Learning Implicit Generative Models by Teaching Explicit Ones

Chao Du*K 1  Kun Xu*K 1  Chongxuan Li 1  Jun Zhu 1  Bo Zhang 1

Abstract
Implicit generative models are difficult to train as no explicit density functions are defined. Generative adversarial nets (GANs) present a minimax framework to train such models, which however can suffer from mode collapse due to the nature of the JS-divergence. This paper presents a learning by teaching (LBT) approach to learning implicit models, which intrinsically avoids the mode collapse problem by optimizing a KL-divergence rather than the JS-divergence in GANs. In LBT, an auxiliary explicit model is introduced to fit the distribution defined by the implicit model while the later one teaches the explicit model to match the data distribution. LBT is formulated as a bilevel optimization problem, whose optimal generator matches the true data distribution. LBT can be naturally integrated with GANs to derive a hybrid LBT-GAN that enjoys complimentary benefits. Finally, we present a stochastic gradient ascent algorithm with unrolling to solve the challenging learning problems. Experimental results demonstrate the effectiveness of our method.

1. Introduction
Deep generative models (DGMs) (Kingma & Welling, 2013; Goodfellow et al., 2014; Oord et al., 2016) are powerful tools to capture the distributions over complicated manifolds (e.g., natural images), especially the recent developments of implicit statistical models (Radford et al., 2015; Arjovsky et al., 2017; Karras et al., 2017; Mohamed & Lakshminarayanan, 2016), also called implicit probability distributions. Implicit models are flexible by adopting a sampling procedure rather than a tractable density. However, they are difficult to learn partly because maximum likelihood estimation (MLE) is not directly applicable.

Generative adversarial networks (GANs) (Goodfellow et al., 2014) address this difficulty by adopting a minimax game, where a discriminator D is introduced to distinguish whether a sample is real (i.e., from the data distribution) or fake (i.e., from a generator G), while G tries to fool D via generating realistic samples. Although GANs can produce high quality samples, it suffers from lacking sample diversity, also known as the mode collapse problem (Goodfellow, 2016).

The main reason for mode collapse arises from the objective function optimized by GANs (Nguyen et al., 2017), which is shown to minimize the JS-divergence between the data distribution \( p_D \) and the generator distribution \( p_G \) (Goodfellow et al., 2014). As shown in (Huszár, 2015; Theis et al., 2015) and illustrated in Fig. 1, JS-divergence can be tolerant to mode collapse whereas the \( KL(p_D || p_G) \) achieves its optimum iff \( p_D = p_G \). Nowozin et al. (2016) enable us to train implicit models via KL-divergence using importance sampling, i.e., estimating the KL-divergence using generated samples. However, it may also fail in practice (Metz et al., 2016) as the KL-divergence will be under-estimated if the generated samples do not capture all modes in training data.

To address the above issues, we propose learning by teaching (LBT), a novel framework to learn implicit models. LBT optimizes the KL-divergence, which is more resistant to mode collapse than the JS-divergence due to the zero-avoiding properties (Nasrabadi, 2007). In LBT, we learn an implicit generator \( G \) by teaching a likelihood estimator \( E \) to match the data distribution. The training scheme consists of two parts:

(a) The estimator \( E \) is trained to maximize the log-likelihood of the samples of the generator \( G \);
(b) The generator \( G \)'s goal is to improve the performance of the trained estimator in terms of the log-likelihood of real data samples.

Though in both LBT and GAN, an auxiliary network is introduced to help the training of the generator, the role of \( E \) in LBT is significantly different from that of \( D \) in GAN, and they are complimentary to each other. The estimator \( E \) in LBT penalize \( G \) for missing modes in training data, whereas the discriminator \( D \) in GAN penalize \( G \) for generating unrealistic samples. In LBT, \( E \) always tracks \( p_G \) and once \( p_G \) misses some modes, the estimator \( E \) will also miss them, resulting a poor likelihood of real data samples, which penalize \( G \) heavily. In such a manner, the estimator in LBT directs the generated samples to overspread the support of data distribution. In contrast, the goal of \( D \) in
the vanilla GAN is to distinguish whether a sample is real or fake. Therefore, during the competing with $D$, $G$ will be penalized much more heavily for generating unrealistic samples than missing modes. Based on this insight, we further conjoin the complimentary advantages to develop LBT-GAN, which augments LBT with a discriminator network. In LBT-GAN, $E$ helps $G$ to overspread the data distribution and $D$ helps $G$ to generate realistic samples.

Formally, LBT (and LBT-GAN) is formulated as a bilevel optimization (Colson et al., 2007) problem, where an upper level optimization problem (i.e., part (b)) is dependent on the optimal solution of a lower level problem (i.e., part (a)). The gradients of the upper problem w.r.t. the parameters of $G$ are intractable since the optimal solution of $E$ cannot be analytically expressed by $G$’s parameters. We propose to use the unrolling technique (Metz et al., 2016) to efficiently approximate the gradients and verify its correctness theoretically. Under nonparametric conditions, the optimum of LBT (and LBT-GAN) is achieved when both the generator and the estimator converge to the data distribution. Besides, we further analyze that an estimator with insufficient capability can still help $G$ to resist to mode collapse in LBT-GAN. Experimental results on both synthetic and real datasets demonstrate the effectiveness of LBT and LBT-GAN.

