Empirical Bayesian GAN

GAN: Definition

The Generative Adversarial Network (GAN) (Goodfellow et al., 2014) provides a novel way for training generative models which seek to learn to generate fake samples with the same statistics as real ones.

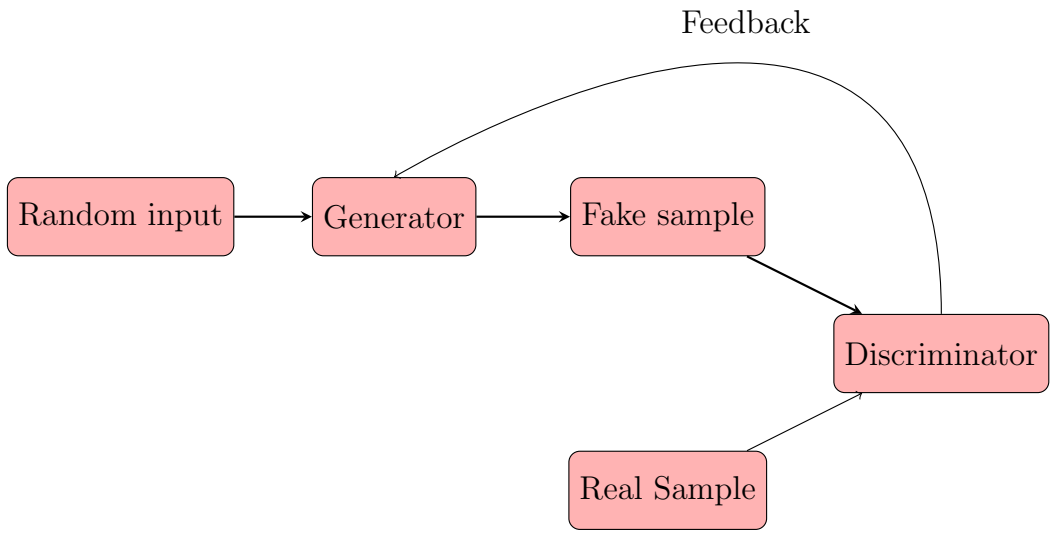


Figure 1: Diagram of GAN, where the generator and discriminator are modeled by deep neural networks.

GAN: mode collapse

GAN is extremely hard to train due to the instability issues such as *mode collapse* (i.e., lack of diversity among fake samples), non-convergence, and vanishing or exploding gradients.

Our Contribution:

- We figure out the reason why GAN suffers from the mode collapse issue: *GAN evaluates fake samples on a one-by-one base through the outputs of the discriminator, lacking a mechanism for enhancing diversity of fake samples.*
- We lay down a new theoretical paradigm for GAN based on randomized decision rules.
- We propose a new training algorithm for GAN, which is immune to mode collapse and whose convergence to the Nash equilibrium is asymptotically guaranteed.

GAN: Exisiting works

The existing methods can be roughly grouped into two categories, metric-based methods and mixture generator methods.

- Metric-based methods: to find a more stable and informative metric to guide the training process, e.g., f-divergence, χ^2-divergence, Wasserstein distance, etc.
- Mixture generator methods: to learn a mixture of generators with a similar motivation as the proposed method, e.g., ensemble GAN, AdaGAN, MAD-GAN, MGAN, Bayesian GAN and ProbGAN.

Pure Strategy Minimax Game: Notations

Let θ_d denote the parameters of the discriminator, let $D_{\theta_d}(x)$ denote its output function, and let $G_{\theta_g}(z)$ denote the generator with parameter θ_g, whose input z is Gaussian or uniform random variable, and whose output distribution is denoted by p_{θ_g}.

Define

$$\begin{aligned} \mathcal{J}_d(\theta_d;\theta_g) &= \mathbb{E}_{x\sim p_{data}}\phi_1(D_{\theta_d}(x)) + \mathbb{E}_{x\sim p_{\theta_g}}\phi_2(D_{\theta_d}(x)), \\ \mathcal{J}_g(\theta_g;\theta_d) &= -\mathbb{E}_{x\sim p_{data}}\phi_1(D_{\theta_d}(x)) + \mathbb{E}_{x\sim p_{\theta_g}}\phi_3(D_{\theta_d}(x)), \end{aligned} \tag{1}$$

where $\phi_1(\cdot)$, $\phi_2(\cdot)$ and $\phi_3(\cdot)$ denote three metrics. For example, $\phi_1(D) = \log(D)$, $\phi_2(D) = \log(1-D)$, and $\phi_3(D) = -\log(1-D)$ or $\log(D)$ in Goodfellow et al. (2014).

