_1 is $\mathcal{N}(0,1)$, while the distribution of the second generator G_2 is $0.34 \cdot \mathcal{N}(0,1) + 0.33 \cdot \mathcal{N}(10,1) + 0.33 \cdot \mathcal{N}(-10,1)$. In this case, we have $D_{\mathrm{KL}}(P, G_1) \approx 1.28$, $D_{\mathrm{KL}}(P, G_2) \approx 1.40$, $D_{\mathrm{KL}}(G_1, P) \approx 0.029$, and $D_{\mathrm{KL}}(G_2, P) \approx 2.81$ (all D_{KL} measures use a log base of 2). To minimize $D_{\mathrm{KL}}(P, G)$, maximum likelihood estimation method will choose the first generator, G_1. The probability of drawing samples from the side modes (in $[-14, -6]$ and $[6, 14]$) of the target distribution P is $\Pr_{x\sim P}[6 \leq |x| \leq 14] \approx 0.02$, but the probability of generating samples from the first generator in the same intervals is $\Pr_{x\sim G_1}[6 \leq |x| \leq 14] \approx 10^{-9}$. Thus, the side modes are almost missed. To make the first generator satisfy Equation (1), we have to choose $\psi \approx 10^{-7}$, which in practice implies no pointwise coverage guarantee. In contrast, the generated distribution of the second generator can satisfy Equation (1) with $\psi > 1/3$, which is a plausible pointwise coverage guarantee.

Let $G \in \mathcal{G}$ be a generator corresponding to the decoder ϕ and the prior Z, and let $g(\cdot)$ be the generative probability density of G. Then, we have the following derivation:

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{x\sim P}[\log g(x)] \\
&= \int_x p(x) \log g(x) \mathrm{d}x \\
&= \int_x p(x) \int_z p_\theta(z|x) \log\left(g(x)\right) \mathrm{d}z \mathrm{d}x \\
&= \int_x p(x) \int_z p_\theta(z|x) \log\left(\frac{p_\phi(x|z)q(z)}{p_\phi(z|x)}\right) \mathrm{d}z \mathrm{d}x \\
&= \int_x p(x) \int_z p_\theta(z|x) \log\left(\frac{p_\phi(x|z)q(z)p_\theta(z|x)}{p_\phi(z|x)p_\theta(z|x)}\right) \mathrm{d}z \mathrm{d}x \\
&= \int_x p(x) \left(\int_z p_\theta(z|x) \log\left(\frac{p_\phi(x|z)q(z)}{p_\theta(z|x)}\right) \mathrm{d}z + \int_z p_\theta(z|x) \log\left(\frac{p_\theta(z|x)}{p_\phi(z|x)}\right) \mathrm{d}z\right) \mathrm{d}x \\
&= \int_x p(x) \left(\int_z p_\theta(z|x) \log\left(\frac{p_\phi(x|z)q(z)}{p_\theta(z|x)}\right) \mathrm{d}z + D_{\mathrm{KL}}\left(p_\theta(z|x) \parallel p_\phi(z|x)\right)\right) \mathrm{d}x \\
&= \int_x p(x) \left(\mathrm{ELBO}_{\theta,\phi}(x) + D_{\mathrm{KL}}\left(p_\theta(z|x) \parallel p_\phi(z|x)\right)\right) \mathrm{d}x.
\end{aligned}
\tag{9}
$$

Algorithm 2 Training on empirical distribution

Parameters: T, a positive integer number of generators, and $\delta \in (0, 1)$, a covering threshold.
Input: a set $\{x_i\}_{i=1}^n$ of i.i.d. samples drawn from an unknown data distribution P.
For each x_i, initialize its weight $w_1(x_i) = 1/n$.
for $t = 1 \to T$ **do**
  Construct an empirical distribution $\widehat{P}_t$ such that each x_i is drawn with probability $\frac{w_t(x_i)}{W_t}$, where $W_t = \sum_i w_t(x_i)$.
  Train G_t on i.i.d. samples drawn from $\widehat{P}_t$.
  Train a discriminator D_t to distinguish the samples from $\widehat{P}_t$ and the samples from G_t.
  For each x_i, if $\left(\frac{1}{D_t(x_i)} - 1\right) \cdot \frac{w_t(x_i)}{W_t} < \frac{\delta}{n}$, set $w_{t+1}(x_i) = 2 \cdot w_t(x_i)$.
  Otherwise, set $w_{t+1}(x_i) = w_t(x_i)$.
end for
Output: a mixture of generators $\boldsymbol{G}^* = \{G_1, \ldots, G_T\}$.

Notice that $D_{\mathrm{KL}}\left(p_\theta(z|x) \parallel p_\phi(z|x)\right)$ is always non-negative and it reaches 0 when $p_\theta(z|x)$ is the same as $p_\phi(z|x)$. This means

$$\mathop{\mathbb{E}}_{x\sim P}[\log g(x)] \geq \int_x p(x) \cdot \mathrm{ELBO}_{\theta,\phi}(x)\mathrm{d}x.$$

If θ is perfectly trained, i.e., $p_\theta(z|x)$ matches exactly $p_\phi(z|x)$, then

$$\max_{G\in\mathcal{G}} \mathop{\mathbb{E}}_{x\sim P}[\log g(x)] = \max_{\theta\in\Theta,\phi\in\Phi} \int_x p(x) \cdot \mathrm{ELBO}_{\theta,\phi}(x)\mathrm{d}x.$$

From this perspective, it becomes evident that optimizing a VAE essentially amounts to a maximum likelihood estimation. Depending on the generator family $\mathcal{G}$ (determined by Φ and Z) and the encoder family Θ, mode collapse may not always happen. But since it is essentially a maximum likelihood estimation method, the pointwise mode coverage (1) can not be guaranteed in theory, as discussed in the previous paragraph.