2. Background

Consider an implicit generative model $G(\cdot; \theta)$ parameterized by $\theta$ that maps a simple random variable $z \in \mathbb{R}^H$ to a sample $x$ in the data space $\mathbb{R}^L$, i.e., $x = G(z; \theta)$, where $H$ and $L$ are the dimensions of the random variables and the data space, respectively. Typically, $z$ is drawn from a standard Gaussian distribution $p_Z$ and $G$ is a feed-forward neural network. The sampling procedure defines a distribution $p_G(x; \theta)$ over the data space. The goal of the generator $G$ is to approximate the data distribution $p_D(x)$, i.e., to produce samples of high quality and diversity.

Since the generator distribution is implicit, it is infeasible to adopt MLE directly to train the generator. To address this problem, GAN (Goodfellow et al., 2014) adopts a minimax game, where a discriminator $D(\cdot; \psi)$ parameterized by $\psi$ is introduced to distinguish generated samples from true data samples, while the generator $G$ tries to fool $D$ via generating realistic samples. The parameters of $G$ and $D$ are learned by solving a minimax game:

$$\min_G \max_D \mathbb{E}_{p_D}[\log D(x)] + \mathbb{E}_{p_G}[\log(1 - D(G(z))]. \quad (1)$$

Goodfellow et al. (2014) show that the discriminator achieves its optimum when $D(x) = p_D(x) / p_D(x) + p_G(x)$, and the minimax problem is equivalent to minimizing the JS-divergence between $p_D(x)$ and $p_G(x)$, whose optimal point is $p_G = p_D$, under the assumption that $G$ and $D$ have infinite capacity. However, GANs can suffer from the mode collapse problem for both theoretical reasons (Nguyen et al., 2017; Huszár, 2015) and practical reasons (Metz et al., 2016; Srivastava et al., 2017; Arjovsky et al., 2017).

From the theoretical perspective, previous work has investigated the mode collapse nature of JS-divergence (Nguyen et al., 2017). By optimizing the JS-divergence, the generative model tends to cover certain modes, rather than overspreading the data distribution (Theis et al., 2015), thus leading to mode collapse in GANs. Fig. 1 (left) presents a simple example, where the local optima with mode collapse can still be found by optimizing the JS-divergence, even if $p_G$ is flexible enough. In contrast, the KL-divergence can overcome this problem because of the zero-avoiding property (Nasrabadi, 2007), and Fig. 1 (right) shows that $KL(p_D||p_G)$ achieves its optima iff $p_G = p_D$.

There are previous attempts on training implicit models by optimizing other divergence, including the KL-divergence (Nowozin et al., 2016; Nguyen et al., 2017). For instance, D2GAN (Nguyen et al., 2017) uses an auxiliary discriminator to diversify the generator distribution, which introduces the KL-divergence into the objective function. However, it practically fails as the discriminators in D2GAN are fixed during the update of the generator, which makes that the gradient of the KL-divergence w.r.t. the generator cannot be propagated through the discriminator and breaks the zero-avoiding property of the KL-divergence. Nowozin et al. (2016) propose to estimate the KL-divergence using importance sampling, i.e., $KL(p_D||p_G) = \mathbb{E}_{p_G}[p_G \log p_G / p_G]$. However, the estimate will be of large variance if the generator fails to capture all modes in data as it is difficult to draw a sample in the missed modes in $p_D$ and the KL-divergence tends to be under-estimated. Therefore, once the generator distribution collapsed, the estimated KL-divergence cannot penalize the generator for missing modes and encourage the generator to capture all modes in training data.

3. Method

To address the mode collapse issue, we present a novel framework learning by teaching (LBT), which enables us
to learn implicit models by optimizing the KL-divergence between \( p_D(x) \) and \( p_G(x) \).

### 3.1. Learning by Teaching (LBT)

We introduce an auxiliary density estimator \( p_E(x; \phi) \) parameterized by \( \phi \) to learn the distribution defined by the implicit generator \( G(z; \theta) \), which provides a surrogate density for \( G \) to estimate the KL-divergence between \( p_D \) and \( p_G \). Specifically, in our method, the estimator’s goal is to learn the generator distribution via MLE, and the generator’s goal is to maximize the taught estimator’s likelihood evaluated on the training data. Formally, LBT is defined as a bilevel optimization problem (Colson et al., 2007):

\[
\begin{align*}
\max_{\theta} \quad & \mathbb{E}_{x \sim p_D}(x) \log p_E(x; \phi^*(\theta)), \\
\text{s.t.} \quad & \phi^*(\theta) = \arg \max_{\phi} \mathbb{E}_{z \sim p_Z} \log p_G(G(z; \theta); \phi),
\end{align*}
\]

where \( \phi^*(\theta) \) indicates that the optimal \( \phi^* \) of the lower level problem depends on \( \theta \), which is the variable to be optimized in the upper level problem. For clarity, we denote the objectives of the upper and lower level problems as

\[
\begin{align*}
f_G(\phi^*(\theta)) &:= \mathbb{E}_{z \sim p_Z}(\log p_G(G(z; \theta); \phi)), \\
f_E(\theta, \phi) &:= \mathbb{E}_{z \sim p_Z} \log p_E(G(z; \theta); \phi).
\end{align*}
\]

We now provide the following theorem to demonstrate the correctness of LBT under the assumption that \( G \) and \( E \) have sufficient capacity, which has been justified by recent advances of deep generative models (Goodfellow et al., 2014; Kingma & Welling, 2013; Oord et al., 2016).