Pure Strategy Minimax Game

The general form of the game introduced by GAN is given as follows:

$$(i) \quad \max_{\theta_d} \mathcal{J}_d(\theta_d; \theta_g), \quad (ii) \quad \max_{\theta_g} \mathcal{J}_g(\theta_g; \theta_d). \tag{2}$$

If $\phi_3 = -\phi_2$, the objective of (2) represents a pure strategy minimax game, i.e.,

$$\min_{\theta_g} \max_{\theta_d} \mathcal{J}_d(\theta_g, \theta_d), \tag{3}$$

which is called minimax GAN.

Mixed Strategy Minimax Game

Let $\pi_g(\theta_g)$ denote a distribution of generators. Based on randomized decision theory, we define a mixed strategy minimax game:

$$\min_{\pi_g} \max_{\theta_d} \mathbb{E}_{\pi_g} \mathcal{J}_d(\theta_d; \theta_g), \tag{4}$$

where $\mathcal{J}_d(\theta_d; \theta_g)$ is as defined in (1), and the expectation is taken with respect to $\pi_g(\theta_g)$.

The new game is to iteratively find an optimal discriminator θ_d by maximizing $\mathbb{E}_{\pi_g} \mathcal{J}_d(\theta_d; \theta_g)$ for a given generator distribution π_g and an optimal generator distribution π_g by minimizing $\max_{\theta_d} \mathbb{E}_{\pi_g} \mathcal{J}_d(\theta_d; \theta_g)$ for a given discriminator θ_d.

Mixed Strategy Minimax Game

Let p_{π_g} denote the distribution of the fake samples produced by the generators drawn from π_g, i.e., $p_{\pi_g}(x) = \int p_{\theta_g}(x)\pi_g(\theta_g)d\theta_g$.

Lemma 1

Suppose the discriminator and generator have enough capacity, $\phi_1(D) = \log(D)$, *and* $\phi_2(D) = \log(1 - D)$. *Then* $\min_{\pi_g} \max_{\theta_d} \mathbb{E}_{\pi_g} \mathcal{J}_d(\theta_d; \theta_g) = -\log(4)$. *If* $\tilde{\theta}_d = \arg\max_{\theta_d} \mathbb{E}_{\tilde{\pi}_g} \mathcal{J}_d(\theta_d; \theta_g)$ *for some* $\tilde{\pi}_g$, *then* $(\tilde{\theta}_d, p_{\tilde{\pi}_g})$ *is a Nash equilibrium point if and only if* $\mathbb{E}_{\tilde{\pi}_g} \mathcal{J}_d(\tilde{\theta}_d; \theta_g) = -\log(4)$.

Mixture Strategy Nash Equilibrium

Let $q_g(\theta_g)$ denote the prior distribution of θ_g, and let N denote the training sample size. Define

$$\pi(\theta_g|\theta_d, \mathcal{D}) \propto \exp\{\mathbb{J}_g(\theta_g; \theta_d)\} q_g(\theta_g), \tag{5}$$

where

$$\begin{aligned}\mathbb{J}_g(\theta_g; \theta_d) &= N\mathcal{J}_g(\theta_g; \theta_d) \\ &= N(-\mathbb{E}_{x\sim p_{data}}\phi_1(D_{\theta_d}(x)) + \mathbb{E}_{x\sim p_{\theta_g}}\phi_3(D_{\theta_d}(x)).\end{aligned}$$

For the game (4), we propose to solve θ_d by setting

$$\tilde{\theta}_d = \arg\max_{\theta_d} \int \mathcal{J}_d(\theta_d; \theta_g)\pi(\theta_g|\theta_d, \mathcal{D})d\theta_g, \tag{6}$$

where $\mathcal{J}_d(\theta_d; \theta_g)$ is as defined in (1) and then setting

$$p_{\tilde{\pi}_g} = \int p_{\theta_g}\pi(\theta_g|\tilde{\theta}_d, \mathcal{D})d\theta_g. \tag{7}$$

Mixture Strategy Nash Equilibrium: Theory

Theorem 2

Suppose that the discriminator and generator have enough capacity, $\phi_1(D) = \log(D)$, $\phi_2(D) = \log(1 - D)$, $\phi_3 = -\log(1 - D)$, *and some regularity conditions hold. Then* $(\tilde{\theta}_d, p_{\tilde{\pi}_g})$ *defined in (6)-(7) is a Nash equilibrium point to the game (4) as* $N \to \infty$.