Generative Adversarial Networks (GANs). Given a target distribution P, the objective of training a GAN [9] is to solve the following optimization problem:

$$\min_{G\in\mathcal{G}} \max_{D} L(G, D),$$

where $L(G, D)$ is defined as

$$L(G, D) = \mathop{\mathbb{E}}_{x\sim P}[\log(D(x))] + \mathop{\mathbb{E}}_{x\sim G}[\log(1 - D(x))] = \int_x p(x)\log(D(x)) + g(x)\log(1 - D(x))\mathrm{d}x.$$

As shown in [9], the optimal discriminator D^* of Nash equilibrium satisfies $D^*(x) \equiv 1/2$. When using D^* in $L(G, D)$, we have

$$L(G, D^*) = D_{\mathrm{KL}}\left(P \parallel \frac{P+G}{2}\right) + D_{\mathrm{KL}}\left(G \parallel \frac{P+G}{2}\right) - 2 = 2D_{\mathrm{JS}}(P \parallel G) - 2,$$

where D_{JS} is the Jensen-Shannon divergence. Thus, GAN essentially is trying to reduce the global statistical distance, measured by Jensen-Shannon divergence.

There are many variants of GANs, which use (more or less) different loss functions $L(G, D)$ in training. But all of them still focus on reducing a global statistical distance. For example, the loss function of the Wasserstein GAN [10] is $\mathbb{E}_{x\sim P}[D(x)] - \mathbb{E}_{x\sim G}[D(x)]$. Optimizing such a loss function over all 1-Lipschitz D is essentially to reduce the Wasserstein distance, another global statistical distance measure.

B Algorithm on Empirical Dataset

In practice, the provided dataset $\{x_i\}_{i=1}^n$ consists of n i.i.d. samples from P. According to the Glivenko-Cantelli theorem [36], the uniform distribution over n i.i.d. samples from P will converge

to P when n approaches to infinity. As a simple example, let P be a discrete distribution over two points, A and B, with $P(A) = 5/7$ and $P(B) = 2/7$. If 7 samples are drawn from P to form the input data, ideally they should be a multiset $\{A, A, A, A, A, B, B\}$. Each sample has a weight $1/7$, and the total weights of A and B are $5/7$ and $2/7$. Then we will train a generator G_1 from the training distribution where point A has training probability $5/7$ and point B has training probability $2/7$.

If the generator G_1 obtained is collapsed, e.g., G_1 samples A with probability 1 and samples B with probability 0, then ideally the discriminator D_1 will satisfy $D_1(A) = 5/12$ and $D_1(B) = 1$. Suppose the parameter $\delta = 1/4$ in Algorithm 1 (and Algorithm 2). We have

$$\left(\frac{1}{D_1(A)} - 1\right) \cdot \frac{w_1(A)}{W_1(A)} = \left(\frac{1}{D_1(A)} - 1\right) \cdot \frac{5}{7} \cdot \frac{1}{5} \geq \delta \cdot P(A) \cdot \frac{1}{5} = \delta/n = \frac{1/4}{7}$$

and

$$\left(\frac{1}{D_1(B)} - 1\right) \cdot \frac{w_1(B)}{W_1(B)} = \left(\frac{1}{D_1(B)} - 1\right) \cdot \frac{2}{7} \cdot \frac{1}{2} < \delta \cdot P(B) \cdot \frac{1}{2} = \delta/n = \frac{1/4}{7}.$$

Thus, each sample B will double the weight, and each sample A will remain the same weight unchanged. The total weight of A is $5/7$, and the total weight of B is $4/7$. In the second iteration, the total probability of A will be decreased to $5/9$ and the total probability of B will be increased to $4/9$. We will use the new probability to train the generator G_2 and the discriminator D_2, and repeat the above procedure.

In practice, we do not need to know the probability density $p(x)$ of P; every sample x_i is considered to have a finite and uniform probability measure. After the generator G is trained over this dataset, its generated sample distribution should approximate well the data distribution P. In light of this, the Algorithm 1 can be implemented empirically as what is outlined in Algorithm 2.

C Statistical Distance from Lower-bounded Pointwise Coverage

Equation (1) (i.e., $\forall x \in \mathcal{X}, g(x) \geq \psi \cdot p(x)$) is a pointwise lower-bounded data coverage that we pursue in this paper. If Equation (1) is satisfied, then the total variation distance between P and G is automatically upper bounded, because

$$\begin{aligned} D_{\text{TV}}(P \parallel Q) =& \frac{1}{2}\int_{\mathcal{X}} |p(x) - g(x)| \mathrm{d}x = \int_{\mathcal{X}} \mathbb{1}(p(x) > g(x)) \cdot (p(x) - g(x)) \mathrm{d}x \\ \leq & \int_{\mathcal{X}} \mathbb{1}(p(x) > g(x)) \cdot (p(x) - \psi \cdot p(x)) \mathrm{d}x \\ = & (1 - \psi) \cdot \int_{\mathcal{X}} \mathbb{1}(p(x) > g(x)) \cdot p(x) \mathrm{d}x \\ \leq & 1 - \psi. \end{aligned}$$

D Proof of Equation (3)

Suppose two arbitrary distributions P and Q are defined over a data space $\mathcal{X}$. G is the distribution of generated samples over $\mathcal{X}$. If the total variation distance between Q and G is at most 0.1, then we have

$$\begin{aligned} \Pr_{x \sim Q}\left[g(x) \geq \frac{1}{4}p(x)\right] = & \int_{\mathcal{X}} \mathbb{1}\left(g(x) \geq \frac{1}{4}p(x)\right) \cdot q(x) \mathrm{d}x \\ \geq & \int_{\mathcal{X}} \mathbb{1}\left(g(x), q(x) \geq \frac{1}{4}p(x)\right) \cdot q(x) \mathrm{d}x \\ = & \int_{\mathcal{X}} \mathbb{1}\left(q(x) \geq \frac{1}{4}p(x)\right) \cdot q(x) \mathrm{d}x - \int_{\mathcal{X}} \mathbb{1}\left(q(x) \geq \frac{1}{4}p(x) > g(x)\right) \cdot q(x) \mathrm{d}x \\ \geq & \frac{3}{4} - \int_{\mathcal{X}} \mathbb{1}\left(q(x) \geq \frac{1}{4}p(x) > g(x)\right)(q(x) - g(x) + g(x)) \mathrm{d}x \\ \geq & \frac{3}{4} - 0.1 - \frac{1}{4} = 0.4, \end{aligned}$$

where the first term of the right-hand side of the second inequality follows from

$$\int_{\mathcal{X}} \mathbb{1}\left(q(x) \geq \frac{1}{4}p(x)\right) \cdot q(x)\mathrm{d}x = 1 - \int_{\mathcal{X}} \mathbb{1}\left(q(x) < \frac{1}{4}p(x)\right) \cdot q(x)\mathrm{d}x \geq 1 - \int_{\mathcal{X}} \frac{1}{4}p(x)\mathrm{d}x = \frac{3}{4}.$$