**Theorem 1 (Proof in Appendix).** Solving problem (2) is equivalent to minimizing the KL-divergence between the data distribution and the generator distribution, and it’s optima is achieved when

\[
p_G = p_E = p_D. \quad (4)
\]

Theorem 1 shows that the global optimum of LBT is achieved at \( p_G = p_E = p_D \) if the estimator has enough capacity. Below, we give a further analysis to provide a weaker conclusion for LBT under a mild assumption that the estimator has only limited capacity.

**Exponential family:** Consider the case where the estimator is formulated in the form of an exponential family, i.e., \( p_E(x) = h(x)e^{\eta T(x) - A(\eta)} \), where \( T(x) \) denotes the sufficient statistics and \( \eta \) are the natural parameters. In this case, for a certain distribution \( q \), the \( KL(q\|p_E) \) achieves optimal iff \( p_E \) captures the sufficient statistics of \( q \), i.e., \( \mathbb{E}_{p_E} T(x) = \mathbb{E}_q T(x) \) (Nasrabadi, 2007). Hence, for a given generator distribution \( p_G \), the estimator \( E \) achieves optimal when \( \mathbb{E}_{\phi E} p(x) = \mathbb{E}_{p_G} T(x) \). To make the taught estimator \( E^* \) achieve optimal likelihood on data samples (or equivalently, optimal \( KL(p_D\|p_E^*) \)), \( G \) should teach \( E \) to capture the sufficient statistics of the data distribution, i.e., \( \mathbb{E}_{p_D} T(x) = \mathbb{E}_{p_D} T(x) \). Therefore, the estimator can still successfully regularize \( p_G \) to match \( p_D \) in terms of sufficient statistics,

\[
\mathbb{E}_{p_G} T(x) = \mathbb{E}_{p_D} T(x), \quad (5)
\]

which is a weaker conclusion with fewer assumptions compared to Eqn. (4). We provide an example to verify the above analysis in Sec. 3.2 and demonstrate the effectiveness of an estimator beyond the exponential family on real applications in Sec. 5.2.

### 3.2. Augmenting LBT with a Discriminator

KL-divergence is known to be zero-avoiding (Nasrabadi, 2007) in that it encourages the model distribution to cover the data distribution. However, in practice it may also result in low quality of generated samples (Tolstikhin et al., 2017). This property makes LBT complementary to GAN which tends to generate samples of high quality but lack sample diversity (Goodfellow, 2016). To combine the best of both worlds, we further propose to augment LBT with a discriminator as in GANs, and call the hybrid model LBT-GAN. Formally, LBT-GAN solves the bilevel problem:

\[
\begin{align*}
\max_{\theta} \quad & f_G(\phi^*(\theta)) + \lambda_G \mathbb{E}_{z \sim p_Z} \log D(G(z; \theta); \psi^*), \\
\text{s.t.} \quad & \phi^*(\theta) = \arg \max_{\phi} f_E(\theta, \phi), \\
& \psi^* = \arg \max_{\phi} \left( \mathbb{E}_{z \sim p_Z} \log D(z; \psi) + \mathbb{E}_{z \sim p_D} \log (1 - D(G(z; \theta); \psi)) \right),
\end{align*}
\]

where \( \psi \) denotes the parameters of the discriminator \( D \) and \( \lambda_G \) balances the weight between two losses. Under the assumption that \( G \) and \( D \) have sufficient capacity, Goodfellow et al. (2014) show that the optimum of GAN’s minimax framework is achieved at \( p_G = p_D \), which is consistent with the conditions in Eqn. (4) and Eqn. (5) for LBT. Therefore, it is straightforward that LBT-GAN has the same global optimal solution as GAN, i.e., \( p_G = p_D \).

We show that LBT-GAN has advantages compared to GAN even when the estimator has only limited capacity. As mentioned above, GAN can suffer from mode collapse problem since gradient-based optimization methods could fall into a mode-collapsed local optimum of the JS-divergence. However these mode-collapsed local optima are less likely to satisfy the condition in Eqn. (5). The estimator can provide training signal to the generator and help it to escape the local optima that violates Eqn. (5), and therefore make LBT-GAN more resistant to the mode collapse problem.

