BAYESIAN MODELLING AND MONTE CARLO INFERENCE FOR GAN

Anonymous authors
Paper under double-blind review

ABSTRACT

Bayesian modelling is a principal framework to perform model aggregation, which has been a primary mechanism to combat mode collapsing in the context of Generative Adversarial Networks (GANs). In this paper, we propose a novel Bayesian modelling framework for GANs, which iteratively learns a distribution over generators with a carefully crafted prior. Learning is efficiently triggered by a tailored stochastic gradient Hamiltonian Monte Carlo with novel gradient approximation to perform Bayesian inference. Our theoretical analysis further reveals that our treatment is the first Bayesian modelling framework that yields an equilibrium where generator distributions are faithful to the data distribution. Empirical evidence on synthetic high-dimensional multi-modal data and the natural image database CIFAR-10 demonstrates the superiority of our method over both start-of-the-art multi-generator GANs and other Bayesian treatment for GANs.

1 INTRODUCTION

Generative Adversarial Networks (GAN) (Goodfellow et al., 2014) is a popular method to learn a distribution on complex data such as natural images, videos and texts. However, it is notoriously hard to train and suffers from *mode collapse*. There has been a series of works addressing these issues. One noticeable thread focuses on objective design, which improves the original Jensen-Shannon divergence with more stable pseudo-metrics such as f-divergence (Nowozin et al., 2016), χ^2-divergence (Mao et al., 2017), and Wasserstein distance (Arjovsky et al., 2017). However, such treatment is inherently limited when a single generator does not include enough model capacity to capture the granularity in data distribution in practice. Clearly, such generator can never produce accurate samples regardless of the choice of objectives.

An alternative remedy is to learn multiple generators instead of a single generator. This type of methods (Hoang et al., 2018; Tolstikhin et al., 2017; Wang et al., 2016) is motivated by a straightforward intuition that multiple generators can better model multi-modal distributions since each generator only needs to capture a subset of the modes. To entail model aggregation, Bayesian modelling is a natural and principle framework to articulate the aggregation process in a probabilistic way.

Recently, Saatci & Wilson (2017) proposes Bayesian GAN, a probabilistic framework for GAN under Bayesian inference. It shows that modelling the distribution of generator helps alleviate mode collapse and motivates interpretability of the learned generators. The probabilistic model is built upon conditional distribution of generator and discriminator, whose maximum likelihood estimation can be realized as a metaphor of typical GAN objective. Empirical study shows Bayesian GAN has amazing superiority on semi-supervised image classification task. However, a critical theoretical question of this framework is still not well-understood: does it really converge to the generator distribution that produce the real data distribution? Accutally, our emprical study show that Bayesian GAN suffers from severe performance loss due to this lack of convergence guarantee.

With this observation, we follow the prior work to exploit Bayesian modelling as a principled way to realize model aggregation, but approach this problem from a theoretical perspective; we try to analyze the developed treatment (including choice of priors, approximate inference, and its convergence property) and simultaneously propose a new Bayesian modelling framework with the desirable convergence guarantee and consequently superior empirical performance.

Table 1: Common GAN objective functions.

	$\phi_1(D)$	$\phi_2(D)$	$\phi_3(D)$	MIN-MAX STYLE
GAN (MIN-MAX)	$\log(D)$	$\log(1-D)$	$-\log(1-D)$	YES
GAN (NON-SATURATING)	$\log(D)$	$\log(1-D)$	$\log(D)$	NO
WASSERSTEIN GAN	D	$-D$	D	YES
LEAST-SQUARES GAN	$(D-1)^2$	D^2	$(D-1)^2$	NO

Our main contributions are:

- We theoretically establish, to our best knowledge, the first Bayesian treatment of GANs such that any generator distribution faithful to the data distribution is an equilibrium.
- We prove the previous Bayesian method (Saatci & Wilson, 2017) for any minimax GAN objective induces incompatibility of its defined conditional distributions.
- We propose two special Monte Carlo inference algorithms for our Bayesian model which efficiently approximate the gradient of a non-differentiable criterion.
- Empirical studies on synthetic high-dimensional multi-modal data and CIFAR-10 demonstrate the superiority of the proposed framework over the state-of-the-art GAN methods.

2 RELATED WORK

Generative Adversarial Networks is powerful class of methods to learn a generative model for any complex target data distribution. There is a game between a generator and a discriminator. Both of them adapt their strategies to maximize their own objective function involving the other:

$$\begin{aligned} &\max_{\theta_d} \mathcal{J}_d(\theta_d;\theta_g) = \mathbb{E}_{\mathbf{x}\sim p_{data}}[\phi_1(D(\mathbf{x};\theta_d)])] + \mathbb{E}_{\mathbf{x}\sim p_{gen}(\cdot;\theta_g)}[\phi_2(D(\mathbf{x};\theta_d))], \\ &\max_{\theta_g} \mathcal{J}_g(\theta_g;\theta_d) = \mathbb{E}_{\mathbf{x}\sim p_{gen}(\cdot;\theta_g)}[\phi_3(D(\mathbf{x};\theta_d))]. \end{aligned} \tag{1}$$

Eqn. 1 gives a general mathematical form where p_{data} is real data distribution and $p_{gen}(\cdot;\theta_g)$ are generated data distribution with generator parameter θ_g. The objective functions ϕ_1, ϕ_2, ϕ_3 (termed as *GAN objective* in this paper) are elaborately chosen such that at the equilibrium, the generator generates the target data distribution. Table 1 summarizes several widely used GAN objectives, including the original min-max version, non-saturating version of original GAN (Goodfellow (2016)), LSGAN (Mao et al. (2017)), and WGAN (Arjovsky et al. (2017)). As reported in Table 1, some GAN objectives which satisfy $\phi_3(\cdot) = -\phi_2(\cdot)$ actually represent a min-max game, i.e. $\min_{\theta_g} \max_{\theta_g} \mathcal{J}_d(\theta_d;\theta_g)$.

Training GAN with multiple generators is considered in several recent works to mitigate the mode collapse problem. In the spirit of boosting algorithm, Wang et al. (2016) propose to progressively train new generator using a subset of training data that are not well captured by previous generators, while Tolstikhin et al. (2017) further propose a more robust mechanism to reweight samples in the training set for new generator. From the perspective of game theory, MIX-GAN (Arora et al., 2017) extends the game between a single generator and discriminator to the multiple-player setting. Other works resort to third party classifiers to help multiple generators and discriminators achieve better equilibrium, such as MAD-GAN (Ghosh et al., 2017) and the recent state-of-art method, MGAN (Hoang et al., 2018).

Bayesian GAN proposed by Saatci & Wilson (2017) adapts a different approach which models generator and discriminator distributions by defining the conditional posteriors (Eqn. 8). The posterior likelihood is specially designed such that maximizing it exactly corresponds to optimizing GAN objectives. The authors argue that compare to point mass ML estimation, learning the generator distribution which is multi-modal itself offers better ability to fit a multi-modal data distribution.

To facilitate discussion, we categorize GAN frameworks into the following taxonomy: *optimization-based methods* and *probabilistic methods*. Optimization-based methods set up an explicit mini-max game between the generator and discriminator, where an ideal equilibrium typically characterize a generator faithful to data distribution. In probabilistic methods, generators and discriminators evolve as particles of underlying distributions, where an equilibrium is searched from a stochastic exploration in the distribution space (of the generators and discriminators).

3 METHODOLOGY

To start with, we first summarize the notations. Second, we elaborate our Bayesian modelling for GAN and develop constituent prior and likelihood formulations. We delay a detailed discussion of the motivation of our modelling and the comparison with previous Bayesian method until Section 4. Finally, we develop algorithms to infer the posterior for our Bayesian modelling.

3.1 NOTATIONS

$p_{data}(\mathbf{x})$ over a sample space $\mathcal{X}$ is the data distribution we want to learn. Our generator and discriminator are parameterized by $\theta_g \in \Theta_g$ and $\theta_d \in \Theta_d$. A generator with parameter θ_g defines a mapping from a random noise vector $\mathbf{z} \sim p_z$ to a random vector $G(\mathbf{z}; \theta_g)$. The induced $G(\mathbf{z}; \theta_g)$ has generated data distribution $p_{gen}(\mathbf{x}; \theta_g)$. A discriminator is a function mapping data to a real-valued score, i.e. $D(\mathbf{x}; \theta_d) : \mathcal{X} \to [0, 1]$ (or $\mathcal{X} \to \mathbb{R}$ in some settings). Further, we use $q_g(\theta_g) \in \mathcal{P}^{\Theta_g}$, $q_d(\theta_d) \in \mathcal{P}^{\Theta_d}$ to denote the distribution over generators and discriminators respectively.

The data distribution generated by generators under the distribution $q_g(\theta_g)$ is naturally a mixture of data distribution given by every single generator, $p_{model}(\mathbf{x}) = \mathbb{E}_{\theta_g \sim q_g(\theta_g)}[p_{gen}(\mathbf{x}; \theta_g)]$. Our goal is to learn a generator distribution such that the total mixture of generated data distribution matches our target, i.e. $p_{model}(\mathbf{x}) \simeq p_{data}(\mathbf{x})$.

$\mathcal{J}_g(\theta_g; \theta_d)$ and $\mathcal{J}_d(\theta_d; \theta_g)$ denote objective functions of generator and discriminator. The common choices[1] are listed in Table 1. With a slight abuse of the notation, we extend the notation $\mathcal{J}_g(\theta_g; D^*)$ by replacing $D(\mathbf{x}; \theta_d)$ in equation 1 with any score function D^*. Then $\mathcal{J}_g(\theta_g; \theta_d)$ can be viewed as an abbreviation of $\mathcal{J}_g(\theta_g; D(\cdot; \theta_d))$. Similarity we define $\mathcal{J}_d(\theta_d; p(\cdot))$ to represent objective for discriminator given a virtual generator that generates data from distribution $p(\cdot)$.

3.2 BAYESIAN MODELLING FOR GAN

Likelihood. The likelihood function indicates a preference for generators and discriminators given current distributions of generator and discriminator, thus should encode the information that distributionally reflect the objective of generators $\mathcal{J}_g$ and discriminators $\mathcal{J}_d$. An ideal solution is to form a distribution that is proportional to quantities that evaluates such fitness:

$$p(\theta_g) \propto \exp\{\mathcal{J}_g(\theta_g; D^{(t)})\}, \quad p(\theta_d) \propto \exp\{\mathcal{J}_d(\theta_d; p^{(t)}_{model})\}. \tag{2}$$

where $p^{(t)}_{model}(\mathbf{x}) = \mathbb{E}_{\theta_g \sim q_g^{(t)}}[p_{gen}(\mathbf{x}; \theta_g)]$ is the mixed data distribution under current generator distribution $q_g^{(t)}$ and $D^{(t)}(\cdot) = \mathbb{E}_{\theta_d \sim q_d^{(t)}}[D(\cdot; \theta_d)]$ is the averaged discriminating score function under current discriminator distribution $q_d^{(t)}$. We emphasize, although sharing the same spirit of reflecting the GAN objectives in likelihood, there is a crucial difference between our likelihood model and that of Bayesian GAN. We will revisit it in the later theory section.

Prior. While the likelihood design is rather straightforward, an ideal prior is more involved. When the generated data distribution is increasingly close to the real data distribution, there will be less information for discriminator to distinguish between them; consequently, the discriminator tends to assign equal scores to all data samples, resulting in equal likelihoods for all generators. At that stage, a good strategy is to keep the generator distribution the same as the previous time step, since it already generates the desired data distribution perfectly. Hence, we use the generator distribution in the previous time step as a prior for the next time step. Such dynamically evolving prior for generator turns out to be crucial. In Section 4.2, we show the Bayesian model in previous work suffers from bad convergence due to its fixed and weakly informative prior. In contrast, we set a uniform improper prior on the discriminator to pursuit unrestricted adaptability to evolving generators.