And the third inequality follows from

$$\int_{\mathcal{X}} \mathbb{1}\left(q(x) \geq \frac{1}{4}p(x) > g(x)\right)(q(x) - g(x))\mathrm{d}x \leq \int_{\mathcal{X}} \mathbb{1}(q(x) > g(x))(q(x) - g(x))\mathrm{d}x \leq 0.1,$$

and

$$\int_{\mathcal{X}} \mathbb{1}\left(q(x) > \frac{1}{4}p(x) > g(x)\right)g(x)\mathrm{d}x \leq \int_{\mathcal{X}} \mathbb{1}\left(\frac{1}{4}p(x) > g(x)\right)g(x)\mathrm{d}x \leq \int_{\mathcal{X}} \frac{1}{4}p(x)\mathrm{d}x \leq \frac{1}{4}.$$

E Theoretical Analysis Details

In this section, we provide proofs of the lemmas and theorem presented in Section 4. We repeat the statements of the lemmas and theorem before individual proofs. We also provide details to further elaborate the discussion provided in Sec. 4.3 of the paper.

We follow the notations introduced in Sec. 4 of the main text. In addition, we will use $\log(\cdot)$ to denote $\log_2(\cdot)$ for short.

E.1 Proof of Lemma 1

Lemma 1. *Consider two distributions, P and Q, over the data space $\mathcal{X}$, and a generator G producing samples in $\mathcal{X}$. For any $\delta, \gamma \in (0, 1]$, if $D_{TV}(G \parallel Q) \leq \gamma$, then G can $(\delta, 1 - 2\delta - \gamma)$-cover (P, Q).*

Proof. Since $D_{\mathrm{TV}}(G||Q) \leq \gamma$ and $\int_{\mathcal{X}} q(x)\mathrm{d}x = \int_{\mathcal{X}} g(x)\mathrm{d}x = 1$, we know that

$$D_{\mathrm{TV}}(G \parallel Q) = \frac{1}{2}\int_{\mathcal{X}} |q(x) - g(x)|\mathrm{d}x = \int_{\mathcal{X}} \mathbb{1}(q(x) > g(x)) \cdot (q(x) - g(x))\mathrm{d}x \leq \gamma. \tag{10}$$

Next, we derive a lower bound of $\Pr_{x\sim Q}[x \text{ is } \delta\text{-covered by } G \text{ under } P]$:

$$\begin{aligned}
&\Pr_{x\sim Q}[x \text{ is } \delta\text{-covered by } G \text{ under } P] \\
=\ & \int_{\mathcal{X}} \mathbb{1}(g(x) \geq \delta \cdot p(x)) \cdot q(x)\mathrm{d}x \geq \int_{\mathcal{X}} \mathbb{1}(g(x), q(x) \geq \delta \cdot p(x)) \cdot q(x)\mathrm{d}x \\
=\ & \int_{\mathcal{X}} \mathbb{1}(q(x) \geq \delta \cdot p(x)) \cdot q(x)\mathrm{d}x - \int_{\mathcal{X}} \mathbb{1}(q(x) \geq \delta \cdot p(x) > g(x)) \cdot q(x)\mathrm{d}x \\
=\ & 1 - \int_{\mathcal{X}} \mathbb{1}(q(x) < \delta \cdot p(x)) \cdot q(x)\mathrm{d}x - \int_{\mathcal{X}} \mathbb{1}(q(x) \geq \delta \cdot p(x) > g(x)) \cdot q(x)\mathrm{d}x \\
\geq\ & 1 - \delta\int_{\mathcal{X}} p(x)\mathrm{d}x - \int_{\mathcal{X}} \mathbb{1}(q(x) \geq \delta \cdot p(x) > g(x)) \cdot q(x)\mathrm{d}x \\
=\ & 1 - \delta - \int_{\mathcal{X}} \mathbb{1}(q(x) \geq \delta \cdot p(x) > g(x)) \cdot (q(x) - g(x) + g(x))\mathrm{d}x \\
=\ & 1 - \delta - \int_{\mathcal{X}} \mathbb{1}(q(x) \geq \delta \cdot p(x) > g(x)) \cdot (q(x) - g(x))\mathrm{d}x - \int_{\mathcal{X}} \mathbb{1}(q(x) \geq \delta \cdot p(x) > g(x)) \cdot g(x)\mathrm{d}x \\
\geq\ & 1 - \delta - \gamma - \int_{\mathcal{X}} \mathbb{1}(q(x) \geq \delta \cdot p(x) > g(x)) \cdot g(x)\mathrm{d}x \\
\geq\ & 1 - \delta - \gamma - \delta\int_{\mathcal{X}} p(x)\mathrm{d}x = 1 - 2\delta - \gamma,
\end{aligned}$$

where the first equality follows from definition, the second equality follows from $\mathbb{1}(q(x) \geq \delta \cdot p(x)) = \mathbb{1}(g(x), q(x) \geq \delta \cdot p(x)) + \mathbb{1}(q(x) \geq \delta \cdot p(x) > g(x))$, the third inequality follows from Equation (10), and the last inequality follows from

$$\int_{\mathcal{X}} \mathbb{1}(q(x) \geq \delta \cdot p(x) > g(x)) \cdot g(x)\mathrm{d}x \leq \int_{\mathcal{X}} \mathbb{1}(\delta \cdot p(x) > g(x)) \cdot g(x)\mathrm{d}x \leq \int_{\mathcal{X}} \delta \cdot p(x)\mathrm{d}x.$$

□

E.2 Proof of Lemma 2

Here we first assume that the probability density g_t of generated samples is known. In Appendix E.5, we will consider the case where g_t is estimated by a discriminator as described in Section 3.