To give an example to verify our argument, we consider the settings of the toy example in Fig. 1. In LBT-GAN, we suppose that the estimator \( E \) is a single Gaussian \( p_E(x) = N(\phi, 1) \) with a learnable mean \( \phi \), which can only capture the mean of a distribution. In this case, the condition
Figure 2. An illustration of LBT-GAN where the estimator has insufficient capacity. We consider the same data distribution $p_D(x)$ and model distribution $p_G(x)$ as described in Fig. 1. We train both GAN and LBT-GAN to learn the model $p_G(x)$. For LBT-GAN, we use an estimator with insufficient capacity, i.e., $p_E(x) = N(\phi, 1)$. The training processes of GAN (Left) and LBT-GAN (Middle) and their learned distributions (Right) are shown. We observe that while GAN learns a mode-collapsed model, LBT-GAN can escape the local optimum and capture the data distribution in this case.

Eqn. (5) requires that $p_G$ has the same mean as $p_D$. Therefore, if $p_G$ is around the bad local optimum of GAN (where the gradients of the GAN part will be nearly zero), the gradients of the LBT part will encourage the generator to escape the local optima with non-zero mean. A clear demonstration is shown in Fig. 2, where we identically initialize the means of $G$ around $-3$ in both LBT-GAN and GAN. GAN converges to a local optimum of JS- divergence, whereas LBT-GAN can converge to the global optimum quickly as the estimator regularizes the generator to a distribution with zero mean. The further experimental results on real datasets are illustrated in Sec. 5.2.

3.3. Stochastic Gradient Ascent via Unrolling

The bilevel problem is generally challenging to solve. Here, we present a stochastic gradient ascent algorithm (i.e., Algorithm 1) by using an unrolling technique (Metz et al., 2016) to derive the gradient. For clarity, we focus on learning LBT and the methods can be directly applied to learn LBT-GAN. Specifically, to perform gradient ascent, we calculate the gradient of $f_G$ with respect to $\theta$ as follows:

$$\frac{\partial f_G(\phi^*(\theta))}{\partial \theta} = \frac{\partial f_G(\phi^*(\theta))}{\partial \phi^*(\theta)} \frac{\partial \phi^*(\theta)}{\partial \theta} = \frac{\partial f_G(\phi^*(\theta))}{\partial \phi^*(\theta)} \int \frac{\partial G(z; \theta)}{\partial \theta} p_Z dz,$$

where both $\frac{\partial f_G(\phi^*(\theta))}{\partial \phi^*(\theta)}$ and $\frac{\partial G(z; \theta)}{\partial \theta}$ are easy to calculate. However, the term $\frac{\partial f_G(\phi^*(\theta))}{\partial \phi^*(\theta)}$ is intractable since $\phi^*(\theta)$ cannot be expressed as an analytic function of the generated samples $G(z; \theta)$. We instead consider a local optimum $\phi^*$ of the density estimator parameters, which can be expressed as the fixed point of an iterative optimization procedure,

$$\phi^0 = \phi,$$

$$\phi^{k+1} = \phi^k + \eta \cdot \frac{\partial f_E(\theta, \phi)}{\partial \phi} \bigg|_{\phi^k},$$

$$\phi^* = \lim_{k \to \infty} \phi^k,$$

where $\eta$ is the learning rate. Since the samples used to evaluate the likelihood $f_E(\theta, \phi)$ are generated by $G(\cdot; \theta)$, each step of the optimization procedure is dependent on $\theta$. We thus write $\phi^k(\theta, \phi^0)$ to clarify that $\phi^k$ is a function of $\theta$ and the initial value $\phi^0$. In the following, we rewrite $G(z; \theta)$ as $x_z$ and $\frac{\partial f_E(\theta, \phi)}{\partial \phi}$ as $\nabla \phi$ for simplicity. Since $\nabla \phi$ is differentiable w.r.t. $x_z$, we can optimize for $\theta$ for most density estimators such as VAEs, $\phi^k(\theta, \phi)$ is also differentiable w.r.t. $x_z$. By unrolling for $K$ steps, namely, using $\phi^K(\theta, \phi^0)$ to approximate $\phi^*(\theta)$ in the objective $I_F(\phi^*(\theta))$, we optimize a surrogate objective formulated as $f_G(\phi^k(\theta, \phi^0))$ for the generator. Thus, the term $\frac{\partial \phi^*(\theta)}{\partial x_z}$ is approximated as $\frac{\partial \phi^*(\theta)}{\partial x_z} \approx \frac{\partial \phi^k(\theta, \phi^0)}{\partial x_z}$, which is known as the unrolling technique (Metz et al., 2016).

Finally, the generator and the likelihood estimator can be updated using the following process,

$$\theta \leftarrow \theta + \eta_\theta \frac{\partial f_G(\phi^K(\theta, \phi))}{\partial \theta},$$

$$\phi \leftarrow \phi + \eta_\phi \frac{\partial f_E(\theta, \phi)}{\partial \phi},$$

where $\eta_\theta$ and $\eta_\phi$ are the learning rates for the generator and the estimator, respectively. We perform several updates of $\phi$ per update of $\theta$ to keep the estimator good. Note that for other gradient-based optimization methods such as Adam (Kingma & Ba, 2014), the unrolling procedure is similar (Metz et al., 2016). In our experiments, only a few steps of unrolling, e.g., 5 steps, are sufficient for the training.

The whole training algorithm is described in Algorithm 1.