Posterior. Integrating the prior mentioned above and the likelihood in Eqn. 2, we formulate our iterative Bayesian model as

$$q_g^{(t+1)}(\theta_g) \propto \exp\{\mathcal{J}_g(\theta_g; D^{(t)})\} \cdot q_g^{(t)}(\theta_g), \quad q_d^{(t+1)}(\theta_d) \propto \exp\{\mathcal{J}_d(\theta_d; p^{(t)}_{model})\}. \tag{3}$$

[1] The concepts of minimax version and non-saturating version of vanilla GAN are first introduced in Goodfellow (2016).

This iterative process for updating $q_g^{(t)}$ and $q_d^{(t)}$ can be *universally* applied to any common GAN objectives with a guarantee of *effectiveness* (i.e., any desired generator distribution is a convergence point of the iteration) which will be formally stated in Section 4.1.

3.3 Inference Algorithm

So far we have introduced our Bayesian modelling for GAN. In this section we develop novel inference algorithms to compute the posterior. Similar to most advanced Bayesian methods, exact calculation of the posterior is *intractable*. Following the strategy in Saatci & Wilson (2017), we adopt Stochastic Gradient Hamiltonian Monte Carlo to generate samples from the posterior. In each iteration, M samples $\{\theta_{g,m}^{(t)}\}_{m=1}^{M}$ are used to approximate the generator distribution $q_g^{(t)}$.

$$\nabla_{\theta_d} \log q_d^{(t+1)}(\theta_d) = \nabla_{\theta_d} \mathcal{J}_d(\theta_d; p_{model}^{(t)}) \simeq \frac{1}{M_g} \sum_{m=1}^{M_g} \nabla_{\theta_d} \mathcal{J}_d(\theta_d; p_{gen}(\cdot; \theta_{g,m}^{(t)})), \tag{4}$$

$$\nabla_{\theta_g} \log q_g^{(t+1)}(\theta_g) = \nabla_{\theta_g}(\mathcal{J}_g(\theta_g; D^{(t)}) + \log q_g^{(t)}(\theta_g)) \simeq \nabla_{\theta_g}(\mathcal{J}_g(\theta_g; \frac{1}{M_d} \sum_m D(\cdot; \theta_{d,m}^{(t)})) + \log q_g^{(t)}(\theta_g)). \tag{5}$$

Empowered by the adapted SGHMC (algorithm 1 in appendix), we are able to sample from $q_g^{(t+1)}$ and $q_d^{(t+1)}$ based on gradients in Eqn. 4 and Eqn. 5. The gradients come from two sides: the GAN objective $\mathcal{J}_g, \mathcal{J}_d$ and the prior $q_g^{(t)}$. Getting GAN objective's gradient is easy while computing the prior's gradient, $\nabla_{\theta_g} \log q_g^{(t)}(\theta_g)$, is actually non-trivial since we have no exact analytic form of $q_g^{(t)}(\theta_g)$. To address this challenge, we propose the following two methods to approximate $\nabla_{\theta_g} \log q_g^{(t)}(\theta_g)$, leading to two practical inference algorithms.

Gaussian Mixture Approximation (GMA). Although the analytic form of the distribution $q_g^{(t)}(\theta_g)$ is unknown, we have M_g Monte Carlo samples $\{\theta_{g,m}^{(t)}\}_{m=1}^{M_g}$ which enables us to directly approximate the distribution as a Mixture of Gaussian in left side of Eqn. 6, where σ is a hyperparameter and C is the normalization constant. Then we derive the prior gradient approximation as shown in the right side of Eqn. 6.

$$q_g^{(t+1)}(\theta_g) \approx C \exp\{\sum_{m=1}^{M_g} \frac{\|\theta_g - \theta_{g,m}^{(t)}\|_2^2}{2\sigma^2}\} \Rightarrow \nabla_{\theta_g} \log q_g^{(t+1)}(\theta_g) \approx \sum_{m=1}^{M_g} \frac{1}{\sigma^2}(\theta_g - \theta_{g,m}^{(t)}). \tag{6}$$

Partial Summation Approximation (PSA). From Eqn. 5, actually we can make an interesting observation that the prior gradient can be recursively unfolded as a summation over all historical GAN objective gradients, shown as:

$$\nabla_{\theta_g} \log q_g^{(t+1)}(\theta_g) = \nabla_{\theta_g} \mathcal{J}_g(\theta_g; D^{(t)}) + \nabla_{\theta_g} \log q_g^{(t)}(\theta_g) = \sum_{i=0}^{t} \nabla_{\theta_g} \mathcal{J}_g(\theta_g; D^{(i)}). \tag{7}$$

Therefore if we store all historical discriminator samples $\{\theta_{d,m}^{(i)}\}_{i=1,m=1}^{t,M_d}$, the prior gradient can be computed accurately via simple summation. Practically, computing gradients with all discriminator samples costs huge amount of storage and computational time, which is unaffordable. Hence we propose to maintain a subset of discriminators by subsampling the whole sequence of discriminators.

4 Theory

In this section, we first present the good convergence property of our Bayesian model. Second, we theoretically analyze the distribution evolution of Bayesian GAN (which will be referred as BGAN in the rest of paper) and make a comparison with our model. All proofs are delayed to the appendix (Section A) due to limited page.

4.1 Convergence Property of Our Model

We call a generator distribution is ideal if the generator following this distribution produces target data distribution. Theorem 1 states that any the ideal generator distribution is an equilibrium of the

dynamics defined in Eqn. 3. Although its mathematical proof evolves more elaboration, the idea behind is quite simple. When the generator distribution is ideal, discriminator should not be able distinguish the synthetic data from real. Thus averaged discriminator function will degenerated to a constant function. Then generator distribution will remain unchanged since discriminator essentially puts no preference between generators.

Theorem 1. *Assume the GAN objective and the discriminator space are symmetry. For any* ideal *generator distribution $q_g^*(\theta_g)$ satisfying $p^*_{model} \triangleq \mathbb{E}_{\theta_g \sim q_g^*}[p_{gen}(\cdot;\theta_g)] = p_{data}$, there exists a discriminator distribution q_d^* such that $D^*(\cdot) \triangleq \mathbb{E}_{\theta_d \sim q_d^*} D(\cdot;\theta_d) \equiv \frac{c}{2}$. Moreover, q_g^* and q_d^* is an equilibrium of the dynamic defined in Eqn. 3.*

4.2 Comparison with Bayesian GAN

In this paragraph, we first analyze the distribution evolution behind the BGAN algorithm which is uncovered in the original paper. Corollary 1 states our derivation where $p(\theta_g|\alpha_g), p(\theta_d|\alpha_d)$ are the predefined priors. In practice, BGAN use weakly informative Gaussian prior.

Corollary 1. *The Bayesian GAN algorithm actually performs distribution dynamics in Eqn. 8.*

$$q_g^{(t+1)} \propto \exp\{\mathbb{E}_{\theta_d \sim q_d^{(t)}} \mathcal{J}_g(\theta_g;\theta_d)\} p(\theta_g|\alpha_g), \quad q_d^{(t+1)} \propto \exp\{\mathbb{E}_{\theta_g \sim q_g^{(t)}} \mathcal{J}_d(\theta_d;\theta_g)\} p(\theta_d|\alpha_d). \tag{8}$$

Between our model (Eqn. 3) to BGAN (Eqn. 8), there are essentially two differences, the choice of prior and likelihood. Using weakly informative prior as BGAN do, actually causes problem. As we mentioned in last paragraph, intuitively, better the generator distribution is, less informative the discriminator is. A non-informative discriminator distribution plus a weakly informative prior will provide bad guidence to generator.

In terms of likelihood, BGAN uses *expectation of objective values* (e.g. $\mathbb{E}_{\theta_g \sim q_g^{(t)}} \mathcal{J}_d(\theta_d;\theta_g)$), while our model adopts *objective value of expectation* (e.g. $\mathcal{J}_d(\theta_d; \mathbb{E}_{\theta_g \sim q_g^{(t)}}[p_{gen}(\cdot;\theta_g)])$). We argue that computing the objective value of expectation makes more sense since our goal is to match the expectation (over generators) of generated data distribution to the target data distribution.

Compatibility Issue. We further show BGAN's choice of likelihood and prior leads theoretical issues. Specifically, BGAN is not suitable for any minimax-style GAN objective due to the incompatibility of its conditional posteriors. This problem may limit the usage of BGAN since many widely used GAN objective are in min-max fashion, such as the original GAN and WGAN.

Consider a simple case where we use only one Monte Carlo sample for the distributions $q_g^{(t)}$ and $q_d^{(t)}$. Then the distribution evolution in Eqn. 8 will degenerated as a Gibbs sampling process.

$$\begin{aligned} \theta_g^{(t+1)} &\sim q_g^{(t+1)}(\theta_g) \propto \exp\{\mathcal{J}_g(\theta_g;\theta_d = \theta_d^{(t)})\} p(\theta_g|\alpha_g) \\ \theta_d^{(t+1)} &\sim q_d^{(t+1)}(\theta_d) \propto \exp\{\mathcal{J}_d(\theta_d;\theta_g = \theta_g^{(t)})\} p(\theta_d|\alpha_d) \end{aligned} \tag{9}$$

Thus implicitly $\theta_g^{(t)}$ and $\theta_d^{(t)}$ are sampled from a joint distribution of θ_d and θ_g defined by the conditionals $p(\theta_g|\theta_d) = \exp\{\mathcal{J}_g(\theta_g;\theta_d)\} p(\theta_g|\alpha_g)$ and $p(\theta_d|\theta_g) = \exp\{\mathcal{J}_d(\theta_d;\theta_g)\} p(\theta_d|\alpha_d)$. However, our theoretical analysis shows that such a presumed joint distribution does not exist when $\mathcal{J}_d(\theta_d;\theta_g) = -\mathcal{J}_g(\theta_g;\theta_d)$. Specifically, Lemma 1 shows the existence of a joint distribution satisfying the conditionals in Eqn. 9 requires the GAN objective to be decomposable, i.e. $\exists \phi_g, \phi_d$, s.t. $\mathcal{J}_d(\theta_d;\theta_g) = \phi_g(\theta_g) + \phi_d(\theta_d)$. Apparently, no GAN objective is decomposable. Therefore, conditionals in Eqn. 9 are actually incompatible. Sampling with incompatible conditional distribution is problematic and leads unpredictable behavior (Arnold & Press, 1989).

Lemma 1. *Consider a joint distribution $p(x,y)$ of variable X and Y. Its conditional distributions can be represented in the forms of $p(x|y) \propto \exp\{L(x,y)\} q_x(x)$ and $p(y|x) \propto \exp\{-L(x,y)\} q_y(y)$* only if *$X$ and Y are independent, i.e., $p(x,y) = p(x)p(y)$ and $L(x,y)$ is decomposable, i.e., $\exists L_x$ and $L_y, L(x,y) = L_x(x) + L_y(y)$.*

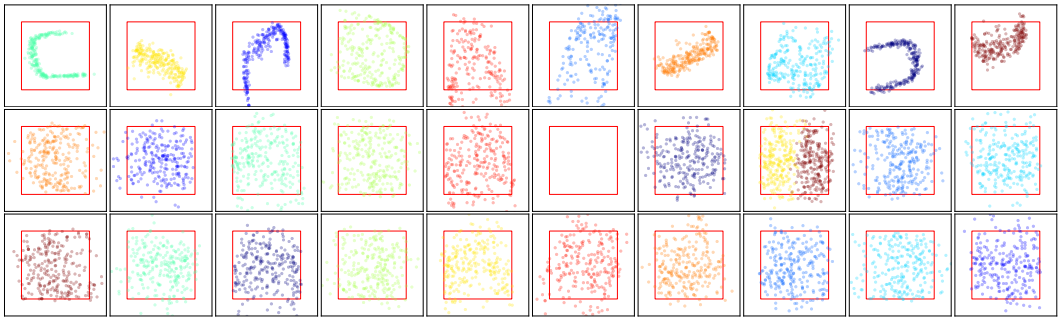


Figure 1: Visualization of projected hit sets. Rows from top to bottom correspond to MGAN, BGAN, and ours-PSA trained with GAN-MM objective. See Figure 3 (in appendix) for results of all models. In one row, projected hit sets for each mode are plotted in different panels, where the red boxes indicate real data regions $\mathcal{U}[-1,1]^2$. Different colors indicate samples from different generators.