Training Algorithm: Idea

Equation (6) can be solved by solving the equation $\nabla_{\theta_d} \int \mathcal{J}_d(\theta_d; \theta_g)\pi(\theta_g|\theta_d, \mathcal{D})d\theta_g = 0$. When N is sufficiently large, the latter equation can be solved by solving the mean field equation

$$h(\theta_d) = \int H(\theta_d, \theta_g)\pi(\theta_g|\theta_d, \mathcal{D}) = 0, \tag{8}$$

using a stochastic approximation algorithm, where $H(\theta_d, \theta_g)$ denotes an unbiased estimator of $\nabla_{\theta_d}\mathcal{J}_d(\theta_d; \theta_g)$.

The convergence of the solution to the Nash equilibrium can be checked via some plots shown later.

Training Algorithm: Algorithm

By the standard theory of stochastic approximation MCMC, (8) can be solved by iterating between the following two steps:

(i) Simulate $\theta_g^{(t)}$ by a Markov transition kernel which leaves the conditional posterior $\pi(\theta_g|\theta_d^{(t-1)}, \mathcal{D}) \propto \exp\{\mathbb{J}_g(\theta_g; \theta_d^{(t-1)})\} q_g(\theta_g)$ invariant.

(ii) Update the estimate of θ_d by setting $\theta_d^{(t)} = \theta_d^{(t-1)} + w_t H(\theta_d^{(t-1)}, \theta_g^{(t)})$, where w_t denotes the step size used at iteration t.

Stochastic gradient MCMC algorithms, such as SGLD, SGHMC and momentum SGLD, can be used in step (i).

Training Algorithm: Convergence Analysis

Under appropriate conditions, we will show that $|\theta_d^{(t)} - \tilde{\theta}_d| \to 0$ in probability and $\theta_g^{(t)} \sim \pi(\theta_g|\tilde{\theta}_d, \mathcal{D})$ as $t \to \infty$. That is, the proposed algorithm converges to the Nash equilibrium of the mixed strategy minimax game (4).

Lemma 3 (Convergence of discriminator)

If the learning rate ϵ is sufficiently small, then there exist a constant γ, an iteration number t_0 and an optimum $\tilde{\theta}_d = \arg\max_{\theta_d} \int \mathcal{J}_d(\theta_d; \theta_g)\pi(\theta_g|\theta_d, \mathcal{D})d\theta_g$ such that for any $t \geq t_0$,

$$\mathbb{E}\|\theta_d^{(t)} - \tilde{\theta}_d\|^2 \leq \gamma w_t,$$

where t indexes iterations, w_t is the step size converging to 0.

Training Algorithm: Convergence Analysis

Lemma 4 (Ergodicity of generator)

For a smooth test function $\psi(\theta_g)$ *with* $\|\psi(\theta_g)\| \leq C(1 + \|\theta_g\|)$ *for some constant* C*, define*

$$\hat{\psi}_T = \frac{\sum_{t=1}^{T} \epsilon_t \psi(\theta_g^{(t)})}{\sum_{t=1}^{T} \epsilon_t}, \quad \bar{\psi} = \int \psi(\theta_g) \pi(\theta_g | \tilde{\theta}_d, \mathcal{D}) d\theta_g. \tag{9}$$

Under appropriate conditions, we have

$$\mathbb{E}\|\hat{\psi}_T - \bar{\psi}\|^2 \to 0, \quad \text{as } T \to \infty.$$

A Single Gaussian Example

Consider a 2-D Gaussian example, where the real samples are generated in the following procedure:

- Generate the cluster mean: $\mu \sim \mathcal{N}(0, I_2)$, where I_2 denotes a 2-dimensional identity matrix.
- Generate a mapping matrix: $M \sim \mathcal{N}(0, I_{2\times 2})$, that is, M is a 2×2-matrix with each element being independently drawn from $\mathcal{N}(0, 1)$.
- Generate 10000 observations: $x_i \sim (\mathcal{N}(0, I_2) + \mu) \times M^T$, for $i = 1, 2, \ldots, 10000$.

Both the discriminator and generators used for this example are fully connected neural networks with ReLU activation. The discriminator has a structure of $2 - 1000 - 1$, and the generator has a structure of $10 - 1000 - 2$.