3.3.1. CONNECTION TO INFLUENCE FUNCTION

We now build connections between the above approximate gradients and the exact gradients given by the influence function (Koh & Liang, 2017). The influence function provides a way to calculate the gradient of $\phi^*(\theta)$ w.r.t. the generated samples $x_z$ as follows:

$$\frac{\partial \phi^*(\theta)}{\partial x_z} = -H_{\phi^*}^{-1} \nabla x_z \left( \frac{\partial f_E(\theta, \phi)}{\partial \phi} \bigg|_{\phi^*} \right),$$

where $H_{\phi^*}$ is the Hessian of the objective $f_E$ w.r.t. $\phi$ at $\phi^*$ and is negative semi-definite (Koh & Liang, 2017). However, calculating and inverting $H_{\phi^*}$ is computationally expensive.

Assuming that the parameters of the density estimator is at its optimum $\phi^*$, i.e., $\phi^0 = \phi^*$, we examine the case of $K = 1$. The result of one step unrolling is given by:

$$\frac{\partial \phi^1}{\partial x_z} = \frac{\partial (\phi^0 + \eta \nabla \phi)}{\partial x_z} = \eta \nabla x_z \left( \frac{\partial f_E(\theta, \phi)}{\partial \phi} \bigg|_{\phi^0} \right),$$

$$= \eta \nabla x_z \left( \frac{\partial f_E(\theta, \phi)}{\partial \phi} \bigg|_{\phi^0} \right),$$

\footnote{We have omitted the learning rate decay for simplicity.}
Algorithm 1 Stochastic Gradient Ascent Training of LBT with the Unrolling Technique

\begin{algorithm}
\textbf{Input:} data $x$, learning rate $\eta_{\theta}$ and $\eta_{\phi}$, unrolling steps $K$ and inner update iterations $M$.

Initialize parameters $\theta_0$ and $\phi_0$, and $t = 1$.

\begin{repeat}
\textbf{for} $i = 1$ to $M$ \textbf{do}
\begin{align*}
\phi^{i}_t &\leftarrow \varphi^{i-1} + \eta_{\phi} \cdot \frac{\partial f_K(\theta, \phi)}{\partial \phi} |_{\phi^{i-1}} \\
\end{align*}
\textbf{end for}

Update $\phi$: $\varphi^M_t \leftarrow \varphi^M_t$

Unrolling: $\varphi^K_t \leftarrow \varphi^0_t + \sum_{i=1}^{K} \eta_{\phi} \cdot \frac{\partial f_K(\theta, \phi)}{\partial \phi} |_{\phi^{i-1}}$

Update $\theta$: $\theta_t \leftarrow \theta_{t-1} + \eta_{\theta} \cdot \frac{\partial f_K(\phi^K)}{\partial \theta} |_{\phi^{i-1}}$

Update $t$: $t \leftarrow t + 1$
\end{repeat}

\textbf{until} Both $\theta$ and $\phi$ converge.
\end{algorithm}

Note that the inner product of $\frac{\partial \varphi^i}{\partial x^i}$ and $\frac{\partial \varphi^*}{\partial x^*}$ given by the influence function is positive because the Hessian $H_{\varphi^i}$ is negative semi-definite (Koh & Liang, 2017). Therefore, the unrolling technique essentially gives an approximation of the influence function in Eqn. (10) under the condition that the estimator is good enough (near to its optimum $\varphi^*$). Besides, the unrolling technique is much more efficient as it does not need to inverse the Hessian matrix.

4. Related Work

Implicit statistical models (Mohamed & Lakshminarayanan, 2016) are of great interests with the emergence of GAN (Goodfellow et al., 2014) that introduces a minimax framework to train such models. Nowozin et al. (2016) generalize the original GANs via introducing a broad class of $f$-divergence for optimization. In comparison, LBT provides a different way to optimize the KL-divergence and achieves good results on avoiding mode collapse and generating realistic samples when combined with GAN. Arjovsky et al. (2017) propose to minimize the earth mover’s distance to avoid the problem of gradient vanishing in vanilla GANs. Besides, Li et al. (2015) train implicit models by matching momentum between generated samples and real samples.

Mode collapse is a well-known problem in practical training of GANs. Much work has been done to alleviate the problem (Arjovsky & Bottou, 2017; Arjovsky et al., 2017; Mao et al., 2017; Metz et al., 2016; Srivastava et al., 2017). Unrolled GAN (Metz et al., 2016) proposes to unroll the update of the discriminator in GANs. The unrolling helps capturing how the discriminator would react to a change in the generator. Therefore it reduces the tendency of the generator to collapse all samples into a single mode. Srivastava et al. (2017) propose VEEGAN that introduces an additional reconstructor net to map data back to the noise space. Then a discriminator on the joint space is introduced to learn the joint distribution, similar as in ALI (Dumoulin et al., 2016). Lin et al. (2017) propose to modify the discriminator to distinguish whether multiple samples are real or generated.

Different from existing methods (Nowozin et al., 2016; Nguyen et al., 2017), our method evaluates the KL-divergence with data samples and makes $\varphi^*$ a function of $\theta$ via unrolling. The first property enables us to accurately evaluate the KL-divergence, regardless of whether the generator collapses or not. By unrolling the optimization process, the estimation of the KL-divergence can be differentiable w.r.t. the generator and can be optimized in practice.