5 Experiments

In this section, we evaluate our model with two inference algorithms proposed in Section 3.3 (denoted as *ours-GMA* and *ours-PSA*). We compare with three baselines: 1) *GAN (or DCGAN*[2]*)*: naively trained multiple generators in the vanilla GAN framework; 2) *MGAN*: Mixture GAN (Hoang et al., 2018) which is the start-of-art method to train GAN with multiple generators; 3) *BGAN*: Bayesian GAN (Saatci & Wilson, 2017).

For each model, we conduct thorough experiments under the four different GAN objectives introduced in Table 1 referred as GAN-MM, GAN-NS, WGAN and LSGAN here. For fair comparison, each model have the same number of generators with the same architecture. Discriminator architectures are also the same except for that of MGAN which has an additional branch of classifier. To facilitate reproducibility, we report implementation and experiment details in Section C of the appendix.

5.1 High-dimensional Multi-modal Synthetic Dataset

Dataset. Consider learning a data distribution in a high dimensional space $\mathcal{X} = \mathbb{R}^D$, which is a uniform mixture of n modes. Each mode lies on a d-dimensional sub-space of $\mathcal{X}$. We call this d-dimensional sub-space as mode-space of the i-th mode. Specifically, the data of the i-th mode is generated by the following process,

$$\mathbf{z} \sim \mathcal{U}[-1,1]^d, \quad \mathbf{x} = \mathbf{A}_i(\mathbf{z} + \mathbf{b}_i), \quad \mathbf{A}_i \sim \mathcal{N}(0, \sigma_A^2 I_{D\times d}), \quad \mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, \sigma_b^2 I_d) \tag{10}$$

In our experiment, n, D, and d are set to 10, 100, and 2. Hyper-parameters for $\mathbf{A}$ and $\mathbf{b}$ are set to be $\sigma_A = \sigma_b = 5$. Each model train ten generators.

Metric. We define *projection error* ϵ_p for generated data sample $\mathbf{x}$ as the minimum of Euclidean distance from $\mathbf{x}$ to any of mode-spaces i.e. $\epsilon_p(\mathbf{x}) = \min_i \epsilon_i(\mathbf{x}) \triangleq \|\mathbf{x} - \mathbf{A}_i(\mathbf{A}_i^T\mathbf{A}_i)^{-1}\mathbf{A}_i^T\mathbf{x}\|_2$. Then we set threshold [3] η to judge the belonging of $\mathbf{x}$, i.e. only the data whose Euclidean distance to the mode-space lowers than η is considered as belonging to that mode. The trained models are evaluated via the samples $\{\mathbf{x}_k\}_{k=1}^K \sim p_{model}$ it generates. We define *hit set* $\mathcal{H}_i \triangleq \{\mathbf{x}_k | \epsilon_i(\mathbf{x}_k) < \eta\}$ to indicate the samples belong to each mode. We further define *projected hit set*, $\mathcal{PH}_i \triangleq \{(\mathbf{A}_i^T\mathbf{A}_i)^{-1}\mathbf{A}_i^T\mathbf{x} - \mathbf{b}_i | \mathbf{x} \in \mathcal{H}_i\}$ to project data in each hit set back to the canonical low dimensional space.

Now we introduce three evaluation metrics: *hit ratio*, *hit error*, and *cover error*. *Hit ratio* $\mathcal{H}_r \triangleq \sum_{i=1}^n |\mathcal{H}_i| / K$ is the percentage of generated data belonging to any of the real data mode. *Hit error* $\mathcal{H}_e \triangleq \sum_{i=1}^n \sum_{\mathbf{x}\in\mathcal{H}_i} \epsilon_i(\mathbf{x}) / \sum_{i=1}^n |\mathcal{H}_i|$ is the averaged projection error of the data. Lastly, *cover error* $\mathcal{C}_e$ evaluates how well the generated data covers each mode. Essentially it computes the KL-divergence between the estimated distribution of samples in $\mathcal{PH}_i$ and the uniform distribution over $[-1,1]^d$. Formally, it is defined as the averaged KL-divergence on n modes i.e. $\mathcal{C}_e \triangleq \frac{1}{n}\sum_{i=1}^n \mathrm{KL}(\hat{p}(\cdot;\mathcal{PH}_i) \| \mathcal{U}[-1,1]^d)$. The intuition is that if data generated is close to the ground truth distribution, they should be uniformly distributed in the square area of each mode.

[2] We use original GAN for synthetic dataset and DCGAN for image generation task

[3] Based on the fact that the average distance between the data from two different modes is 800, we set a threshold of $\eta = 40$.

Table 2: Hit ratios ($\mathcal{H}_r$), hit errors ($\mathcal{H}_e$), cover errors ($\mathcal{C}_e$) of different models with different GAN objectives. Note, if the model failed to capture all the modes of real data, by definition the cover error is ∞. In that case, we report the averaged KL-divergence on modes captured by the model in brackets.

	$\mathcal{H}_r$ (HIGHER IS BETTER), $\mathcal{H}_e$ (LOWER IS BETTER)				$\mathcal{C}_e$ (LOWER IS BETTER)			
	GAN-MM	GAN-NS	WGAN	LSGAN	GAN-MM	GAN-NS	WGAN	LSGAN
GAN	0.86, 22.6	0.85, 23.1	0.78, 26.7	0.74, 23.1	12.11	8.86	7.20	∞ (12.07)
MGAN	0.82, 24.2	0.84, 25.5	0.67, 31.7	0.81, 23.6	5.46	6.31	5.00	∞ (4.25)
BGAN	1.0, **5.5**	1.0, **6.4**	1.0, **12.1**	1.0, 6.3	∞ (1.73)	1.76	4.32	1.80
OURS-GMA	1.0, 7.4	1.0, 7.7	1.0, 15.5	1.0, **5.3**	1.84	**1.73**	3.01	1.79
OURS-PSA	1.0, 5.8	1.0, **6.4**	1.0, 12.5	1.0, 6.4	**1.75**	1.75	**2.28**	**1.74**

Optimization-Based v.s. Probabilistic. The left part of Table 2 summarizes the results in terms of *hit ratio* and *hit error*. Probabilistic methods including our algorithms and BGAN always achieve a hit ratio of 1, which means every data point generated from these models is very close to one mode of the target distribution. On the other hand, optimization based methods, both GAN and MGAN, consistently have a significantly larger hit error, and sometimes may even generate data samples that do not belong to any mode. Moreover, the data distribution generated by the optimization-based methods fits the target uniform distribution much worse than its probabilistic counterparts, which is quantitatively reflected by the cover error showed in the right side of Table 2 and visually demonstrated by the projected hit sets in Figure 1. According to the visualization, data generated by GAN or MGAN tend to be under dispersed and hardly cover the whole square region of the true mode, while data generated by probabilistic methods aligns much better with the ground truth distribution. We attributes this superiority to stronger exploration power in the generator space coming from the randomness in probabilistic methods.

Bayesian GAN v.s. Our Methods. The incompatibility issue of BGAN with minimax-style GAN objectives theoretically derived in Section 4.2 is empirically verified in our experiments. As visualized in Figure 1, with the GAN-MM objective, BGAN is trapped in a local equilibrium and fails in capturing one mode of the true data. Besides, as shown in Table 2, BGAN with the WGAN objective achieves much poorer coverage than with other GAN objectives, while our model is much more robust to the choice of GAN objectives (consistently lower cover errors). A qualitatively comparison is made in Figure 6 (in appendix) which shows the data distribution generated by BGAN trained with WGAN objective tends to shrink. More visual illustrations under different GAN objectives are placed in Section D of the appendix.

5.2 NATURAL IMAGE DATASET

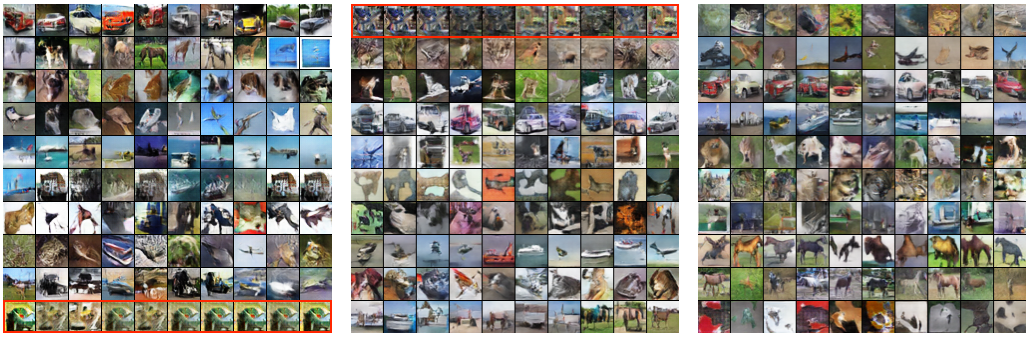
Datasets. CIFAR-10 is a well-understood benchmark dataset (Krizhevsky et al. (2010)) with 50k training and 10k test 32x32 RGB images. It contains 10 classes of images: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. In our experiments, we use all 50k training samples to train and evalute models.

Evaluation Protocols. We make a thorough comparison for all methods with different GAN objectives under two widely adopted metrics: *Inception Score* (Salimans et al. (2016)) and *Fréchet Inception Distance* (Heusel et al. (2017)). Inception Score computes $\exp(\mathbb{E}_{\mathbf{x}}[KL(p(y|\mathbf{x})\|p(y))])$ where $p(y|\mathbf{x})$ standards is predicted label distribution by a pretrained Inception model (Szegedy et al. (2015)). While Inception Score (IS) reflects the fideality and diversity of images and is known to be well-correlated with human judgment (Salimans et al. (2016)), it does not measure the closeness between real data distribution and synthetic data. Therefore, we employ Fréchet Inception Distance (FID). To compute it, we embed real and synthetic images to the last hidden layer of Inception model and evaluate the Fréchet distance between two distributions in the embedding space. We delay details of computing these metrics into appendix.

Model Architectures. Inspired by Hoang et al. (2018), we adapted the parameter sharing technique for both generators and discriminators. Specifically, generators (of one model) are not disjoint neural networks. They only differ at the first layer and sharing all parameters at the following layers. This tied parameter design reduced the model complexity and boost the performance. Especially for probabilistic model such as Bayesian GAN, we observe significant enhancement of generated image qualities compare to the original result (Figure 7 in Saatci & Wilson (2017)). More implementation details such as hyper-parameters of network layers are delayed to Section C in the appendix.

Table 3: Inception score and FID results of all models with four different GAN objectives.

	INCEPTION SCORES (HIGHER IS BETTER)				FIDS (LOWER IS BETTER)			
	GAN-MM	GAN-NS	WGAN	LSGAN	GAN-MM	GAN-NS	WGAN	LSGAN
DCGAN	6.85	7.30	7.02	7.23	20.54	19.75	21.37	21.54
MGAN	6.92	7.05	6.88	6.69	21.24	20.85	22.22	23.47
BGAN	7.12	7.33	**7.08**	7.37	24.00	**19.34**	23.26	22.63
OURS-PSA	**7.30**	**7.35**	7.06	**7.40**	**19.20**	19.83	**19.50**	**20.02**



(a) MGAN (Epoch 250) (b) BGAN (Epoch 250) (c) ours-PSA (Epoch 250)

Figure 2: Images generated by MGAN, BGAN, our-PSA model trained on CIFAR 10 with GAN-NS objective. The tenth generator in Figure 2(a) and the first in Figure 2(b) collapse while generators trained by our method have no such issue. DCGAN baseline also have this issue, see the Figure 7 in appendix. Note that, mode collapse also happens when baseline models trained with other GAN objectives.