A Single Gaussian Example: Convergence Path

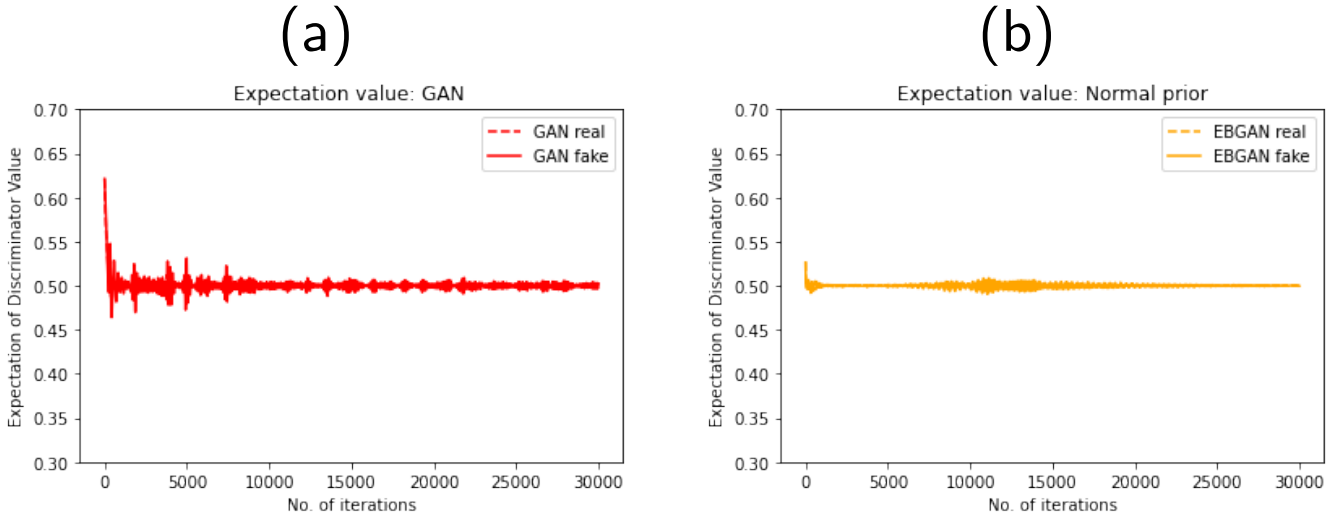
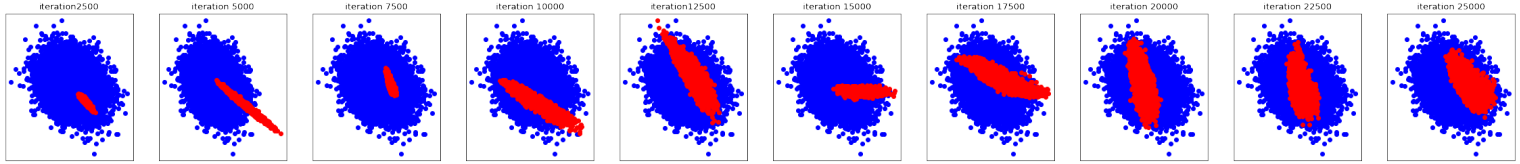


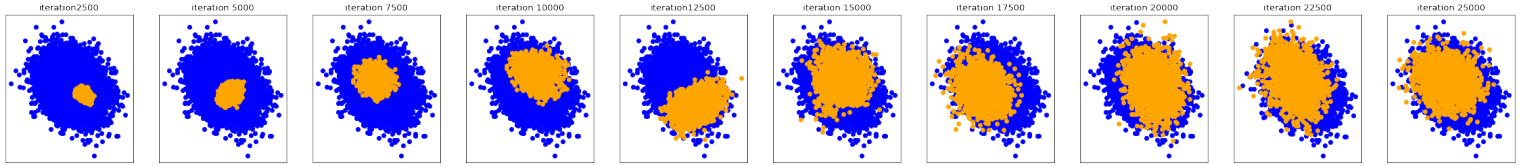
Figure 2: Empirical means of $D_{\theta_d^{(t)}}(x_i)$ and $D_{\theta_d^{(t)}}(\tilde{x}_i)$ produced by (a) GAN and (b) EBGAN with a Gaussian prior along with iterations.