Lemma 2. *Recall that $T \in \mathbb{N}_{>0}$ and $\delta \in (0,1)$ are the input parameters of Algorithm 1. For any $\varepsilon \in [0,1)$ and any measurable subset $\mathcal{S} \in \mathcal{B}(\mathcal{X})$ whose probability measure satisfies $P(\mathcal{S}) \geq 1/2^{\eta T}$ with some $\eta \in (0,1)$, if in every round $t \in [T]$, G_t can $(\delta, 1-\varepsilon)$-cover (P, P_t), then the resulting mixture of generators $\boldsymbol{G}^*$ can $(1-\varepsilon/\ln 2 - \eta)\delta$-cover $\mathcal{S}$ under distribution P.*

Proof. First, we consider the total weight W_{t+1} after t rounds, we derive the following upper bound:

$$\begin{aligned}
W_{t+1} &= \int_{\mathcal{X}} w_{t+1}(x)\mathrm{d}x = \int_{\mathcal{X}} w_t(x) \cdot (1 + \mathbb{1}(g_t(x) < \delta \cdot p(x)))\mathrm{d}x \\
&= W_t + W_t \cdot \int_{\mathcal{X}} \mathbb{1}(g_t(x) < \delta \cdot p(x)) \cdot \frac{w_t(x)}{W_t}\mathrm{d}x \\
&= W_t + W_t \cdot \int_{\mathcal{X}} \mathbb{1}(g_t(x) < \delta \cdot p(x)) \cdot p_t(x)\mathrm{d}x \\
&= W_t + W_t \cdot \Pr_{x\sim P_t}[g_t(x) < \delta \cdot p(x)] \\
&= W_t + W_t \cdot (1 - \Pr_{x\sim P_t}[g_t(x) \geq \delta \cdot p(x)]) \\
&\leq W_t + W_t \cdot (1 - (1-\varepsilon)) \\
&\leq W_t \cdot (1+\varepsilon),
\end{aligned}$$

where the first equality follows from definition, the second equality follows from Line 9 of Algorithm 1, the forth equality follows from the construction of distribution P_t. In addition, the first inequality follows from that G_t can $(\delta, 1-\varepsilon)$-cover (P, P_t). Thus, $W_{T+1} \leq W_1 \cdot (1+\varepsilon)^T = (1+\varepsilon)^T$.

On the other hand, we have

$$\begin{aligned}
W_{T+1} &= \int_{\mathcal{X}} w_{T+1}(x)\mathrm{d}x \geq \int_{\mathcal{S}} w_{T+1}(x)\mathrm{d}x \geq \int_{\mathcal{S}} 2^{\sum_{t=1}^T \mathbb{1}(g_t(x) < \delta\cdot p(x))} p(x)\mathrm{d}x \\
&= \mathop{\mathbb{E}}_{x\sim P}\left[2^{\sum_{t=1}^T \mathbb{1}(g_t(x) < \delta\cdot p(x))} \middle| x \in \mathcal{S}\right] \Pr_{x\sim P}[x \in \mathcal{S}],
\end{aligned} \tag{11}$$

where the first equality follows from definition, the first inequality follows from $\mathcal{S} \subseteq \mathcal{X}$, and the second inequality follows from Line 9 of Algorithm 1. Dividing both sides by $\Pr_{x\sim P}[x \in \mathcal{S}]$ of (11) and taking the logarithm yield

$$\begin{aligned}
\log\left(\frac{W_{T+1}}{\Pr_{x\sim P}[x \in \mathcal{S}]}\right) &\geq \log\left(\mathop{\mathbb{E}}_{x\sim P}\left[2^{\sum_{t=1}^T \mathbb{1}(g_t(x) < \delta\cdot p(x))} \middle| x \in \mathcal{S}\right]\right) \\
&\geq \mathop{\mathbb{E}}_{x\sim P}\left[\sum_{t=1}^T \mathbb{1}(g_t(x) < \delta \cdot p(x)) \middle| x \in \mathcal{S}\right],
\end{aligned} \tag{12}$$

where the last inequality follows from Jensen's inequality.

Lastly, we have a lower bound for $\Pr_{x\sim G}[x \in \mathcal{S}]$:

$$\begin{aligned}
\Pr_{x\sim G}[x \in \mathcal{S}] &= \int_{\mathcal{S}} \frac{1}{T}\sum_{t=1}^{T} g_t(x)\mathrm{d}x \geq \int_{\mathcal{S}} \frac{1}{T}\sum_{t=1}^{T}(\mathbb{1}(g_t(x) \geq \delta \cdot p(x)) \cdot g_t(x))\mathrm{d}x \\
&\geq \int_{\mathcal{S}} \frac{1}{T}\sum_{t=1}^{T}(\mathbb{1}(g_t(x) \geq \delta \cdot p(x)) \cdot \delta \cdot p(x))\mathrm{d}x \\
&= \frac{\delta}{T}\int_{\mathcal{S}} \sum_{t=1}^{T}\mathbb{1}(g_t(x) \geq \delta \cdot p(x)) \cdot p(x)\mathrm{d}x \\
&= \frac{\delta}{T}\mathop{\mathbb{E}}_{x\sim P}\left[\sum_{t=1}^{T}\mathbb{1}(g_t(x) \geq \delta \cdot p(x))\middle| x \in \mathcal{S}\right] \cdot \Pr_{x\sim P}[x \in \mathcal{S}] \\
&= \frac{\delta}{T}\left(T - \mathop{\mathbb{E}}_{x\sim P}\left[\sum_{t=1}^{T}\mathbb{1}(g_t(x) < \delta \cdot p(x))\middle| x \in \mathcal{S}\right]\right) \cdot \Pr_{x\sim P}[x \in \mathcal{S}] \\
&\geq \delta(1 - \log(W_{T+1}/\Pr_{x\sim P}[x \in \mathcal{S}])/T) \cdot \Pr_{x\sim P}[x \in \mathcal{S}] \\
&\geq \delta(1 - \varepsilon/\ln 2 - \eta) \cdot \Pr_{x\sim P}[x \in \mathcal{S}],
\end{aligned}$$

where the third inequality follows from Equation (12), while the last inequality follows from $\log(W_{T+1}) \leq \log((1+\varepsilon)^T) \leq \varepsilon T/\ln 2$ and $\Pr_{x\sim P}[x \in \mathcal{S}] = P(\mathcal{S}) \geq 1/2^{\eta T}$. □