Quantitative Results. We train every model 500 epochs. We denote the epoch delivering highest IS-0.1FID as the *best epoch* and report the scores of the model at this epoch. The results are summarized in Table 3. Overall, our proposed model outperforms all baselines under all GAN objectives. Generally, probabilistic methods achieves better scores then optimization based methods which indicates injecting stochasticity into GAN training helps generating more multi-modal images. Moreover, we would highlight that Bayesian GAN is observed to have a visible performance drop when accompany with min-max style GAN objectives, which provides another empirically evidence for our theory analysis in Section 4.2. While our Bayesian method fits any GAN objectives and constantly performs well.

Qualitative Results. Figure 2 display the samples randomly generated by baselines and our model. Every row in the figure represents samples of one learned generator. As Figure 2 and Figure 7 shows, all three baselines suffer from mode collapses problem. Hoang et al. (2018) already notice that mode collapse in one of the generators could happen after a roughly long training procedure (around 250 epochs). Our experiment shows Bayeisan GAN also have this problem. However, our novel Bayesian model is robust to mode collapse. Specifically, mode collapse occurs to our model only once in a total of over ten trials. In contrast, mode collapse happens five times more frequently in baseline methods. Section E in appendix contains more images generated by all models at their best epochs. Actually, for the baseline models, their best epoch always appear before epoch 250. In another word, training too much epochs leads performance drop to those models which indicates their instability near the equilibrium.

6 DISCUSSION

In this paper, we propose a novel Bayesian modelling framework for GAN, with a likelihood function establishing a connection to existing GAN models and a novel prior stabilizing the inference process. We propose scalable Bayesian inference algorithms which are asymptotically correct. As future work, we plan to extend the proposed framework to non-parametric Bayesian modelling and investigate more theoretical properties of GANs in the Bayesian context. Developing Bayesian generalization for deep learning models is not a recent idea and happens in many fields other than generative models. By this work, we emphasize the importance of going beyond the intuition and understanding theoretical behavior of Bayesian model. We hope that our work helps inspire continued exploration into Bayesian deep learning from a more rigorous perspective.

REFERENCES

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

Barry C Arnold and S James Press. Compatible conditional distributions. *Journal of the American Statistical Association*, 84(405):152–156, 1989.

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.

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

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International Conference on Machine Learning*, pp. 1683–1691, 2014.

Arnab Ghosh, Viveka Kulharia, Vinay Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. *arXiv preprint arXiv:1704.02906*, 2017.

Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

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*, pp. 2672–2680, 2014.

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*, pp. 6626–6637, 2017.

Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. Mgan: Training generative adversarial nets with multiple generators, 2018.

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

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 2010. URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, pp. 3, 2013.

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)*, pp. 2813–2821. IEEE, 2017.

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

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

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

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*, pp. 2234–2242, 2016.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

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*, pp. 5430–5439, 2017.

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

A OMITTED PROOFS

Theorem 1. This theorem is general and holds when the GAN objective and the discriminator space have symmetry. The symmetry of GAN objective means its functions ϕ_1 and ϕ_2 satisfy that $\exists c \in \mathbb{R}, \forall x \in \mathbb{R}, \phi_1(x) \equiv \phi_2(c-x)$. While the symmetry of discriminator space Θ_d indicates that for any $\theta_d \in \Theta_d$, there is a $\theta'_d \in \Theta_d$ such that $D(\mathbf{x};\theta_d) \equiv c - D(\mathbf{x};\theta'_d)$. Note that the symmetry condition are very weak, first it holds for all the common choices of GAN objectives such as those listed in Table 1. Second, it holds for neural network which is the most common parameterization for discriminator in practice.

Proof.

$$\begin{aligned} q_d^*(\theta_d) &\propto \exp(\mathbb{E}_{\mathbf{x}\sim p_{data}}[\phi_1(D(\mathbf{x};\theta_d))] + \mathbb{E}_{\mathbf{x}\sim p^*_{model}}[\phi_2(D(x;\theta_d))]) \\ &= \exp(\mathbb{E}_{\mathbf{x}\sim p_{data}}[\phi_1(D(\mathbf{x};\theta_d)) + \phi_2(D(x;\theta_d))]) \\ &= \exp(\mathbb{E}_{\mathbf{x}\sim p_{data}}[\phi_2(c - D(\mathbf{x};\theta_d)) + \phi_1(c - D(x;\theta_d))]) \\ &= \exp(\mathbb{E}_{\mathbf{x}\sim p_{data}}[\phi_2(D(\mathbf{x};\mathcal{S}(\theta_d))) + \phi_1(D(x;\mathcal{S}(\theta_d)))]) \\ &\Rightarrow q_d^*(\theta_d) = q_d^*(\mathcal{S}(\theta_d)), \end{aligned} \tag{11}$$

$$\begin{aligned} D^*(\mathbf{x}) = \mathbb{E}_{\theta_d\sim q_d^*} D(\mathbf{x};\theta_d) &= \int_{\theta_d\in\Theta_d} q_d^*(\theta_d) D(\mathbf{x};\theta_d) d\theta_d \\ &= \frac{1}{2}(\int_{\theta_d\in Td} q_d^*(\theta_d) D(\mathbf{x};\theta_d) d\theta_d + \int_{\theta'_d=\mathcal{S}(\theta_d)\in Td} q_d^*(\theta'_d) D(\mathbf{x};\theta'_d) d\theta'_d) \\ &= \frac{1}{2}\int_{\theta_d\in\Theta_d} q_d^*(\theta_d)(D(\mathbf{x};\theta_d) + D(\mathbf{x};\mathcal{S}(\theta_d))) d\theta_d \\ &= \frac{1}{2}\int_{\theta_d\in\Theta_d} q_d^*(\theta_d)\cdot c \cdot d\theta_d = \frac{c}{2}. \end{aligned} \tag{12}$$

Eqn. 11 and Eqn. 12 prove that $q_d^*(\theta_d) \propto \exp(\mathcal{J}_d(\theta_d; p^*_{model}))$ satisfies $\mathbb{E}_{\theta_d\sim q_d^*} D(\cdot;\theta_d) \equiv \frac{c}{2}$. Note that, in the above equations, $\mathcal{S}(\theta_d)$ denotes the symmetric discriminator of θ_d satisfying $D(\mathbf{x};\mathcal{S}(\theta_d)) \equiv c - D(\mathbf{x};\theta_d)$. Hence according Eqn. 13, we know $\mathcal{J}_g(\theta_g; D^*)$ is a constant.

$$\mathcal{J}_g(\theta_g; D^*) = \mathbb{E}_{\mathbf{x}\sim p_{gen}(\cdot;\theta_g)}[\phi_3(D^*(\mathbf{x}))] = \phi_3(\frac{c}{2}), \quad \forall \mathbf{x} \in \mathcal{X}. \tag{13}$$

Thus the generator distribution will not change based on the dynamics in Eqn. 3 since $q_g^*(\theta_g) \propto \exp\{\mathcal{J}_g(\theta_g; D^*)\} q_g^*(\theta_g)$. □

Corollary 1

Proof.

$$\log q_g^{(t+1)}(\theta_g) = \frac{1}{J_g J_d}\sum_{i=1}^{J_g}\sum_{k=1}^{J_d} \log p(\theta_g|\mathbf{z}^{(i)}, \theta_d^k) \tag{14}$$

$$\log q_g^{(t+1)}(\theta_g) \simeq \mathbb{E}_{\theta_d\sim q_d(\theta_d)} \mathcal{J}_g(\theta_g;\theta_d) + \log p(\theta_g|\alpha_g) \tag{15}$$

Algorithm 1 from the original paper (Saatci & Wilson, 2017) implies that Eqn. 14 where $\theta_d^k \sim q_d^{(t)}(\theta_d)$ are Monte Carlo samples of discriminator and every $\mathbf{z}^{(i)}$ is a mini-batch containing n_g noise samples. Be definition $p(\theta_g|\mathbf{z}^{(i)}, \theta_d^k) \propto \prod_{j=1}^{n_g} D(G(\mathbf{z}_j^{(i)};\theta_g);\theta_d^k) p(\theta_g|\alpha_g)$ (Eqn. 1 in the original paper), each term[4] $\log p(\theta_g|\mathbf{z}^{(i)}, \theta_d^k) = \sum_{j=1}^{n_g} \phi_3(D(G(\mathbf{z}_j^{(i)};\theta_g);\theta_d^k)) + \log p(\theta_g|\alpha_g)$. Hence, the total summation

[4]Note that in BGAN paper, the GAN objective is GAN-NS. Thus ϕ_3 equals $\log(\cdot)$.

is a Monte Carlo approximation of the expectation in the right side of Eqn. 15. The same derivation can be done to $q_d^{(t+1)}$. Together, we get Corollary 1. □

Lemma 1

Proof.

$$
\begin{aligned}
& p(x|y) = \alpha(y)\exp\{L(x,y)\}q_x(x), \\
& p(y|x) = \beta(x)\exp\{-L(x,y)\}q_y(y), \\
\Longrightarrow\ & p(x,y)^2 = p(x|y)p(y) \times p(y|x)p(x) \\
& \quad = p(x)p(y)\alpha(y)\beta(x)q_x(x)q_y(y) \\
\Longrightarrow\ & X, Y \text{ are independent.} \\
\Longrightarrow\ & p(x) = p(x|y) \\
\Longrightarrow\ & L(x,y) = \log p(x) - \log q_x(x) - \log \alpha(y) \\
\Longrightarrow\ & L(x,y) \text{ is decomposable.}
\end{aligned} \tag{16}
$$

where $\alpha(y) = (\int \exp\{L(x,y)\}q_x(x)dx)^{-1}$ and $\beta(x) = (\int \exp\{-L(x,y)\}q_y(y)dy)^{-1}$. □

B INFERENCE ALGORITHM

Stochastic Gradient Hamiltonian Monte Carlo (Chen et al., 2014) is a gradient based MCMC sampling method. It use noise gradient estimation of potential function to generate sample from a given distribution. Our inference algorithm (algorithm 1) is based on this brilliant algorithm.

Input: Initial Monte Carlo samples of $\{\theta_{d,m}^{(0)}\}_{m=1}^{M_d}$ and $\{\theta_{g,m}^{(0)}\}_{m=1}^{M_g}$, learning rate η, SGHMC noise factor α, number of updates in SGHMC procedure L, number of updating iterations T.
for $t = 1$ **to** T **do**
  for $m = 1$ **to** M_d **do**
    $\theta_{d,m} \leftarrow \theta_{d,m}^{(t)}$
    for $l = 1$ **to** L **do**
      $\mathbf{v} \leftarrow (1-\alpha)\mathbf{v} + \eta\nabla_{\theta_d}\log q_d^{(t+1)}(\theta_{d,m}) + \mathbf{n}; \mathbf{n} \sim \mathcal{N}(0, 2\alpha\eta I)$
      $\theta_{d,m} \leftarrow \theta_{d,m} + \mathbf{v}$
    end for
    $\theta_{d,m}^{(t+1)} \leftarrow \theta_{d,m}$
  end for
  for $m = 1$ **to** M_g **do**
    $\theta_{g,m} \leftarrow \theta_{g,m}^{(t)}$
    for $l = 1$ **to** L **do**
      $\mathbf{v} \leftarrow (1-\alpha)\mathbf{v} + \eta\nabla_{\theta_g}\log q_g^{(t+1)}(\theta_{g,m}) + \mathbf{n}; \mathbf{n} \sim \mathcal{N}(0, 2\alpha\eta I)$
      $\theta_{g,m} \leftarrow \theta_{g,m} + \mathbf{v}$
    end for
    $\theta_{g,m}^{(t+1)} \leftarrow \theta_{g,m}$
  end for
end for

Algorithm 1: Our Adapted SGHMC Inference Algorithm

C EXPERIMENT DETAILS

We implement our model and baselines and conduct experiments in PyTorch (Paszke et al., 2017) except for the MGAN model in CIFAR-10 experiment where we adapt its official tensorflow implementation[5].