We directly compare our method with these existing methods in our experiments.

5. Experiments

We now present the empirical results of LBT and LBT-GAN on both synthetic and real datasets. Throughout the experiments, we use Adam (Kingma & Ba, 2014) with the default setting to optimize both the generator and the estimator (and the discriminator for LBT-GAN). We set the unrolling steps $K = 5$ and set the inner update iterations $M = 15$ for the estimator. We use variational auto-encoders (VAEs) as the density estimators for both LBT and LBT-GAN. All the encoders and decoders in VAEs are two-hidden-layer MLPs. In LBT-GAN, we set $\lambda_G = 0.1$ for synthetic datasets and $\lambda_G = 1$ for real datasets. Our code is attached in the supplementary materials for reproducing.

5.1. Synthetic Datasets

We first compare LBT and LBT-GAN with state-of-the-art competitors (Goodfellow et al., 2014; Mao et al., 2017; Metz et al., 2016; Srivastava et al., 2017) on 2-dimensional (2D) synthetic datasets, which are convenient for qualitative and quantitative analysis. Specifically, we construct two datasets: (i) ring: mixture of 8 2D Gaussian distributions arranged in a ring and (ii) grid: mixture of 100 2D Gaussian distributions arranged in a 10-by-10 grid. All of the mixture components are isotropic Gaussian, i.e., with diagonal covariance matrix. For the ring data, the deviation of each Gaussian component is diag(0.1, 0.1) and the radius of the ring is 1 in 2. For the grid data, the spacing between adjacent modes is 0.2 and the deviation of each Gaussian component is diag(0.01, 0.01). Fig. 3a shows the true distributions of the ring data and the grid data, respectively. For fair comparison, we use generators with the same network architectures (two-hidden-layer MLPs) for all methods.

\footnote{In the original Unrolled GAN’s setting (Metz et al., 2016), the std of each component is 0.02 and the radius of the ring is 2. In our setting, the ratio of std to radius is 10 times larger. We choose this setting in order to characterize different performance of “Intra-mode KL-divergence” clearly.}
Learning Implicit Generative Models by Teaching Explicit Ones

For GAN-based methods, the discriminators are also two-hidden-layer MLPs. The numbers of the hidden units for the generators and the estimators (and the discriminators for LBT-GAN) are all 128.

To quantify the quality of the generator learned by different methods, we report 3 metrics to demonstrate different characteristics of generator distributions:

Percentage of High Quality Samples (Srivastava et al., 2017): We count a sample as a high quality sample of a mode if it is within three standard deviations of that mode. We say a sample is of high quality if it is a high quality sample of any modes. We generate 500,000 samples from each method and report the percentage of high quality samples.

Number of Modes Covered: We count a mode as a covered mode if the number of its high quality samples is greater than 20% of the expected number of that, i.e., \(20\% \times \frac{\text{# of samples}}{\text{# of modes}}\).

Intuitively, lower number of modes covered indicates more severe of mode collapse and a lack of global diversity.

Averaged Intra-Mode KL-Divergence: We assign each generated sample to the nearest mode of the true distribution. For each mode, we fit a Gaussian model on its assigned samples, which can be viewed as an estimate of the generator distribution at that mode (where the true distribution is approximately Gaussian). We define \textit{intra-mode KL-divergence} as the KL-divergence between the true distribution and the estimated distribution at each mode. Intuitively, it measures the local mismatch between the generator distribution and the true one. We report the averaged intra-mode KL-divergence over all modes.

Fig. 3 shows the generator distributions learned by different methods. Each distribution is plotted using kernel density estimation with 500,000 samples. We can see that LBT and LBT-GAN manage to cover the largest number of modes.

\[\text{Averaged Inner-Mode KL-Divergence} = \text{KL-divergence (lower the better); Percentage of high quality samples (the higher the better).}\]

\[\text{Number of Modes Covered} = \text{Number of modes covered (the higher the better); Averaged intra-mode KL-divergence (the lower the better); Percentage of high quality samples (the higher the better).}\]

\[\text{Percentage of High Quality Samples} = \text{Percentage of high quality samples (the higher the better).}\]

\[\text{# of samples} \div \text{# of modes}, \text{ according to the three-sigma rule.}\]
Table 1. Degree of mode collapse measured by number of mode captured and KL-divergence between the generated distribution over modes and the uniform distribution over 1,000 modes on Stacked MNIST. Results are averaged over 5 runs.

|               | ALI | Unrolled GAN | VEEGAN | DCGAN | PacGAN | D2GAN | LBT-GAN |
|---------------|-----|--------------|--------|-------|--------|-------|---------|
| # Modes Covered | 16  | 48.7         | 150    | 188.8 | 664.2  | 876.8 | 999.6   |
| KL            | 5.4 | 4.32         | 2.95   | 3.17  | 1.41   | 0.95  | 0.19    |

(a) Generated samples of DCGANs and LBT-GANs with different network architectures of discriminators. From left to right: DCGAN and LBT-GAN with a large discriminator; DCGAN and LBT-GAN with a small discriminator. LBT-GANs can successfully generate realistic and diverse samples with different network architectures of discriminators.