E.3 Proof of Theorem 1

Theorem 1. *Recall that $T \in \mathbb{N}_{>0}$ and $\delta \in (0,1)$ are the input parameters of Algorithm 1. For any measurable subset $\mathcal{S} \in \mathcal{B}(\mathcal{X})$ whose probability measure satisfies $P(\mathcal{S}) \geq 1/2^{\eta T}$ with some $\eta \in (0,1)$, if in every round $t \in [T]$, $D_{TV}(G_t \parallel P_t) \leq \gamma$, then the resulting mixture of generators $\boldsymbol{G}^*$ can $(1 - (\gamma + 2\delta)/\ln 2 - \eta)\delta$-cover $\mathcal{S}$ under distribution P.*

Proof. From Lemma 1, we have $\forall t \in [T]$, G_t can $(\delta, 1 - \gamma - 2\delta)$-cover (P, P_t). Combining it with Lemma 2, we have $\forall \mathcal{S} \subseteq \mathcal{X}$ with $P(\mathcal{S}) \geq 1/2^{\eta T}$, G can $(1 - (\gamma + 2\delta)/\ln 2 - \eta)\delta$-cover $\mathcal{S}$. □

E.4 Choice of T and δ according to Theorem 1

Suppose the empirical dataset has n data points independently sampled from a target distribution P. We claim that in our train algorithm, $T = O(\log n)$ suffices. This is because if a subset $\mathcal{S} \in \mathcal{B}(\mathcal{X})$ has a sufficiently small probability measure, for example, $P(\mathcal{S}) < 1/n^3$, then with a high probability (i.e., at least $1 - 1/n^2$), no data samples in $\{x_i\}_{i=1}^n$ is located in $\mathcal{S}$. In other words, the empirical dataset of size n reveals almost no information of a subset $\mathcal{S}$ if $P(\mathcal{S}) < 1/n^3$, or equivalently if $1/2^{\eta T} \approx 1/n^3$ (according to Theorem 1). This shows that $T = O(\log n)$ suffices.

Theorem 1 also sheds some light on the choice of δ in Algorithm 1 (and Algorithm 2 in practice). We now present the analysis details for choosing δ. We use $\mathcal{G}$ to denote the type of generative models trained in each round of our algorithm. According to Theorem 1, if we know η (depends on T) and γ (depends on $\mathcal{G}$), then we wish to maximize the lower bound $(1 - (\gamma + 2\delta)/\ln 2 - \eta)\delta$ over δ, and the optimal δ is $\frac{(1-\eta)\ln 2 - \gamma}{4}$. Although in practice γ is unknown and not easy to estimate, we note that γ is relatively small in practice, and η can be also small when we increase the number of rounds T.

Given two arbitrary distributions P and Q over $\mathcal{X}$, if the total variation distance between Q and a generated distribution G is at most γ (as we discussed in Sec. 1.1 of the main text), then we have

$$\begin{aligned}
\Pr_{x\sim Q}[g(x) \geq \delta \cdot p(x)] = & \int_{\mathcal{X}} \mathbb{1}(g(x) \geq \delta p(x)) \cdot q(x)\mathrm{d}x \\
\geq & \int_{\mathcal{X}} \mathbb{1}(g(x), q(x) \geq \delta \cdot p(x)) \cdot q(x)\mathrm{d}x \\
= & \int_{\mathcal{X}} \mathbb{1}(q(x) \geq \delta \cdot p(x)) \cdot q(x)\mathrm{d}x - \int_{\mathcal{X}} \mathbb{1}(q(x) \geq \delta \cdot p(x) > g(x)) \cdot q(x)\mathrm{d}x \\
\geq & \ 1 - \delta - \int_{\mathcal{X}} \mathbb{1}(q(x) \geq \delta \cdot p(x) > g(x))(q(x) - g(x) + g(x))\mathrm{d}x \\
\geq & \ 1 - \delta - \gamma - \delta = 1 - 2\delta - \gamma.
\end{aligned}$$

As discussed in Section 1.1, we can find a mixture of generators satisfying pointwise $(1-2\delta-\gamma)\delta$-coverage. Letting $\gamma = 0$, we see that the optimal choice of δ in this setting is $1/4$. And in this case, $(1-2\delta)\delta = 1/8$ is a theoretical bound of the coverage ratio by our algorithm.

E.5 Use of Estimated Probability Density g_t

In Algorithm 1, we use a discriminator D_t to estimate the probability density g_t of generated samples of each generator G_t. The discriminator D_t might not be perfectly trained, causing inaccuracy of estimating g_t. We show that the pointwise lower-bound in our data coverage is retained if two mild conditions are fulfilled by D_t.

1. In each round, only a bounded fraction of covered data points x (i.e., those with $g_t(x) \geq \delta \cdot p(x)$) is falsely classified and their weights are unnecessarily doubled. Concretely, $\forall t \in [T]$, if a sample x is drawn from distribution P_t, then the probability of both events—x is δ-covered by G_t under P and $\left(\frac{1}{D_t(x)} - 1\right) \cdot \frac{w_1(x)}{p(x)W_t} < \delta$—happening is bounded by ε'.
2. For any data point $x \in \mathcal{X}$, if in round t, the weight of x is not doubled, then with a good chance, x is really δ'-covered, where δ' can be smaller than δ. Formally, $\forall x \in \mathcal{X}, |\{t \in [T] | g_t(x) \geq \delta' \cdot p(x)\}| \geq \lambda \cdot \left|\left\{t \in [T] \middle| \left(\frac{1}{D_t(x)} - 1\right) \cdot \frac{w_t(x)}{p(x)W_t} \geq \delta\right\}\right|$. Because $\left(\frac{1}{D_t(x)} - 1\right) \cdot \frac{w_t(x)}{p(x)W_t} < \delta$ happens if and only if $w_{t+1}(x) = 2 \cdot w_t(x)$, we use the event $w_{t+1}(x) = 2 \cdot w_t(x)$ as an indicator of the event $\left(\frac{1}{D_t(x)} - 1\right) \cdot \frac{w_t(x)}{p(x)W_t} < \delta$.