[5]MGAN Code: https://github.com/qhoangdl/MGAN

C.1 HIGH-DIMENSIONAL MULTI-MODAL SYNTHETIC DATASET

Model Architecture: Each generator or discriminator is a three layer perceptron. For the generator, the dimensions of input, hidden layer and output are 10, 1000, and 100 respectively. For the discriminator, the dimensions of input, hidden, output layers are 100, 1000, and 1. All activation functions are leaky ReLU(Maas et al., 2013).

Training hyperparameters: All models are optimized by Adam (Kingma & Ba, 2014) with a learning rate of 10^{-4}. For probabilistic methods, the SGHMC noise factor (α in algorithm 1) is set as 10^{-1}.

C.2 CIFAR-10

Implementation of inception score. As Barratt & Sharma (2018) pointed out, Inception Score is sensitive to the inception model used and number of data splits. We follow the convention of using open-sourced tensorflow inception model [6] and setting split number to ten.

Model architecture: Generator architecture has four deconvolution layers (kernel size 4, stride 2) with the following input, hidden feature-maps, output size: 100x1x1 $\rightarrow$ 512x4x4 $\rightarrow$ 256x8x8 $\rightarrow$ 128x16x16 $\rightarrow$ 3x32x32. Every deconvolution layer is followed by batch-normalization layer and Relu activation except for the last deconvolution layer who is followed by Tanh activation.

Discriminator architecture has four convolution layers (kernel size 5, stride 2) with the following input, hidden feature-maps, output size: 3x32x32 $\rightarrow$ 128x16x16 $\rightarrow$ 256x8x8 $\rightarrow$ 512x4x4 $\rightarrow$ 1x1x1. Batch-normalization is applied to each layer except the last one. Activation are leaky-ReLU.

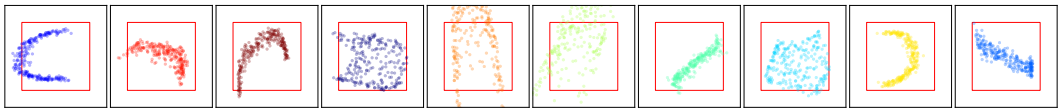
In our experiments, each model are trained with 10 generators. As for discriminator, DCGAN and MGAN has one discriminator while probabilistic models (BGAN and ours) have 4 discriminators (i.e. 4 Monte Carlo samples from discriminator distribution).

Training hyperparameters: All models are optimized by Adam(Kingma & Ba, 2014) with a learning rate of 2×10^4. For probabilistic methods, the SGHMC noise factor is set as 3×10^2. Following the configuration in MGAN, the batch size of generators and discriminators are 120 and 64. Note that, since probabilistic model has 10 generator Monte Carlo samples and 4 discriminators, indeed batch size for every generator and discriminator is 12 and 16 respectively.

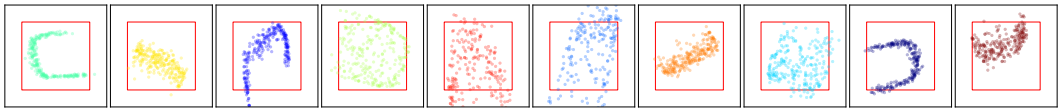
[6] Code for computing Inception Score: https://github.com/tsc2017/inception-score. Code for computing Fréchet Inception Distance : https://github.com/mseitzer/pytorch-fid.

D A FULL VISUALIZATION OF PROJECTED HIT SETS

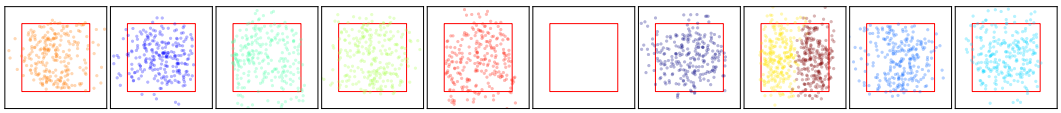
In this section, we shows projected hit sets for all models under different GAN objectives.



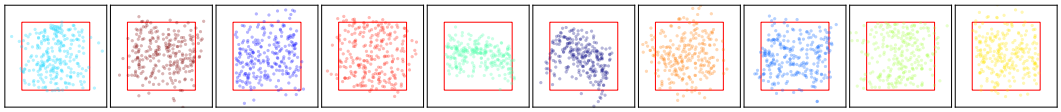
GAN + GAN-MM



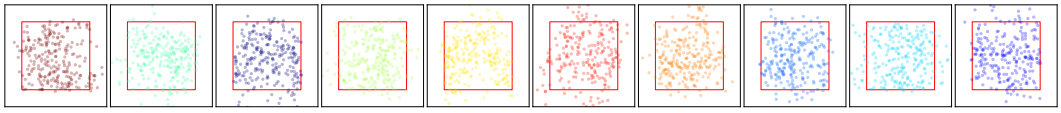
MGAN + GAN-MM



BGAN + GAN-MM



Ours-GMA + GAN-MM



Ours-PSA + GAN-MM

Figure 3: Visualization of the projected hit sets of all models trained with GAN-MM objective. The top two rows show the results of optimization based methods.The bottom three rows present probabilistic method results. In each row, projected hit sets for each mode are plotted in different panels. The red boxes in each panel indicate the region $\mathcal{U}[-1,1]^2$ where the target data uniformly distributed. The data points produced by different generators of a model is painted with different colors.

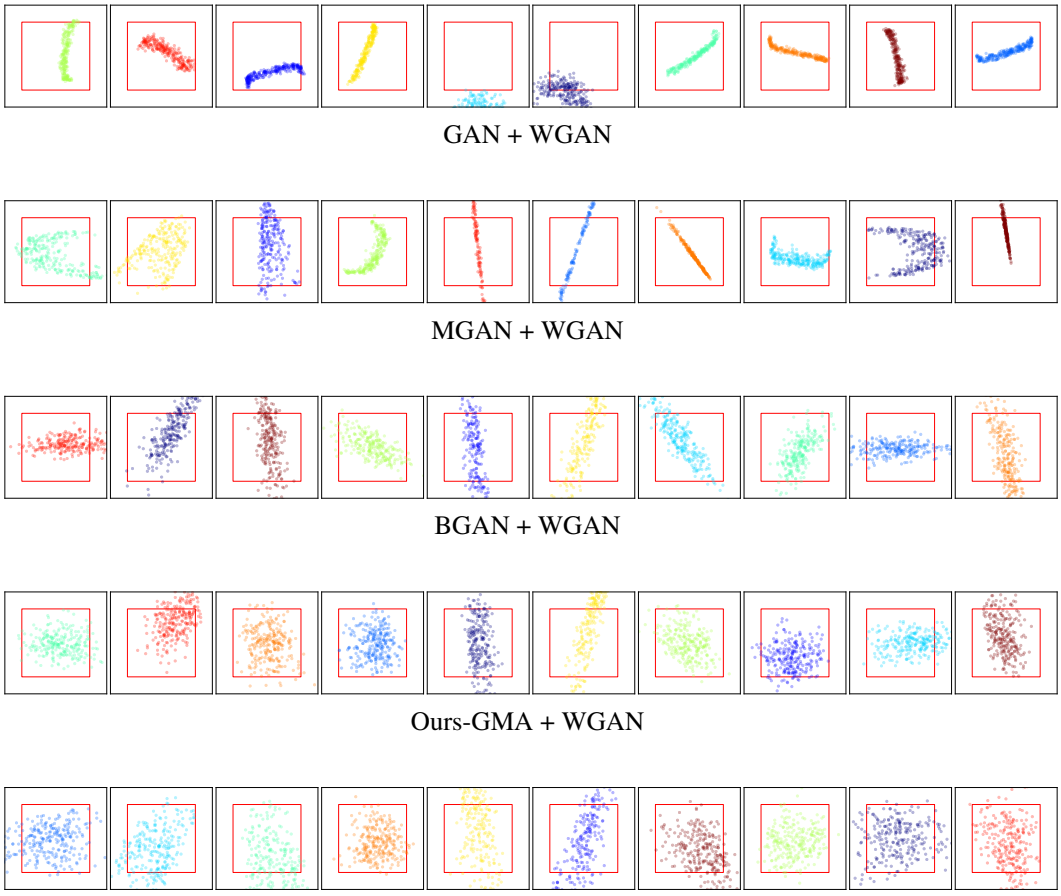


Figure 4: Visualization of the projected hit sets of different models trained with the WGAN objective. As we can see, training the WGAN objective leads to much worse performance for both optimization-based methods and BGAN. On the other hand, our methods are robust to the choice of different GAN objectives and do not suffer from significant performance drop when using the WGAN objective.

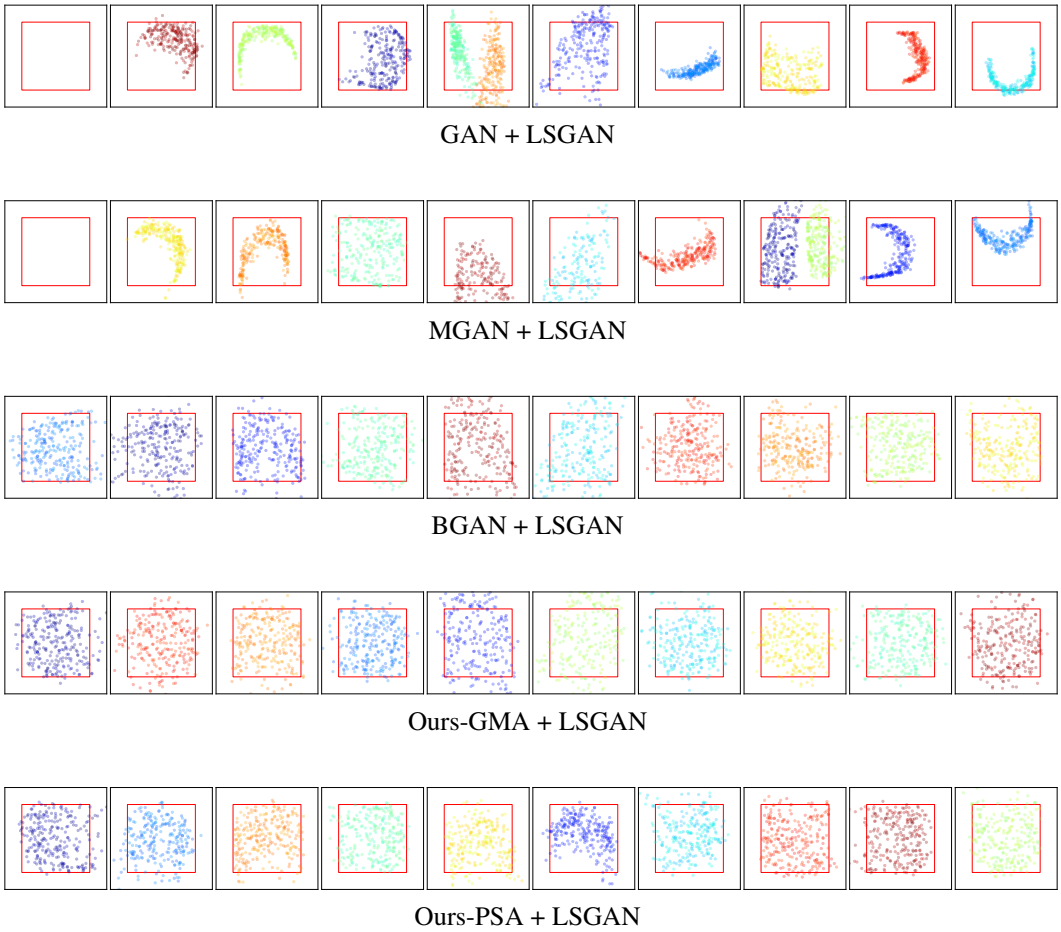


Figure 5: Visualization of the projected hit sets of different models trained with the LSGAN objective. Three probabilistic models performs perfectly in this case, while both the two optimization-based methods miss one mode of the true distribution. This experiment illustrates that although MGAN employs an additional classifier to force the data generated by different generators to be disjoint, it still suffers from mode collapsing problem. This is because in MGAN, generators may still generate disjoint data samples in the same mode and fail in capturing other modes.