(b) The generated samples of LBT-GAN (Left) with the estimator being a smaller VAE and the samples from the estimator (Right).

Figure 5. Generative performance of LBT-GAN with different architectures of the discriminator and the estimator.

on both ring and grid datasets compared to other methods, demonstrating that LBT can help generate globally diverse samples. The quantitative results are included in Fig. 4a. Note that our method covers all the 100 modes on the grid dataset while the best competitors LSGAN and VEEGAN cover 88 modes and 79 modes, respectively. Moreover, the number of modes covered by LBT increases consistently during the training. On the contrary, Unrolled GAN and VEEGAN can sometimes drop the covered modes, attributed to their unstable training.

Fig. 4b shows the results of averaged intra-mode KL-divergence. We can see that LBT and LBT-GAN consistently outperform other competitors, which demonstrates that LBT framework can help capture better intra-mode diversity. According to Fig. 3c and Fig. 3e, although LSGAN and VEEGAN can achieve good mode coverage, they tend to concentrate most of the density near the mode means and fail to capture the local diversity within each mode. In LBT-GAN, the discriminator has a similar effect, while the estimator prevents the generator to over-concentrate the density. Therefore, the Intra-mode KL-divergence of LBT-GAN may oscillate during training as in Fig. 4b.

Finally, we show the percentages of high quality samples for each method in Fig. 4c. We find that LBT-GAN achieves better results than LBT and outperforms other competitors. As LBT-GAN can generate high quality samples while maintaining the global and local mode coverage, we use LBT-GAN in the following experiments.

5.2. Stacked Mnist

Stacked MNIST (Metz et al., 2016) is a variant of the MNIST (LeCun et al., 1998) dataset created by stacking three randomly selected digits along the color channel to increase the number of discrete modes. There are 1,000 modes corresponding to 10 possible digits in each channel. Following (Metz et al., 2016; Srivastava et al., 2017), we randomly stack 128,000 samples serving as the training data and use 26,000 generated samples to calculate the number of modes to which at least one sample belongs. We use a classifier trained on the original MNIST data to identify digits in each channel of generated samples. Besides, we also report the KL-divergence between the generated distribution over modes and the uniform distribution. Since reasonably fine-tuned GAN can generate 1000 modes (Metz et al., 2016), we select much smaller convolutional networks as both the generator and discriminator making our setting comparable to the competitors. For the estimator of LBT-GAN, the number of hidden units of the two-hidden-layer MLP decoder and encoder of VAE are both 1000-400.

Table 1 presents the quantitative results. In terms of the number of captured modes, LBT-GAN surpasses other competitors, which demonstrates the effectiveness of the LBT framework. Specifically, LBT-GAN can successfully capture almost all modes, and the results of KL-divergence indicate that the distribution of LBT-GAN over modes is much more balanced compare to other competitors. Note that PacGAN and D2GAN also report comparable results with ours on different network architectures whereas they fail to capture all modes in our setting. In contrast, LBT-GAN can generalize to PacGAN’s architecture and capture all 1000 modes. Our hypothesis is that the auxiliary estimator helps LBT-GAN generalize across different architectures.

Fig. 5a shows the generated samples of GANs and LBT-GANs with different size of discriminators. The visual quality of the samples generated by LBT-GANs is better than GANs. Further, we find the sample quality of DCGANs...
Learning Implicit Generative Models by Teaching Explicit Ones

is sensitive to the size of the discriminators, while LBT-GANs can generate high-quality samples under different network architectures.

Furthermore, we implement LBT-GAN with a much smaller VAE where both the encoder and the decoder of the estimator are two-hidden-layer MLPs with only 20 units in each hidden layer. Fig. 5b shows that the samples from this VAE are of poor quality, which means that it can hardly capture the distribution of Stacked-MNIST. Nevertheless, even with such a simple VAE, LBT-GAN can still capture 1,000 modes and generate visually realistic samples (See the left panel of Fig. 5b), verifying that an estimator with limited capability can still help our method avoid mode collapse.

5.3. CelebA & CIFAR10
We also evaluate LBT-GAN on natural images, including CIFAR10 (Krizhevsky & Hinton, 2009) and CelebA (Liu et al., 2015) datasets. The generated samples of DCGANs and LBT-GANs are illustrated in Fig. 6. LBT-GAN can generate images with comparable quality as DCGANs, demonstrating that LBT-GAN can successfully scale to natural images.

5.4. Sensitivity Analysis of $K$ and $M$
Theoretically, a larger unrolling steps $K$ allows $\phi^K$ to better approximate $\phi^*$ and a larger inner update iterations $M$ can better approximate the condition in Eqn. (11) as analyzed in Sec. 3.3. However, large $K$ and $M$ on the other hand increase the computational costs. To balance this trade-off, we provide sensitivity analysis of $K$ and $M$ in LBT. We use the experimental settings of the ring problem and adopt the values of the objective function Eqn. (2), i.e., the log-likelihood of real samples evaluated by the learned estimator, as the quantitative measurement.