If the condition (1) is satisfied, then we are able to upper bound the total weight W_{T+1}. Similarly to the proof of Lemma 2, this can be seen from the following derivation:

$$\begin{aligned}
W_{t+1} &= \int_{\mathcal{X}} w_{t+1}(x)\mathrm{d}x \\
&\leq \int_{\mathcal{X}} w_t(x) \cdot (1 + \mathbb{1}(g_t(x) < \delta \cdot p(x)) + \mathbb{1}(g_t(x) \geq \delta \cdot p(x) \wedge w_{t+1}(x) = 2w_t(x)))\mathrm{d}x \\
&= W_t + W_t \cdot \int_{\mathcal{X}} (\mathbb{1}(g_t(x) < \delta \cdot p(x)) + \mathbb{1}(g_t(x) \geq \delta \cdot p(x) \wedge w_{t+1}(x) = 2w_t(x))) \cdot \frac{w_t(x)}{W_t}\mathrm{d}x \\
&= W_t + W_t \cdot \int_{\mathcal{X}} (\mathbb{1}(g_t(x) < \delta \cdot p(x)) + \mathbb{1}(g_t(x) \geq \delta \cdot p(x) \wedge w_{t+1}(x) = 2w_t(x))) \cdot p_t(x)\mathrm{d}x \\
&= W_t + W_t \cdot \Pr_{x\sim P_t}[g_t(x) < \delta \cdot p(x)] + W_t \cdot \Pr_{x\sim P_t}[g_t(x) \geq \delta \cdot p(x) \wedge w_{t+1}(x) = 2w_t(x)] \\
&\leq W_t + W_t \cdot (1 - \Pr_{x\sim P_t}[g_t(x) \geq \delta \cdot p(x)]) + W_t \cdot \varepsilon' \\
&\leq W_t + W_t \cdot (1 - (1 - \varepsilon)) + W_t \cdot \varepsilon' \\
&\leq W_t \cdot (1 + \varepsilon + \varepsilon'),
\end{aligned}$$

Thus, the total weight W_{T+1} is bounded by $(1+\varepsilon+\varepsilon')^T$. Again in parallel to the proof of Lemma 2, we have

$$\begin{aligned} W_{T+1} &= \int_{\mathcal{X}} w_{T+1}(x)\mathrm{d}x \geq \int_{\mathcal{S}} w_{T+1}(x)\mathrm{d}x \geq \int_{\mathcal{S}} 2^{\sum_{t=1}^{T}\mathbb{1}(w_{t+1}(x)=2\cdot w_t(x))}p(x)\mathrm{d}x \\ &= \mathop{\mathbb{E}}_{x\sim P}\left[2^{\sum_{t=1}^{T}\mathbb{1}(w_{t+1}(x)=2\cdot w_t(x))}\middle| x\in\mathcal{S}\right]\Pr_{x\sim P}[x\in\mathcal{S}]. \end{aligned}$$

Dividing both sides by $\Pr_{x\sim P}[x\in\mathcal{S}]$ yields

$$\begin{aligned} \log\left(\frac{W_{T+1}}{\Pr_{x\sim P}[x\in\mathcal{S}]}\right) &\geq \log\left(\mathop{\mathbb{E}}_{x\sim P}\left[2^{\sum_{t=1}^{T}\mathbb{1}(w_{t+1}(x)=2w_t(x))}\middle| x\in\mathcal{S}\right]\right) \\ &\geq \mathop{\mathbb{E}}_{x\sim P}\left[\sum_{t=1}^{T}\mathbb{1}(w_{t+1}(x)=2w_t(x))\middle| x\in\mathcal{S}\right]. \end{aligned}$$

Meanwhile, if the condition (2) is satisfied, then

$$\begin{aligned} \lambda\cdot\left(T-\mathop{\mathbb{E}}_{x\sim P}\left[\sum_{t=1}^{T}\mathbb{1}(w_{t+1}(x)=2w_t(x))\middle| x\in\mathcal{S}\right]\right)\leq \\ T-\mathop{\mathbb{E}}_{x\sim P}\left[\sum_{t=1}^{T}\mathbb{1}(g_t(x)<\delta'\cdot p(x))\middle| x\in\mathcal{S}\right]. \end{aligned} \tag{13}$$

Following the proof of Lemma 2, we obtain

$$\begin{aligned} \Pr_{x\sim G}[x\in\mathcal{S}] &= \int_{\mathcal{S}}\frac{1}{T}\sum_{t=1}^{T}g_t(x)\mathrm{d}x \geq \int_{\mathcal{S}}\frac{1}{T}\sum_{t=1}^{T}(\mathbb{1}(g_t(x)\geq\delta'\cdot p(x))\cdot g_t(x))\mathrm{d}x \\ &\geq \int_{\mathcal{S}}\frac{1}{T}\sum_{t=1}^{T}(\mathbb{1}(g_t(x)\geq\delta'\cdot p(x))\cdot\delta'\cdot p(x))\mathrm{d}x \\ &= \frac{\delta'}{T}\int_{\mathcal{S}}\sum_{t=1}^{T}\mathbb{1}(g_t(x)\geq\delta'\cdot p(x))\cdot p(x)\mathrm{d}x \\ &= \frac{\delta'}{T}\mathop{\mathbb{E}}_{x\sim P}\left[\sum_{t=1}^{T}\mathbb{1}(g_t(x)\geq\delta'\cdot p(x))\middle| x\in\mathcal{S}\right]\cdot\Pr_{x\sim P}[x\in\mathcal{S}] \\ &= \frac{\delta'}{T}\left(T-\mathop{\mathbb{E}}_{x\sim P}\left[\sum_{t=1}^{T}\mathbb{1}(g_t(x)<\delta'\cdot p(x))\middle| x\in\mathcal{S}\right]\right)\cdot\Pr_{x\sim P}[x\in\mathcal{S}] \\ &\geq \frac{\delta'\lambda}{T}\left(T-\mathop{\mathbb{E}}_{x\sim P}\left[\sum_{t=1}^{T}\mathbb{1}(w_{t+1}(x)=2w_t(x))\middle| x\in\mathcal{S}\right]\right) \\ &\geq \delta'\lambda(1-\log(W_{T+1}/\Pr_{x\sim P}[x\in\mathcal{S}])/T)\cdot\Pr_{x\sim P}[x\in\mathcal{S}] \\ &\geq \delta'\lambda(1-(\varepsilon+\varepsilon')/\ln 2-\eta)\cdot\Pr_{x\sim P}[x\in\mathcal{S}], \end{aligned}$$

where the third inequality follows from Equation (13), and other steps are similar to the proof in Lemma 2. By combining with Lemma 1, the final coverage ratio of Theorem 1 with imperfect discriminators D_t should be $(1-(\gamma+2\delta+\varepsilon')/\ln 2-\eta)\delta'\lambda$.