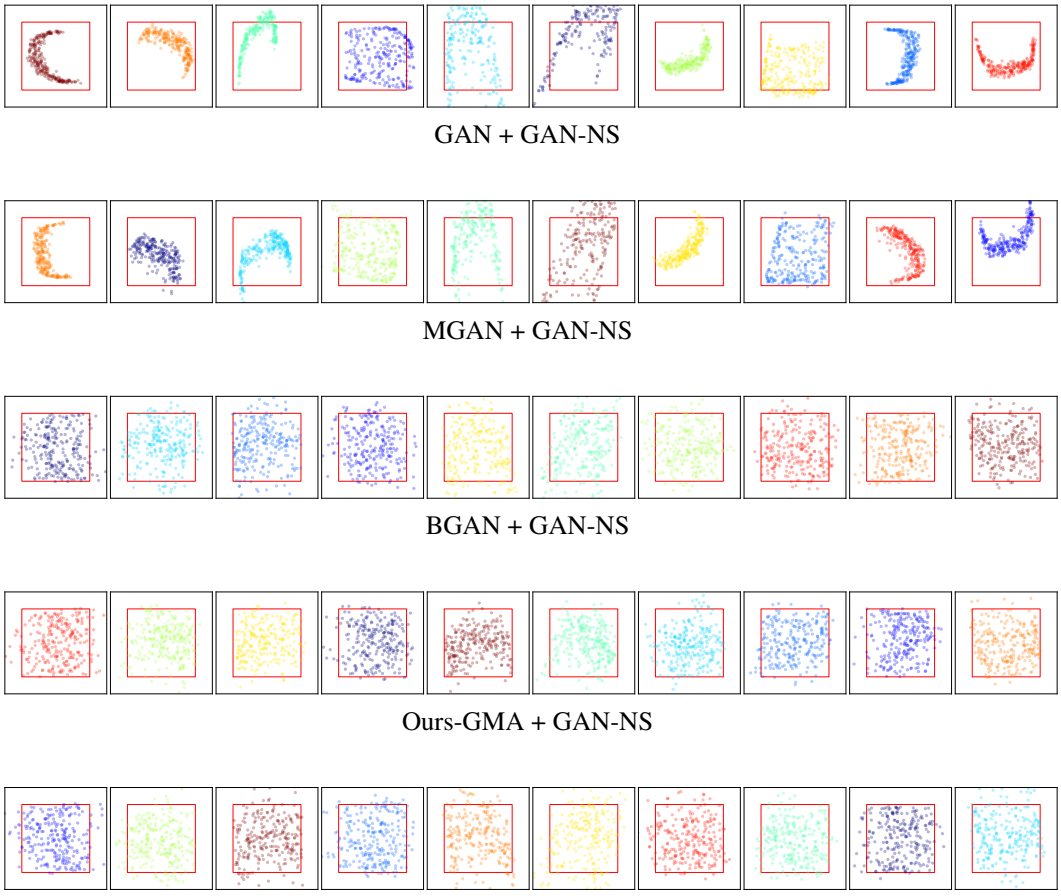


Figure 6: Visualization of the projected hit sets of different models trained with the GAN-NS objective. All the models succeed in fitting each mode of true distribution with one of their generator. Specifically, three probabilistic models generate data almost perfectly covering the ground-truth 'squares' while the optimization-based methods have difficulty covering the whole 'squares' and tend to yield under-dispersed data distributions. Note that since the GAN-NS objective is not in a min-max style, the success of BGAN is expected.

E MORE CIFAR-10 GENERATED IMAGE RESULTS

In this section, we shows images generated by all models trained on CIFAR-10 under different GAN objectives.

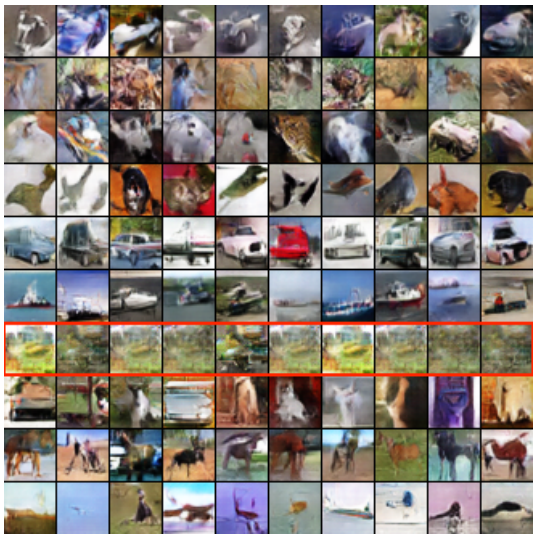


Figure 7: Images generated by **DCGAN** trained on CIFAR-10 with **GAN-NS** objective at epoch 250. Redbox indicates the collapsed generator.

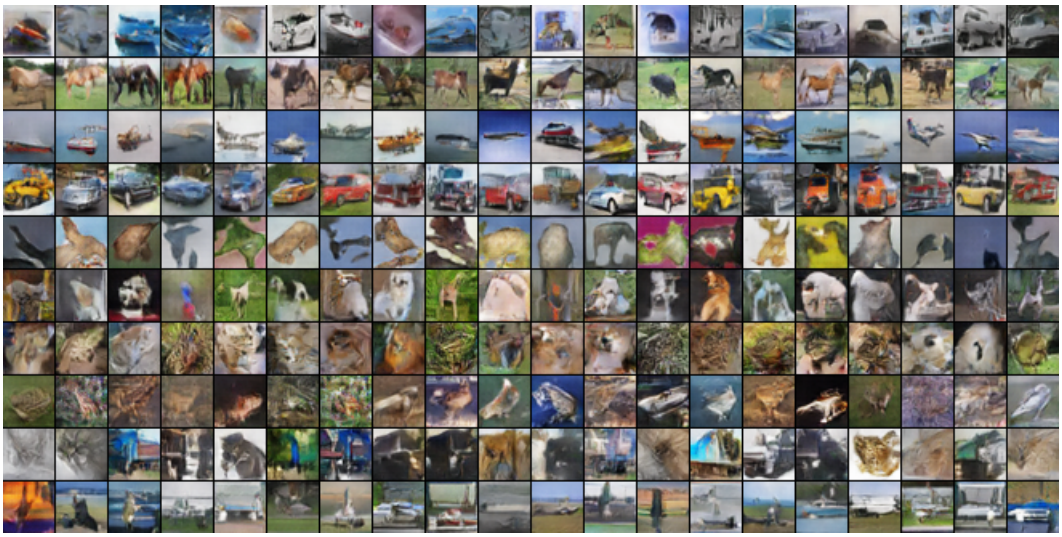


Figure 8: Images generated by **ours-PSA model** trained on CIFAR-10 with **GAN-NS** objective.Images in different rows are generated by different generators.

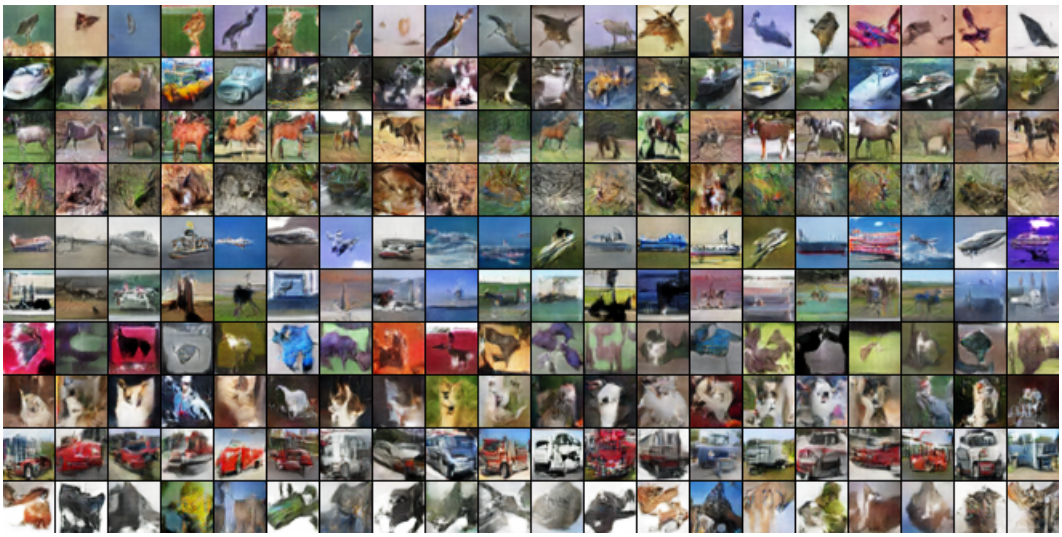


Figure 9: Images generated by **ours-PSA model** trained on CIFAR-10 with **LSGAN** objective.Images in different rows are generated by different generators.

Figure 10: Images generated by **ours-PSA model** trained on CIFAR-10 with **GAN-MM** objective.Images in different rows are generated by different generators.

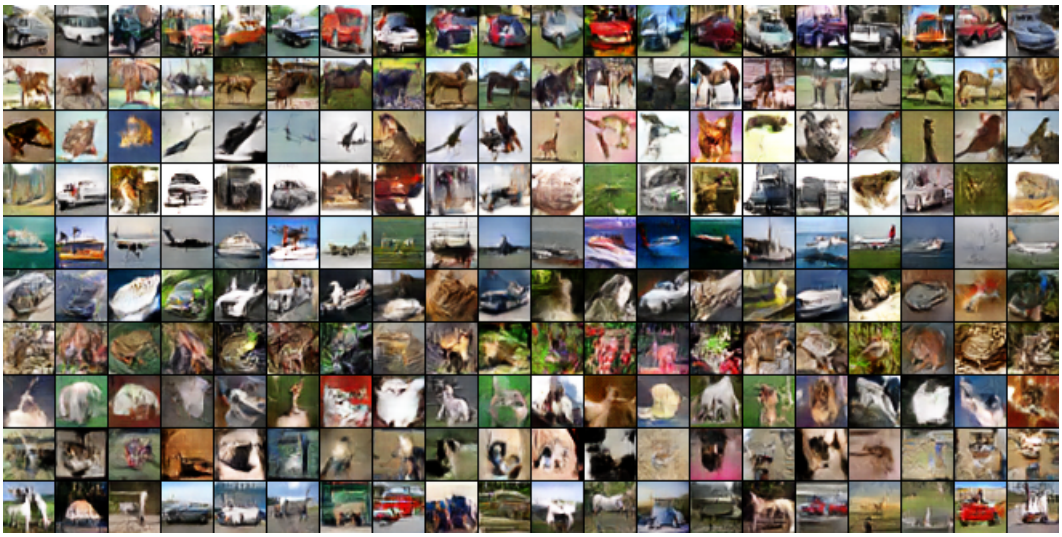


Figure 11: Images generated by **ours-PSA model** trained on CIFAR-10 with **WGAN** objective.Images in different rows are generated by different generators.

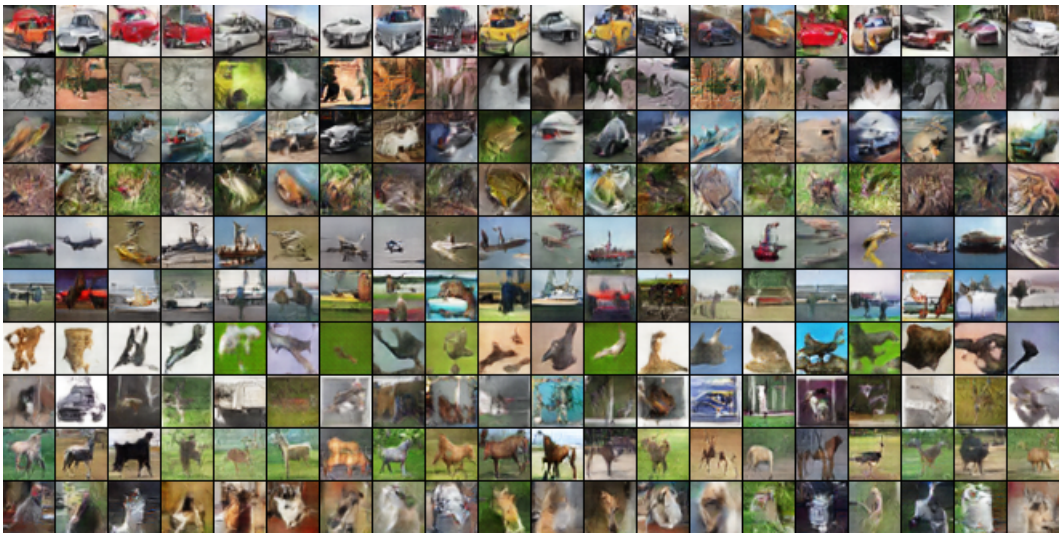


Figure 12: Images generated by **Bayesian GAN** trained on CIFAR-10 with **GAN-NS** objective.Images in different rows are generated by different generators.