We first investigate the influence of the number of unrolling steps $K$ on the training procedure. We vary the value of $K$ and show the learning curves with $K \in \{1, 3, 5, 15\}$ in Fig. 7. We observe that $K = 1$ leads to a suboptimal solution and larger $K$ leads to better solution and convergence speed. We do not observe significant improvement with $K$ larger than 5. We also show the influence of the number of inner update iterations $M$ during training with $M \in \{5, 10, 15, 50\}$ in Fig. 7. Our observation is that larger $M$ leads to faster convergence, which is consistent with the analysis in Sec. 3.3.

6. Conclusions & Discussions
We present a novel framework LBT to train an implicit generative model via teaching an auxiliary likelihood estimator, which is formulated as a bilevel optimization problem. Unrolling techniques are adopted for practical optimization. Finally, LBT is justified both theoretically and empirically.

The main bottleneck of LBT is how to efficiently solve the bilevel optimization problem. For one thing, each update of LBT could be slower than that of the existing methods because the computational cost of the unrolling technique grows linearly with respect to the unrolling steps. For another, LBT may need larger number of updates to converge than GAN because training a density estimator is more complicated than training a classifier. Overall, if the bilevel optimization problem can be solved efficiently in the future work, LBT can be scaled up to larger datasets.

LBT bridges the gap between the training of implicit models and explicit models. For one thing, the auxiliary explicit models can help implicit models overcome the mode collapse problems. For another, the implicit generators can be viewed as approximated samplers of the density estimators like Pixel-CNNs (Oord et al., 2016), from which getting samples is time-consuming. We discuss the former direction in this paper and leave the later direction as future work.
References

Arjovsky, M. and Bottou, L. Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862, 2017.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In International Conference on Machine Learning, pp. 214–223, 2017.

Colson, B., Marcotte, P., and Savard, G. An overview of bilevel optimization. Annals of operations research, 153 (1):235–256, 2007.

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. Adversarially learned inference. arXiv preprint arXiv:1606.00704, 2016.

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

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In Advances in neural information processing systems, pp. 2672–2680, 2014.

Huszár, F. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? arXiv preprint arXiv:1511.05101, 2015.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. arXiv preprint arXiv:1703.04730, 2017.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

Li, Y., Swersky, K., and Zemel, R. Generative moment matching networks. In International Conference on Machine Learning, pp. 1718–1727, 2015.

Lin, Z., Khetan, A., Fant, G., and Oh, S. Pacgan: The power of two samples in generative adversarial networks. arXiv preprint arXiv:1712.04086, 2017.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), December 2015.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Smolley, S. P. Least squares generative adversarial networks. In IEEE International Conference on Computer Vision (ICCV), pp. 2813–2821. IEEE, 2017.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks. arXiv preprint arXiv:1611.02163, 2016.

Mohamed, S. and Lakshminarayanan, B. Learning in implicit generative models. arXiv preprint arXiv:1610.03483, 2016.

Nasrabadi, N. M. Pattern recognition and machine learning. Journal of electronic imaging, 16(4):049901, 2007.

Nguyen, T., Le, T., Vu, H., and Phung, D. Dual discriminator generative adversarial nets. In Advances in Neural Information Processing Systems, pp. 2670–2680, 2017.

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

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. arXiv preprint arXiv:1601.06759, 2016.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

Srivastava, A., Valkoz, L., Russell, C., Gutmann, M. U., and Sutton, C. Veegan: Reducing mode collapse in gans using implicit variational learning. In Advances in Neural Information Processing Systems, pp. 3310–3320, 2017.

Theis, L., Oord, A. v. d., and Bethge, M. A note on the evaluation of generative models. arXiv preprint arXiv:1511.01844, 2015.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. Wasserstein auto-encoders. arXiv preprint arXiv:1711.01558, 2017.
A. Proof of Theorem 1.

Proof. For a fixed generator $G$, the objective of the estimator is to maximize the log-likelihood of the generated samples:

$$
\mathbb{E}_{x \sim p_G(x; \theta)} [\log p_E(x; \phi)]
$$

$$
= \mathbb{E}_{x \sim p_G(x; \theta)} [\log \frac{p_E(x; \phi)}{p_G(x; \theta)}] + \mathbb{E}_{x \sim p_G(x; \theta)} [\log p_G(x; \theta)]
$$

$$
= -KL(p_G(x; \theta) \| p_E(x; \phi)) - H(p_G(x; \theta)),
$$

where $H(p_G(x; \theta))$ is the entropy of the generator distribution, which is a constant with respect to the estimator $E$. Hence, maximizing the above objective is equivalent to minimizing the KL-divergence between $p_G$ and $p_E$. The estimator $E$ thus achieves its optimum when $p_E = p_G$, i.e., $p_E(x; \phi^*(\theta)) = p_G(x; \theta)$. Then it is straightforward that maximizing Eqn. (2) is equivalent to solving the problem: $\max_\theta \mathbb{E}_{x \sim p_D(x)} [\log p_G(x; \theta)]$, which is equivalent to minimizing the KL-divergence between $p_D(x)$ and $p_G(x; \theta)$, whose optimum is achieved when $p_G = p_D$. \qed