E.6 Discussion on Generalization

Recently, Arora et al. [26] proposed the *neural net distance* for measuring generalization performance of GANs. However, their metric still relies on a global distance measure of two distributions, not necessarily reflecting the generalization for pointwise coverage.

While a dedicated answer of this theoretical question is beyond the scope of this work, here we propose our notion of generalization and briefly discuss its implication for our algorithm. Provided

a training dataset consisting of n i.i.d. samples $\{x_i\}_{i=1}^n$ drawn from the distribution P, we train a mixture of generators $\boldsymbol{G}^*$. Our notion of generalization is defined as $\Pr_{x\sim P}[x \text{ is } \psi\text{-covered by } \boldsymbol{G}^*]$, the probability of x being ψ-covered by empirically trained $\boldsymbol{G}^*$ when x is sampled from the true target distribution P. A perfect generalization has a value 1 under this notion. We claim that given fixed T rounds of our algorithm and a constant $\varepsilon \in (0,1)$, if G_t in each round is from a family $\mathcal{G}$ of generators (e.g., they are all GANs with the same network architecture), and if n is at least $\Omega(\varepsilon^{-1}T\log|\mathcal{G}|)$, then we have the generalization $\Pr_{x\sim P}[x \text{ is } \psi\text{-covered by } \boldsymbol{G}^*] \geq 1-\varepsilon$. Here $|\mathcal{G}|$ is the size of *essentially* different generators in $\mathcal{G}$. Next, we elaborate this statement.

Generalization Analysis. Our analysis start with a definition of a *family* of generators. In each round of our algorithm, we train a generator G_t. We now identify a family of generators from which G_t is trained. In general, a generator G can be viewed as a pair $(f(\cdot), Z)$, where Z is the latent space distribution (or prior distribution) over the latent space $\mathcal{Z}$, and $f(\cdot)$ is a transformation function that maps the latent space $\mathcal{Z}$ to a target data domain $\mathcal{X}$. Let z be a random variable of distribution Z. Then, the generated distribution (i.e., distribution of samples generated by G) is denoted by the distribution of $f(z)$. For example, for GANs [9] and VAEs [42], $f(\cdot)$ is a function represented by a neural network, and Z is usually a standard Gaussian or mixture of Gaussians.

In light of this, we define a family $\mathcal{G}$ of generators represented by a pair $(\mathcal{F}, Z)$, where $\mathcal{F}$ is a set of functions mapping from $\mathcal{Z}$ to $\mathcal{X}$. For example, in the framework of GANs, $\mathcal{F}$ can be expressed by a neural network with a finite number of parameters which have bounded values. If the input to the neural network (i.e., the latent space) is also bounded, then we are able to apply *net* argument (see e.g., [26]) to find a finite subset $\mathcal{F}' \subset \mathcal{F}$ such that for any $f \in \mathcal{F}$, there exists a function $f' \in \mathcal{F}'$ sufficiently close to f. Then the size of $\mathcal{F}'$, denoted by $|\mathcal{F}'|$, can be regarded as the number of "essentially different" functions (or neural networks).

Recall that the generator family $\mathcal{G}$ can be represented by $(\mathcal{F}, Z)$. If the latent space Z is fixed (such as a single Gaussian), then we can define "essentially different" generators in a way similar to the definition of "essentially different" functions in $\mathcal{F}$. If the number of "essentially different" generators from $\mathcal{G}$ is finite, we define the size of $\mathcal{G}$ as $|\mathcal{G}|$.

With this notion, the number of different mixture of generators $\boldsymbol{G}^* = \{G_1, ..., G_T\}$ is at most $|\mathcal{G}|^T$. Consider a uniform mixture $\boldsymbol{G}^*$ of generators, $G_1, G_2, \cdots, G_T \in \mathcal{G}$. If $\Pr_{x\sim P}[x \text{ is } \textit{not } \psi\text{-covered by } \boldsymbol{G}^*] \geq \varepsilon$, then for n i.i.d. samples $x_1, x_2, \cdots, x_n \sim P$, the probability that every x_i is ψ-covered by G is at most $(1-\varepsilon)^n$, that is,

$$\Pr_{x_1,...,x_n\sim P}[\text{every } x_1, ..., x_n \text{ is } \psi\text{-convered by } \boldsymbol{G}^*] \leq (1-\varepsilon)^n.$$

Next, by union bound over all possible mixtures $\boldsymbol{G}^*$ that satisfies $\Pr_{x\sim P}[x \text{ is not } \psi\text{-covered by } \boldsymbol{G}^*] \geq \varepsilon$, we have the following probability bound:

$$\Pr_{x_1,...,x_n\sim P}\left[\exists \boldsymbol{G}^* \text{s.t. } \Pr_{x\sim P}[x \text{ is } \textit{not } \psi\text{-covered by } \boldsymbol{G}^*] \geq \varepsilon \text{ and every } x_1, ..., x_n \text{ is } \psi\text{-convered by } \boldsymbol{G}^*\right] \leq (1-\varepsilon)^n|\mathcal{G}|^T. \quad (14)$$

Thus, if $n \geq \Omega(\varepsilon^{-1}T\log|\mathcal{G}|)$, then with a high probability, the inverse of the probability condition above is true, because in this case $(1-\varepsilon)^n$ on the right-hand side of (14) is small—that is, with a high probability, for any mixture $\boldsymbol{G}^*$ that satisfies $\Pr_{x\sim P}[x \text{ is } \textit{not } \psi\text{-covered by } G] \geq \varepsilon$, there must exist a sample x_i such that x_i cannot be ψ-covered by $\boldsymbol{G}^*$. The occurrence of this condition implies that if we find a generator mixture $\boldsymbol{G}^*$ that can ψ-cover every x_i, then $\Pr_{x\sim P}[x \text{ is } \psi\text{-covered by } G] \geq 1-\varepsilon$. In other words, we conclude that if we have $n \geq \Omega(\varepsilon^{-1}T\log|\mathcal{G}|)$ i.i.d. samples $\{x_i\}_{i=1}^n$ drawn from the distribution P, and if our algorithm finds a mixture $\boldsymbol{G}^*$ of generators that can ψ-cover every x_i, then with a high probability, our notion of generalization has $\Pr_{x\sim P}[x \text{ is } \psi\text{-covered by } G] \geq 1-\varepsilon$.