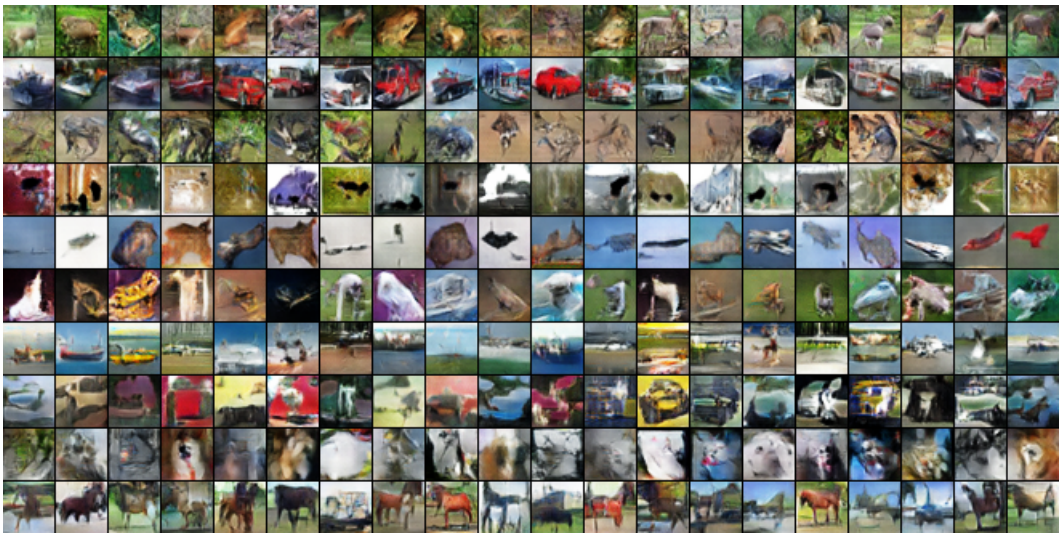


Figure 13: Images generated by **Bayesian GAN** trained on CIFAR-10 with **LSGAN** objective.Images in different rows are generated by different generators.

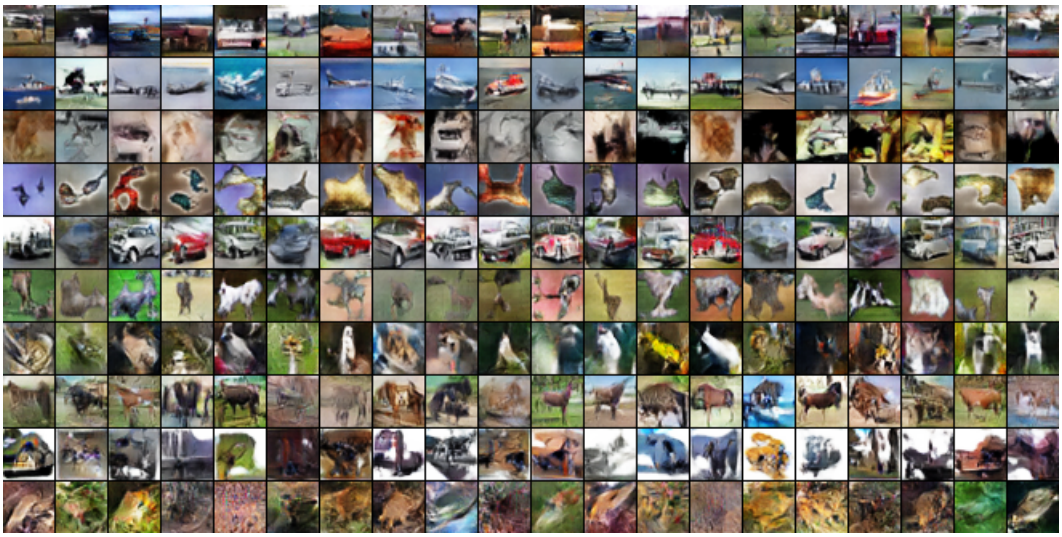


Figure 14: Images generated by **Bayesian GAN** trained on CIFAR-10 with **GAN-MM** objective.Images in different rows are generated by different generators.

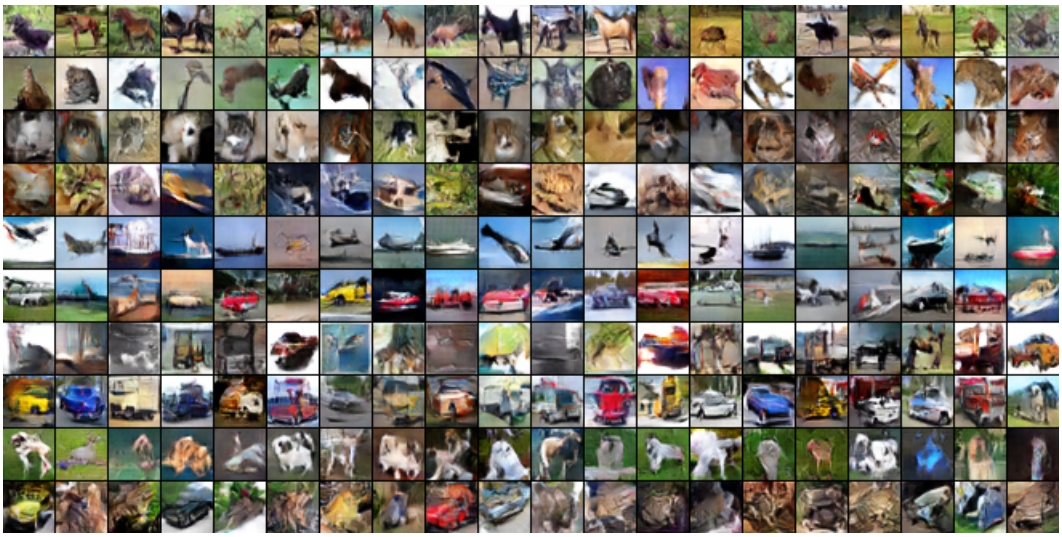


Figure 15: Images generated by **Bayesian GAN** trained on CIFAR-10 with **WGAN** objective.Images in different rows are generated by different generators.

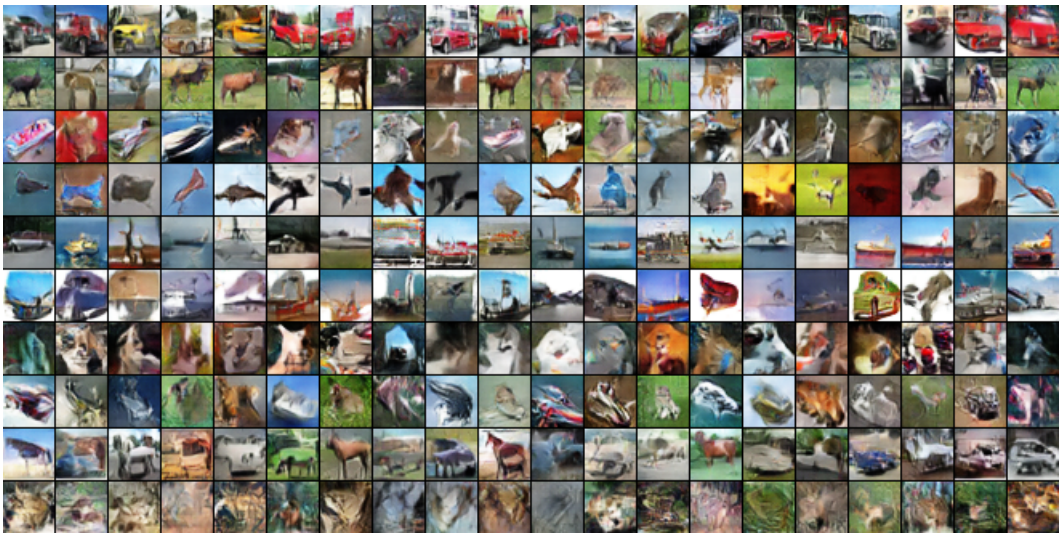


Figure 16: Images generated by **MGAN** trained on CIFAR-10 with **GAN-NS** objective.Images in different rows are generated by different generators.

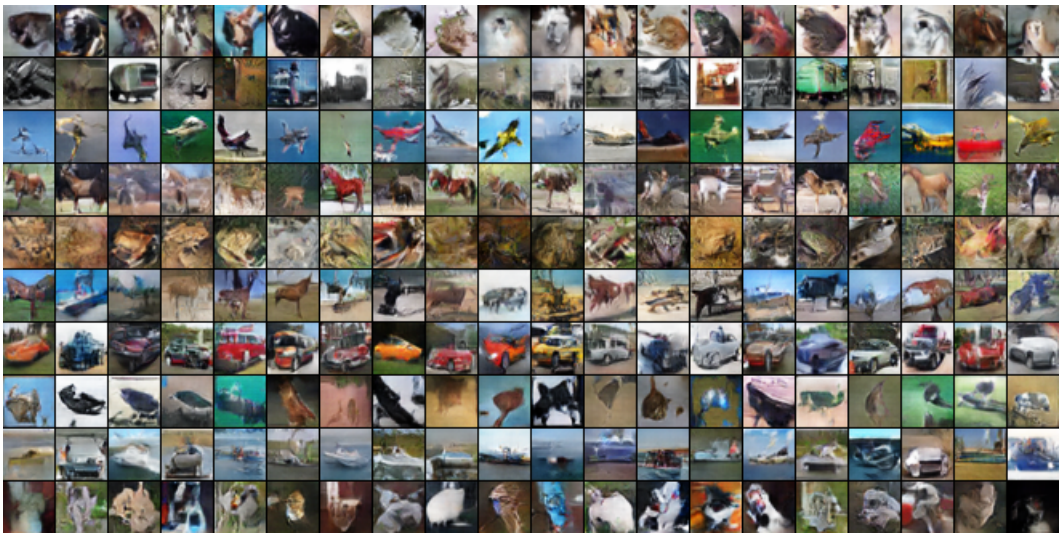


Figure 17: Images generated by **MGAN** trained on CIFAR-10 with **LSGAN** objective.Images in different rows are generated by different generators.

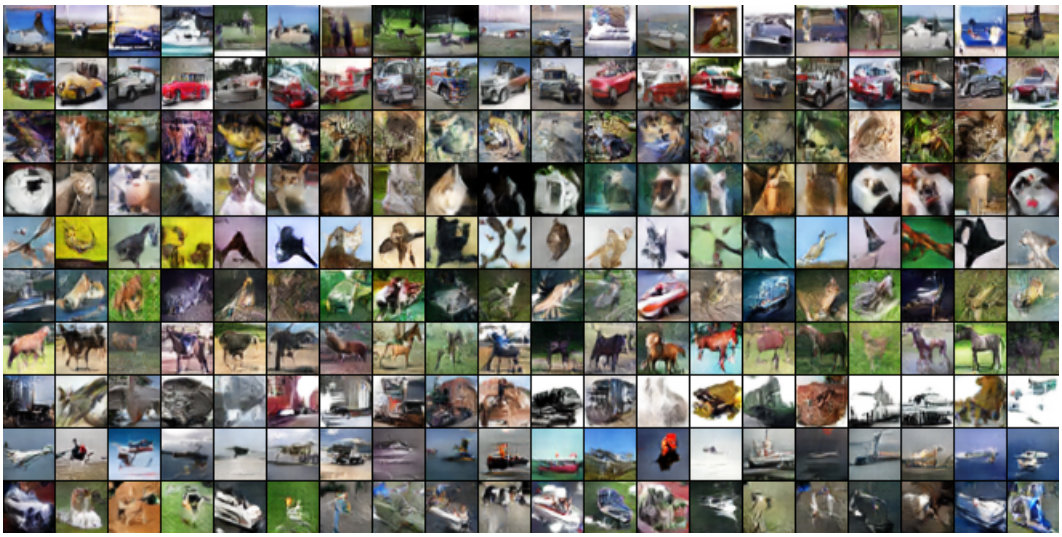


Figure 18: Images generated by **MGAN** trained on CIFAR-10 with **GAN-MM** objective.Images in different rows are generated by different generators.