A Single Gaussian Example: Coverage Plot

(a) Evolution of coverage by GAN



(b) Evolution of coverage by one generator from EBGAN



(c) Evolution of coverage by integration of all 10 generators from EBGAN

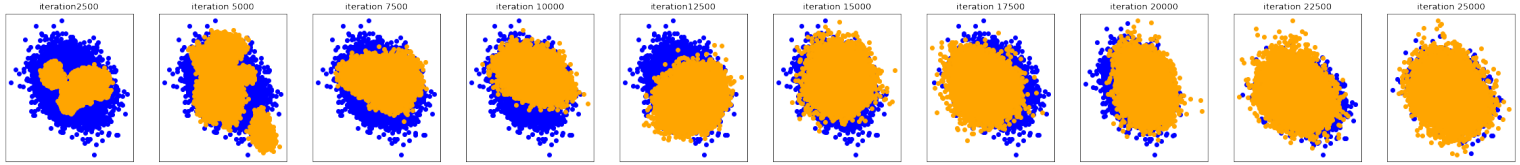


Figure 3: Coverage plots produced by (a) GAN, (b) a single generator of EBGAN, and (c) all 10 generators of EBGAN, where the generators for the plots from left to right are collected at iterations 2500, 5000, 7500, ..., 25000, respectively.

Mixture Gaussian Example

The dataset was generated from a 10-component mixture Gaussian distribution in the following procedure:

(i) Generate 10 cluster means: $\mu^{(j)} \sim 5\mathcal{N}(0, I_2)$, $j = 1, 2, \ldots, 10$.

(ii) generate 10 mapping matrices: $M^{(j)} \sim 5\mathcal{N}(0, I_{100\times 2})$, $j = 1, 2, \ldots, 10$.

(iii) generate 1000 observations of $x^{(j)}$ for each $(\mu^{(j)}, M^{(j)})$: $x_i^{(j)} \sim (\mathcal{N}(0, I_2) * 0.5 + \mu^{(j)}) \times (M^{(j)})^T$, for $j = 1, 2, \ldots, 10$, $i = 1, 2, \ldots, 1000$.

EBGAN was run with the prior $q_g = \mathcal{N}(0, I)$, $k_g = 10$, and $\phi_3(D) = -\log(1 - D)$ and $\log(D)$. The discriminator has a structure of $100 - 1000 - 1$ and the generator has a structure of $10 - 1000 - 100$, which are the same as those used in Saatci and Wilson (2017).