F Experiment Details and More Results

F.1 Network Architecture and Training Hyperparameters.

In our tests, we construct a mixture of GANs. The network architecture of the GANs in show in Table 2 for experiments on synthetic datasets and in Table 3 for real image datasets. All experiments use Adam optimizer [43] with a learning rate of 10^{-3}, and we set $\beta_1 = 0.5$ and $\beta_2 = 0.999$ with a mini-batch size of 128.

layer	output size	activation function
input (dim 10)	10	
Linear	32	ReLU
Linear	32	ReLU
Linear	2	

Table 2: **Network structure** for synthetic data generator.

layer	output size	kernel size	stride	BN	activation function
input (dim 100)	100×1×1				
Transposed Conv	512×4×4	4	1	Yes	ReLU
Transposed Conv	256×8×8	4	2	Yes	ReLU
Transposed Conv	128×16×16	4	2	Yes	ReLU
Transposed Conv	channel×32×32	4	2	No	Tanh

Table 3: **Network structure** for image generator. channel=3 for Stacked MNIST and channel=1 for FasionMNIST+MNIST.

F.2 Additional Experiment Details on Real Data

Stacked MNIST dataset. Stacked MNIST is an augmentation of MNIST dataset [13] for evaluating mode collapse. We randomly sample three images from MNIST dataset and stack them in RGB channels of an image. In this way, we construct a dataset of 100k images, each of which has a dimension of $32 \times 32 \times 3$.

Pre-trained classifier. For Fashion-MNIST with partial MNIST dataset, we use all the training data of Fashion-MNIST and MNIST to train a 11-class classifier. For stacked MNIST dataset, we train a 10-class classifier on MNIST, and use it as a 1000-class classifier on stacked MNIST (by applying the 10-class MNIST classifier on each color channel). For each experiment, we regard each class as a mode, and use the pre-trained classifier to classify the generated samples into individual modes. After classifying generated samples, we can estimate the generation probability for each mode.

F.3 Comparison with AdaGAN on Synthetic Dataset and Stacked MNIST

Mixture of Gaussians and Stacked MNIST. We conduct experiments on the same synthetic dataset and Stacked MNIST as used in AdaGAN [33]. All synthetic data points are distributed on a 2D plane, consisting of M ($M = 10$) Gaussians uniformly sampled in a squared region $[-15, 15] \times [-15, 15]$, all with the same variance $\sigma_0^2 = 0.05$.

We evaluate our algorithm by checking how many iterations (i.e., the parameter T in Algorithm 1) it takes to cover all modes, and compare it with AdaGAN. A mode is considered covered, if in N generated samples, there exist at least $0.01 \cdot N/M$ samples landed within a distance $3\sigma_0$ away from the mode's center. The experiments on both our algorithms and AdaGAN are repeated 15 times. On this synthetic dataset, both our algorithm and AdaGAN can cover all modes in 2 iterations. For Stacked MNIST, both our method and AdaGAN can cover all modes in 5 iterations.

More challenging synthetic datasets. Furthermore, we test our method and AdaGAN on two other synthetic datasets that have more challenging mode distributions. The first one, referred as D_s, has 20 modes distributed along a spiral curve (see Figure 7-left). Each mode is a set of points following a Gaussian distribution (with a variance of 1). The center of i-th mode ($i = 1..20$) is located at $(\cos(i/3) \cdot i \cdot i, \sin(i/3) \cdot i \cdot i)$. The second dataset, referred as D_i, has $21 \times 21 + 1$ modes, among which $21 \cdot 21 = 441$ modes locate on a $[-10, 10] \times [-10, 10]$ uniform grid and one additional mode is isolated at $(100, 100)$ (see Figure 7-right). Each mode is also a set of points under a Gaussian distribution (with a variance of 0.05).

For both datasets, we evaluate how many modes are covered as the number of iterations increases in both our method and AdaGAN. The mode coverage is defined in the same way as in the previous experiment. As shown in Figure 7, our algorithm covers all the modes, and outperforms AdaGAN on both datasets. In terms of efficiency, AdaGAN takes 437 min (25 iterations) and still miss some modes, while our method takes only 134 min (9 iterations) to cover all modes.

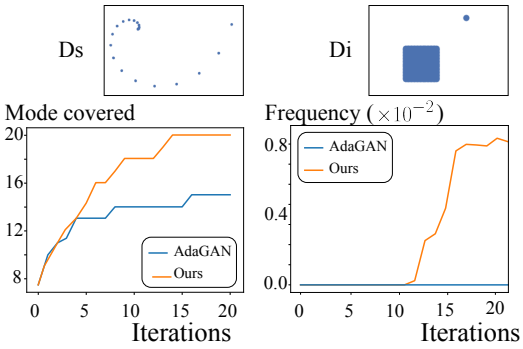

Figure 7: **Challenging datasets.** We compare our method with AdaGAN on two datasets (top). (left) Our method covers all modes in D_s dataset with 20 iterations in average. (right) Our method increases the sampled frequency (sampling weights) of the separate mode as the training iteration progresses, whereas AdaGAN increases the sampling frequency of the separated modes. Eventually, AdaGAN can only cover 14 modes in D_s and never cover the separated mode in D_i. In contrast, our method successfully covers all modes.

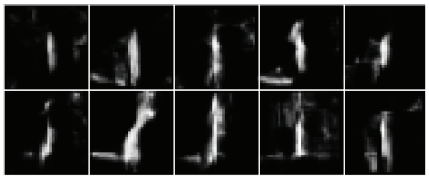

Figure 8: **Sampled "1" images by a single generator.** Based on the observation we draw from Figure 4, we train a single GAN using 60k Fashion-MNIST images together with 300 MNIST "1" images, and the GAN is able to generate images close to "1". Here we show a few generated images from the resulting generator.