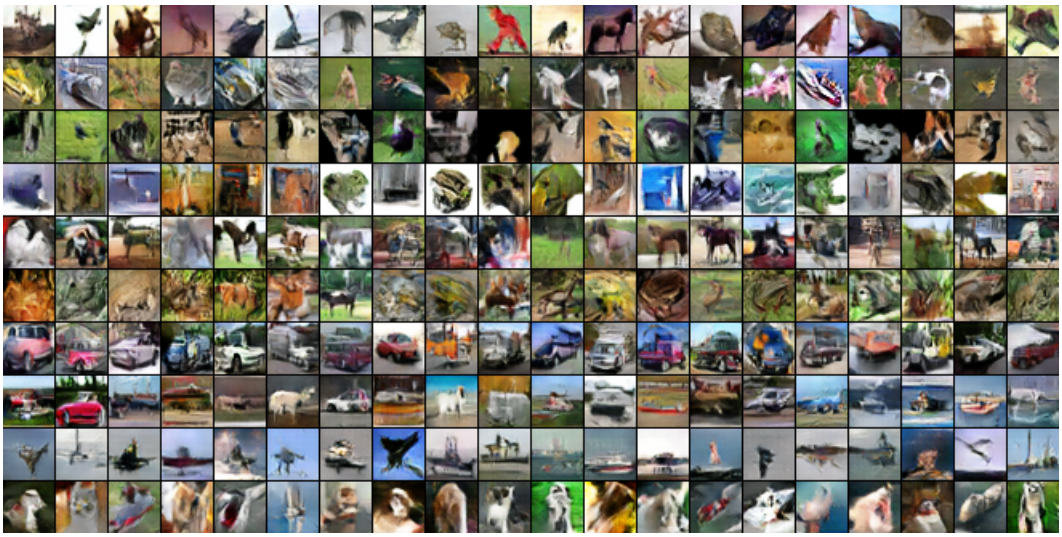


Figure 19: Images generated by **MGAN** trained on CIFAR-10 with **WGAN** objective.Images in different rows are generated by different generators.

Figure 20: Images generated by **DCGAN** trained on CIFAR-10 with **GAN-NS** objective.Images in different rows are generated by different generators.

Figure 21: Images generated by **DCGAN** trained on CIFAR-10 with **LSGAN** objective.Images in different rows are generated by different generators.

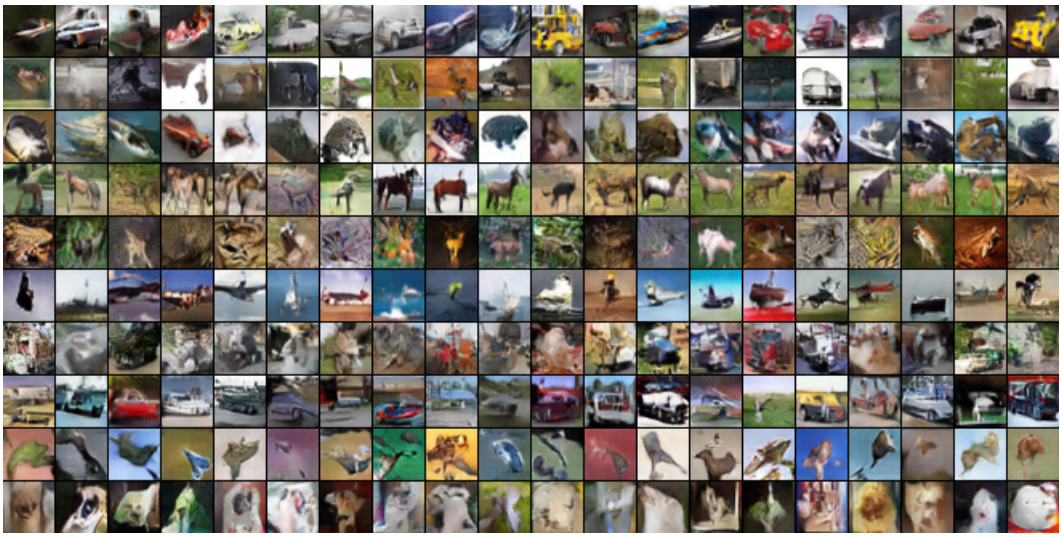


Figure 22: Images generated by **DCGAN** trained on CIFAR-10 with **GAN-MM** objective.Images in different rows are generated by different generators.

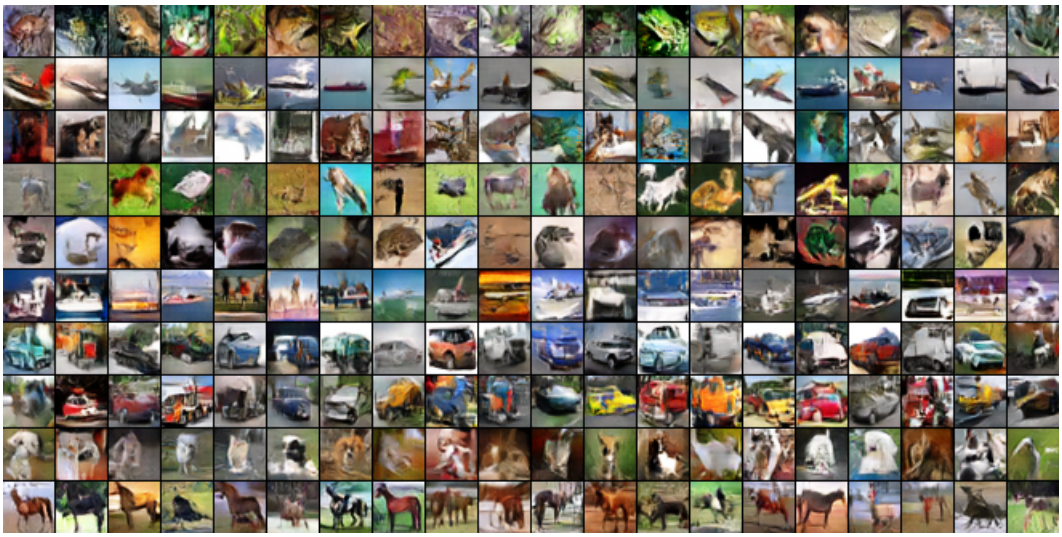


Figure 23: Images generated by **DCGAN** trained on CIFAR-10 with **WGAN** objective.Images in different rows are generated by different generators.

F TOY EXPERIMENT ON CATEGORICAL DISTRIBUTION

Setup. In this toy example, we consider the case where $\mathcal{X}$, Θ_g, and Θ_d are all finite sets, specifically $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$, $\Theta_g = \{\theta_g^1, \cdots, \theta_g^{N_g}\}$, $\Theta_d = \{\theta_d^1, \cdots, \theta_d^{N_d}\}$. The target data distribution is a categorical distribution $\text{Cat}(\lambda_{1:N})$ with $\lambda_i = p_{data}(\mathbf{x}_i)$ as the probability of generating data $\mathbf{x}_i$. The data distribution generated by generator i is another categorical distribution $p_{data}(\mathbf{x}; \theta_g^i) = \text{Cat}(\alpha_{1:N}^i)$. Similarly, the distributions of generator and discriminator can also be parameterized as categorical distributions, i.e. $q_g(\theta_g) = \text{Cat}(\beta_{1:N_g})$ and $q_d(\theta_d) = \text{Cat}(\gamma_{1:N_d})$.

In practice, we set $N = 10, N_g = 20, N_d = 100$. The parameters $\lambda_{1:N}$ of the categorical distribution are obtained by firstly sampling i.i.d $\{\tilde{\lambda_j}\}_{j=1}^N$ from the uniform distribution $\mathcal{U}[0, 1]$ and then normalizing $\lambda_j = \frac{\tilde{\lambda_j}}{\sum_{j=1}^N \tilde{\lambda_j}}$. Other categorical distribution parameters $\alpha_{1:N}^i$, $\beta_{1:N_g}$ and $\gamma_{1:N_d}$ are also initialized in a similar way. For the discriminators, their function values are randomly generated from $\{D(\mathbf{x}_i; \theta_d^j)\}_{i=1,j=1}^{N,N_d} \sim \mathcal{U}[0, 1]$.

Metric. We employ l_1 distance for evaluation which can be directly computed on categorical distributions as follows.

$$\mathcal{D}_{l_1}(p_{data}, p_{model}) = \sum_{\mathbf{x} \in \mathcal{X}} |p_{data}(\mathbf{x}) - p_{model}(\mathbf{x})|. \tag{17}$$

Evaluation. Our Bayesian model (Eqn. 3) and the BGAN (Eqn. 8) are evaluated with three different GAN objectives: min-max, non-saturating and LSGAN. which are listed in Table 1. In each iteration step, the distributions of generator and discriminator are updated in a closed form since the normalization constant of the categorical posterior can be easily calculated via summation.

Result. Figure 24 shows the curves of l_1 distance as a function of number of iterations. For each curve, 20 random trials are averaged. Our model converges to the optimum no matter which objective is employed, showing the robustness of our method. However, the BGAN model is quickly trapped in a bad equilibrium. This might be attributed to the fixed prior of BGAN model which restricts the model space that is explored, while our model maintains a dynamic prior for each iteration which encourages the model to explore towards the optimum.

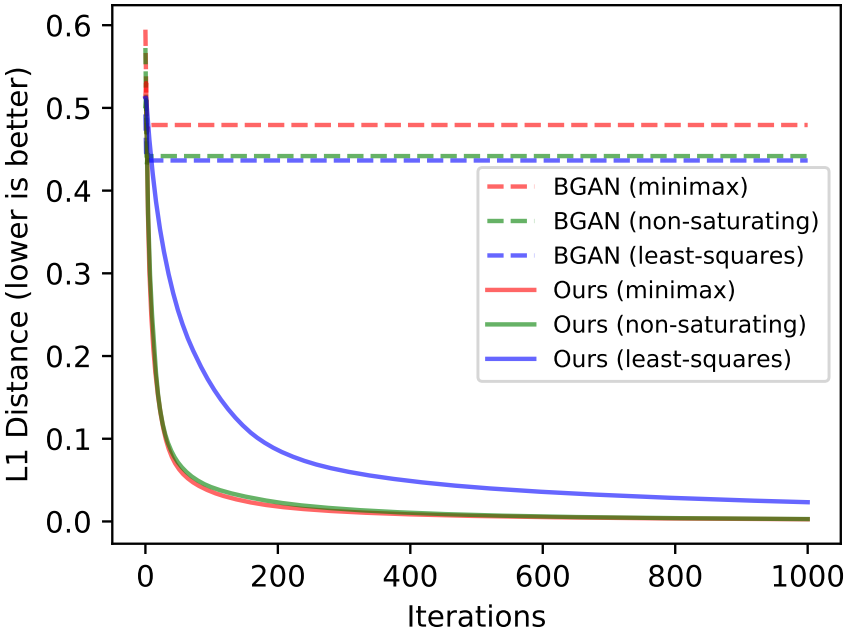


Figure 24: The l_1 distance between generated data distribution and the target versus the number of iterations for our model (in solid line) and BGAN (in dash line). The results with different GAN objectives are indicated by different colors.