Mixture Gaussian Example: Convergence Path

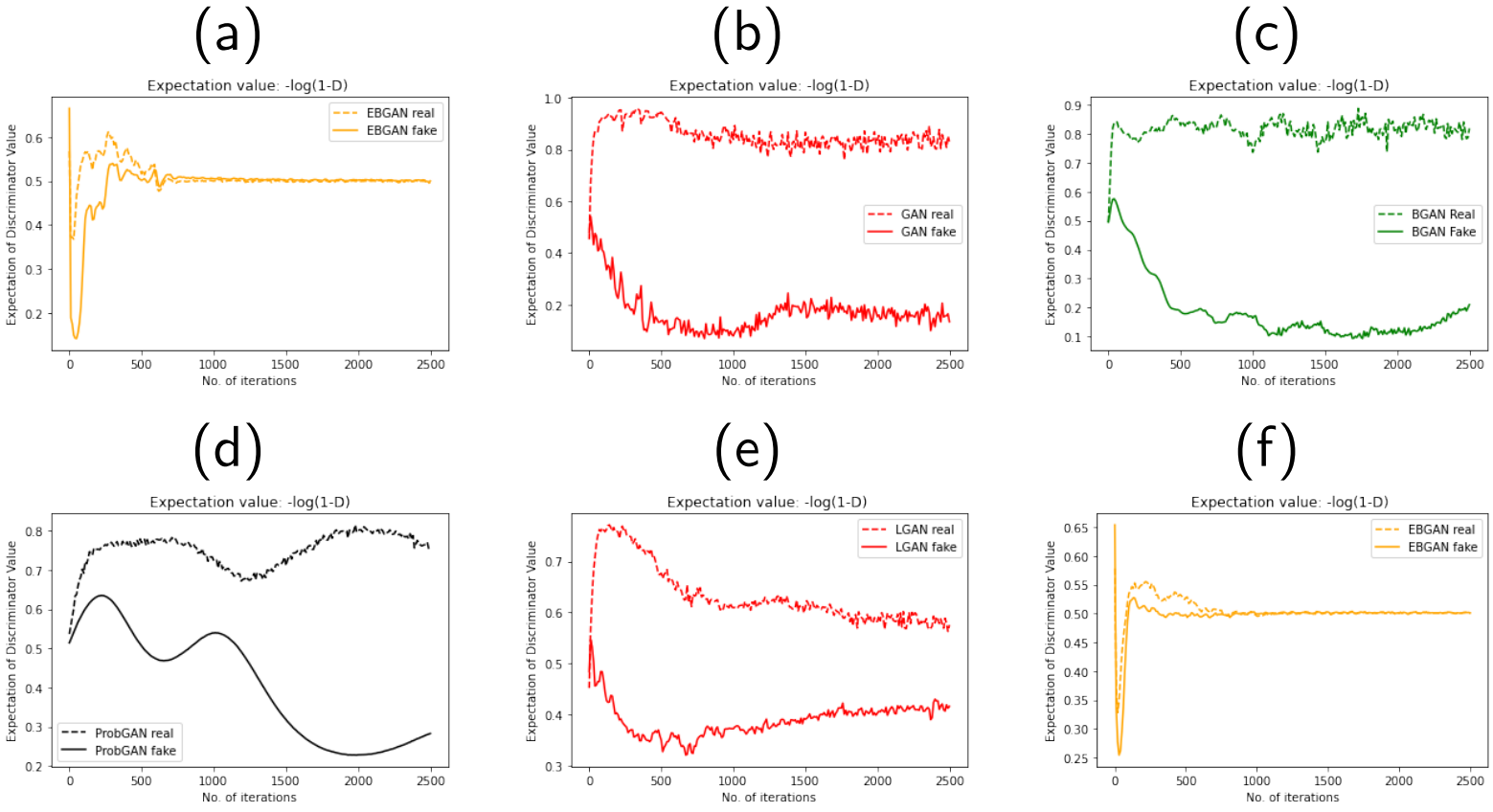


Figure 4: Nash equilibrium convergence plots of the empirical means of $D_{\theta_d^{(t)}}(x_i)$ and $D_{\theta_d^{(t)}}(\tilde{x}_i)$ produced by different methods: (a) EBGAN, (b) minimax GAN, (c) Bayesian GAN, (d) ProbGAN, (e) Lipschitz-GAN, (f) EBGAN with a Lipschitz penalty.

