Learning Neural Random Fields with Inclusive Auxiliary Generators

Yunfu Song and Zhijian Ou, *Senior Member, IEEE*

Abstract—Neural random fields (NRFs), which are defined by using neural networks to implement potential functions in undirected models (sometimes known as energy-based models), provide an interesting family of model spaces for machine learning, besides various directed models such as generative adversarial networks (GANs). In this paper we propose a new approach, the inclusive-NRF approach, to learning NRFs for continuous data (e.g. images), by developing inclusive-divergence minimized auxiliary generators and stochastic gradient sampling. As demonstrations of how the new approach can be flexibly and effectively used, specific inclusive-NRF models are developed and thoroughly evaluated for a number of tasks - unsupervised/supervised image generation, semi-supervised classification and anomaly detection. The proposed models consistently achieve strong experimental results in all these tasks compared to state-of-the-art methods. Remarkably, in addition to superior sample generation, one fundamental additional benefit of our inclusive-NRF approach is that, unlike GANs, it directly provides (unnormalized) density estimate for sample evaluation. With these contributions and results, this paper significantly advances the learning and applications of undirected models to a new level, both theoretically and empirically, which have never been obtained before.

Index Terms—Markov random fields, Undirected graphical models, Deep generative models, Image generation, Semi-supervised learning, Anomaly detection.

1 Introduction

One of the core research problems in machine learning is learning with probabilistic models, which can be broadly classified into two classes - directed and undirected graphical models [1], [2]. Apart from the topology difference, an easy way to tell an undirected model from a directed model is that an undirected model involves the normalizing constant (also called the partition function in physics), while the directed model is self-normalized. Recently, significant progress has been made on learning with deep generative models (DGMs), which generally refer to probabilistic models with multiple layers of stochastic or deterministic variables. There have emerged a bundle of deep directed models, such as variational AutoEncoders (VAEs) [3], generative adversarial networks (GANs) [4] and so on. In contrast, undirected models (also known as random fields [2], energy-based models [5]) received less attention with slow progress. This is presumably because fitting undirected models is more challenging than fitting directed models. In general, calculating the log-likelihood and its gradient is analytically intractable, because it involves evaluating the normalizing constant and, respectively, the expectation with respect to (w.r.t.) the model distribution.

In this paper, we are interested in developing deep undirected models. Generally, an undirected model, or exchangeably termed as a random field (RF), defines a probability distribution of the form $p_\theta(x) = \frac{1}{Z(\theta)} \exp\left[u_\theta(x)\right]$, where $u_\theta(x)$ is usually called the potential function over observation x with parameter θ, and $Z(\theta) = \int \exp\left[u_\theta(x)\right] dx$ is the normalizing constant. In most existing random fields, the potential function $u_\theta(x)$ is often defined as linear functions, e.g. $u_\theta(x) = \theta^T f(x)$, where $f(x)$ is a vector of features (usually hand-crafted) and θ is the corresponding parameter vector. Such RFs are known as log-linear models [2] or exponential families [6].

Note that an attractive property of RF modeling is that one is free to define the potential function in any sensible way, giving it much flexibility. In this paper, we aims to advance the learning of neural random fields, which use neural networks with multiple deterministic layers to define the potential function $u_\theta(x)$. With the potential benefit of exploiting the expressive power of deep neural networks, this type of RFs appeared several times in different contexts with different model definitions, called deep energy models (DEMs) [7], [8], descriptive models [9], generative ConvNet [10], neural random field language models [11]. For convenience, we refer to such models as **neural random fields (NRFs)** in general. Conceptually, compared to traditional log-linear RFs, if we could successful train such NRFs, we can jointly learn the features and the feature weights, which is highly desirable. However, learning NRFs presents much greater challenge.

An important method of maximum likelihood (ML) learning of random fields is called stochastic maximum likelihood (SML) [12], which approximates the model expectations by Monte Carlo sampling for calculating the gradient. A recent progress in learning NRFs as studied in [8], [9], [11], [13] is to pair the target random field p_θ with an auxiliary directed generative model (often called generator) $q_\phi(x)$ parameterized by ϕ, which approximates sampling from the target random field. Learning is performed by maximizing the log-likelihood of training data under p_θ or some bound of the log-likelihood, and simultaneously minimizing some divergence between the target random field p_θ and the auxiliary generator q_ϕ. Different learning algorithms differ in the objective functions used in the joint training of p_θ and q_ϕ, and thus have different computational and statistical properties (partly illustrated in Figure 3). For example, minimizing the **exclusive-divergence** $KL[q_\phi||p_\theta] \triangleq \int q_\phi \log\left(q_\phi/p_\theta\right) = -H\left[q_\phi\right] - \int q_\phi \log p_\theta$ w.r.t.

• *Yunfu Song and Zhijian Ou are with the Department of Electronic Engineering, Tsinghua university, Beijing, China. Email: ozj@tsinghua.edu.cn*

Manuscript received XX, 2019. Corresponding author: Zhijian Ou.