Mixture Gaussian Example: Coverage plot

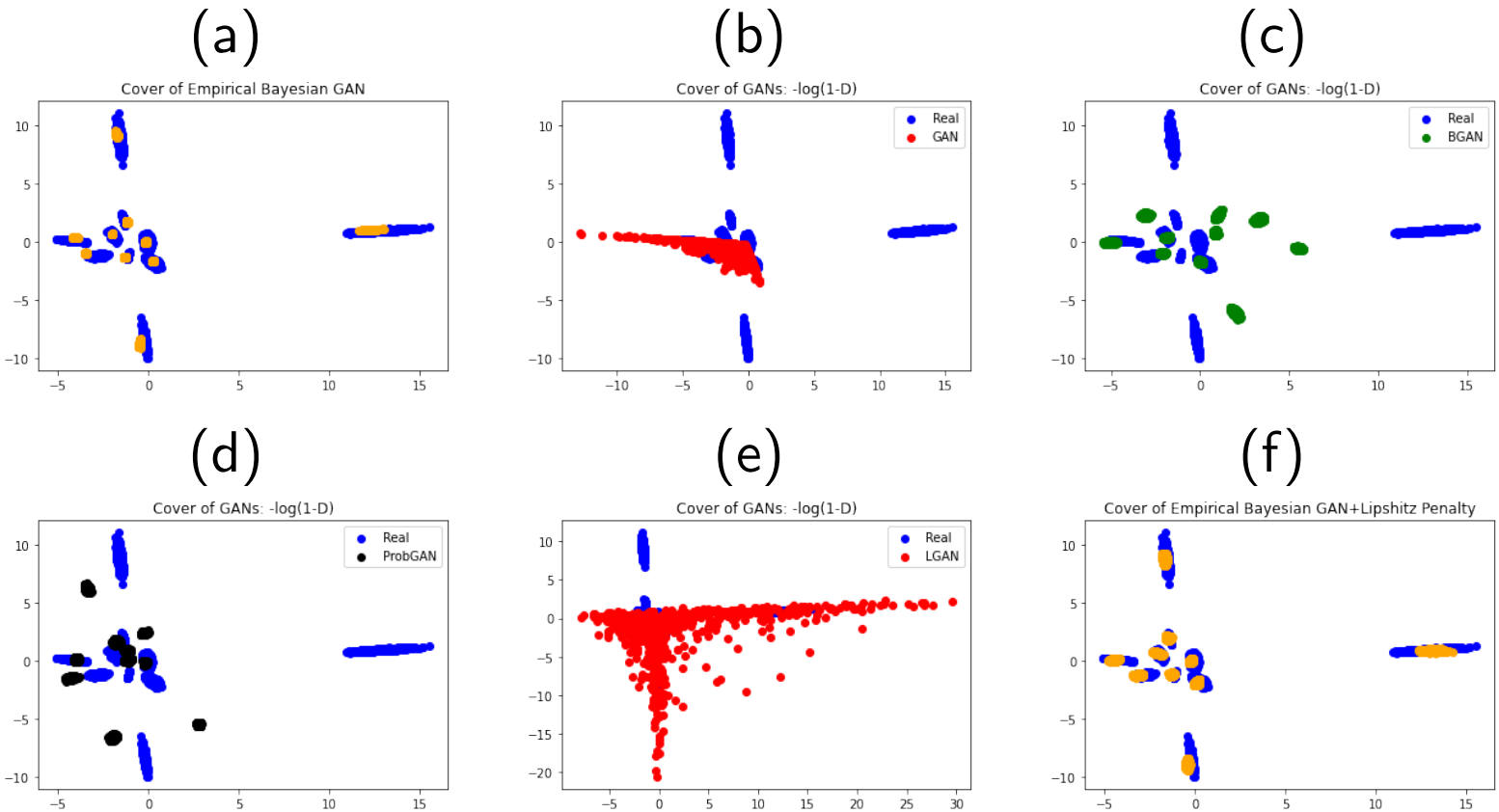


Figure 5: Component recovery plots produced by different methods with $\phi_3(D) = -\log(1 - D)$: (a) EBGAN, (b) minimax GAN, (c) Bayesian GAN, (d) ProbGAN, (e) Lipschitz GAN, and (f) EBGAN with a Lipschitz penalty.

Image Generation: Fashion-MNIST

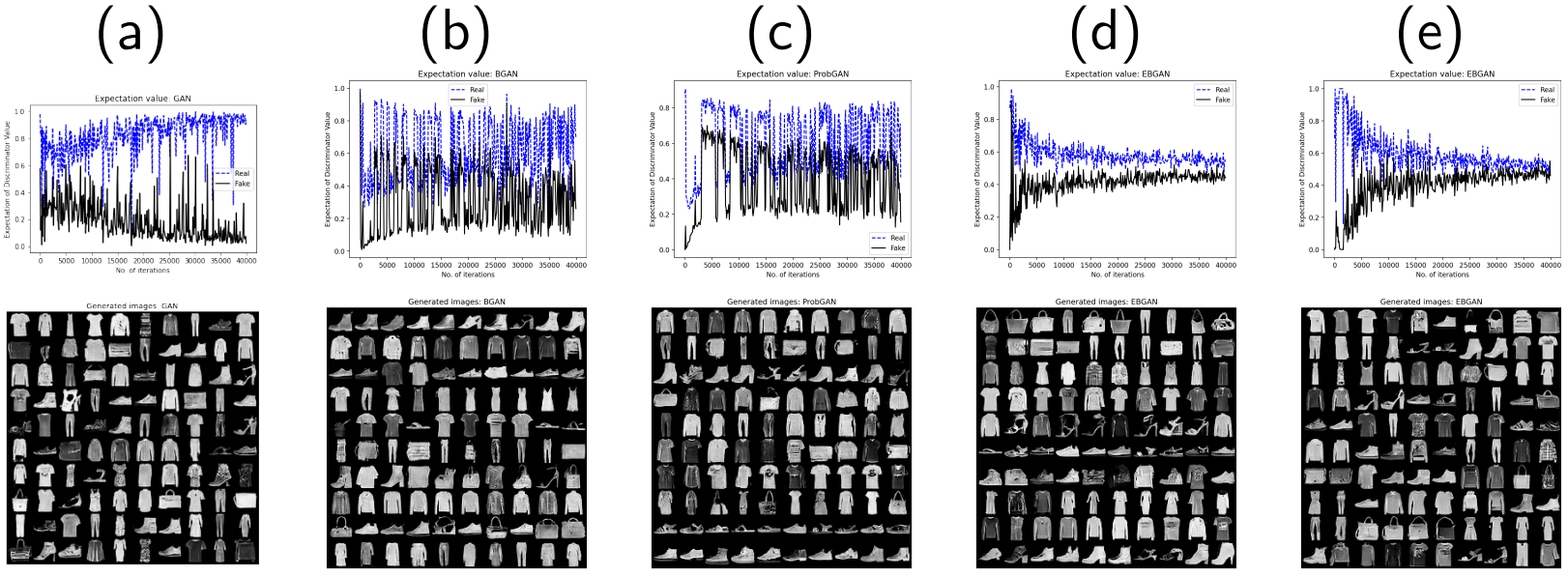


Figure 6: Convergence plots and images produced by (a) GAN, (b) Bayesian GAN, (c) ProbGAN, (d) EBGAN with a KL-divergence prior, and (e) EBGAN with a Gaussian prior.

Nonparametric Clustering

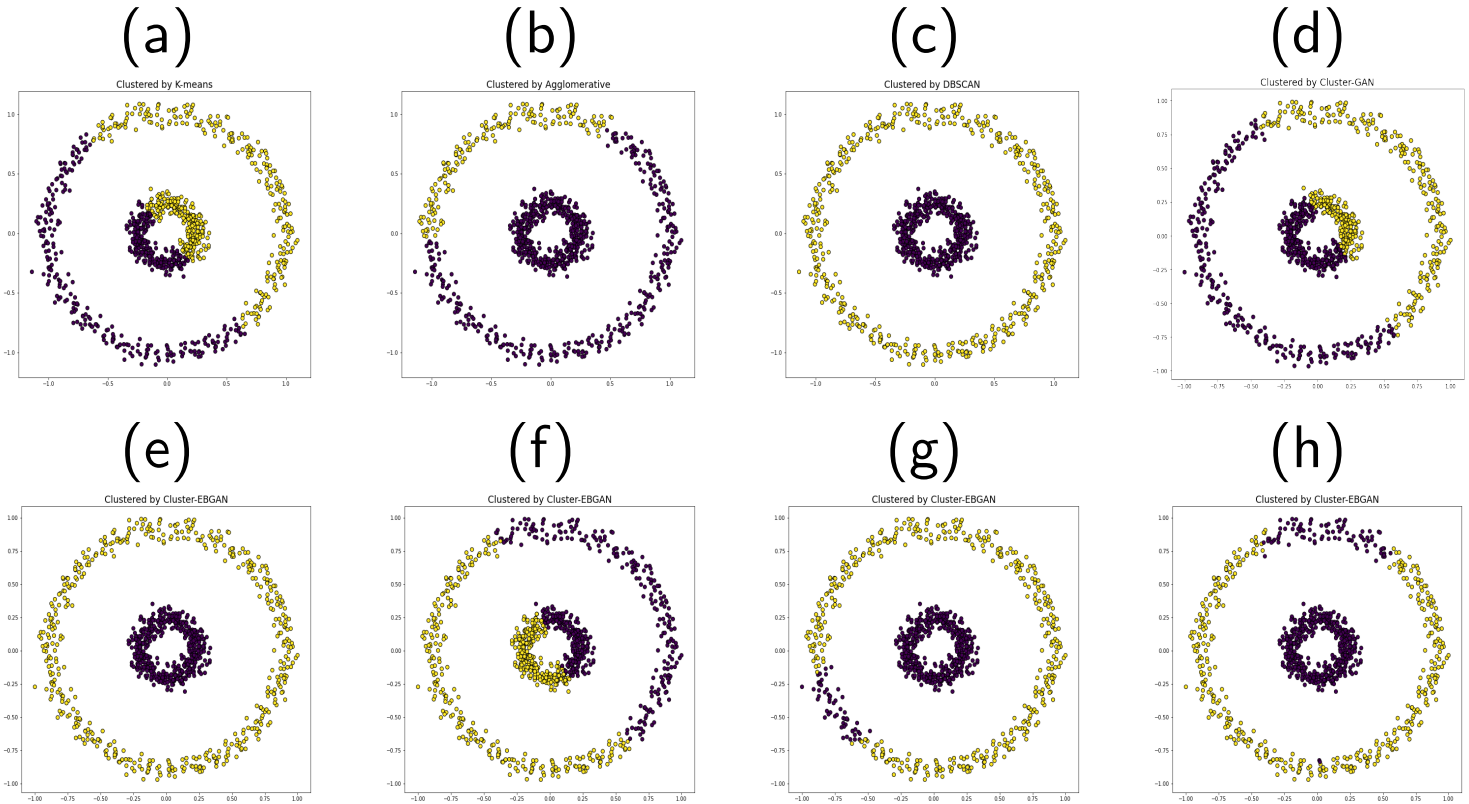


Figure 7: (a) K-means clustering, (b) Agglomerative clustering, (c) DBSCAN, (d) Cluster-GAN, (e)-(h) Cluster-EBGAN in different runs.

Conclusion

- We have figured out the reason why GAN suffers from the mode collapse issue, proposed a new theoretical framework for addressing this issue, and proposed a new method for training GANs.
- The proposed empirical Bayes-like method can be extended in various ways, e.g., more efficient SGMCMC algorithms for simulating generators and more efficient optimization algorithms for training the discriminator. Also, sparse generators and discriminator can be learned for the method with appropriate penalty functions.