ϕ, as employed in [8], involves the intractable entropy term $H[q_\phi]$ and tends to enforce the generator to seek modes, yielding missing modes. There are also other factors, e.g. modeling discrete or continuous data, different model choices of the target RF and the generator, which lead to different learning algorithms. We leave detailed comparison and connection of our approach with existing studies to Section 4 (Related work).

In this paper, we propose to use inclusive-divergence minimized auxiliary generators (Section 3.2). And particularly for continuous data (e.g. images), we propose to use stochastic gradient samplers, including but not limited to SGLD (stochastic gradient Langevin dynamics) and SGHMC (stochastic gradient Hamiltonian Monte Carlo), to exploit noisy gradients in NRF model sampling (Section 3.3). Within our formulation, SGHMC is successfully developed, which improves over SGLD in learning NRFs. Notably, this SGLD/SGHMC development is not like in previous applications ([14], [15]) which mainly simulate Bayesian posterior samples in large-scale Bayesian inference, though we use the same terms.

Featured by developing inclusive auxiliary generators and stochastic gradient sampling, this new approach to learning NRFs for continuous data, abbreviated as the **inclusive-NRF** approach, is the main contribution of this paper. Conceptually, minimizing the **inclusive-divergence** $KL[p_\theta||q_\phi] \triangleq \int p_\theta \log(p_\theta/q_\phi)$ w.r.t. ϕ avoids the annoying entropy term and tends to drive the generator to cover modes of the target density p_θ. The SGLD/SGHMC sampling further pushes the samples towards the modes of p_θ. Presumably, this helps to produce Markov chains that mix fast between modes and facilitate model learning.

As demonstrations of how the new approach can be flexibly and effectively used, specific inclusive-NRF models are developed and thoroughly evaluated for a number of tasks - unsupervised/supervised image generation, semi-supervised classification and anomaly detection. The proposed models consistently achieve strong experimental results in all these tasks compared to state-of-the-art methods, which are summarized as follows:

- Inclusive-NRFs achieve state-of-the-art sample generation quality, measured by both Inception Score (IS) and Frechet Inception Distance (FID). On CIFAR-10, we obtain unsupervised IS 8.28 (FID 20.9) and supervised IS 9.06 (FID 18.1), both using unconditional generation.
- Semi-supervised inclusive-NRFs show strong classification results on par with state-of-the-art DGM-based semi-supervised learning (SSL) methods, and simultaneously achieve superior generation, on the widely benchmarked datasets - MNIST, SVHN and CIFAR-10.
- By directly using the potential function for sample evaluation, inclusive-NRFs achieve state-of-the-art performance in anomaly detection on the widely benchmarked datasets - KDDCUP, MNIST, and CIFAR-10. This shows that, unlike GANs, the new approach can provide informative density estimate, besides superior sample generation.

The remainder of this paper is organized as follows. After presenting some background on random fields in Section 2, we introduce the inclusive-NRF approach. In Section 4, we discuss related work. The extensive experimental evaluations are given in Sections 5 . We conclude the paper with a discussion in Section 6.

2 Background on Random fields

Undirected models, or exchangeably termed as random fields form one of the two main classes of probabilistic graphical models [1], [2]. In defining the joint distribution, directed models use conditional probability functions, with the directionality given by the conditioning relationship, whereas undirected models use unnormalized potential functions and are more suitable for capturing interactions among variables, especially when the directionality of a relationship cannot be clearly defined (e.g. as in between neighboring image pixels).

A random field (RF) defines a probability distribution for a collection of random variables $x \in \mathbb{R}^{d_x}$ with parameter θ in the form:

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp\left[u_\theta(x)\right] \tag{1}$$

where $Z(\theta) = \int \exp\left[u_\theta(x)\right] dx$ is the normalizing constant, $u_\theta(x)$ is called the potential function[1] which assigns a scalar value to each configuration of x. High probability configurations correspond to high potential/low energy configurations.

There is an extensive literature devoted to maximum likelihood (ML) learning of random fields, as briefly reviewed in [16]. It is usually intractable to maximize the data log-likelihood $logp_\theta(\tilde{x})$ for observed $\tilde{x}$, since the gradient involves expectation w.r.t. the model distribution, as shown below:

$$\begin{aligned} \nabla_\theta \log p_\theta(\tilde{x}) &= \nabla_\theta u_\theta(\tilde{x}) - \nabla_\theta \log Z(\theta) \\ &= \nabla_\theta u_\theta(\tilde{x}) - E_{p_\theta(x)}\left[\nabla_\theta u_\theta(x)\right]. \end{aligned} \tag{2}$$

In graphical modeling terminology, without loss of generality, let each component of x indexed by a node in a graph. The whole potential function $u_\theta(x)$ is defined to be decomposed over cliques of the graph (i.e., fully connected subsets of nodes):

$$u_\theta(x) = \sum_{c \in \mathcal{C}} u_\theta(x_c) \tag{3}$$

where $\mathcal{C}$ denotes the collection of cliques in the graph, and $u_\theta(x_c)$ denotes the clique potential defined over the vector of variables indexed by the clique c. Such decomposition reduces the complexity of model representation but maybe at the sacrifice of model expressive capacity, and should respect the inherent modularity in the interactions among variables. In previous studies, particularly in computer vision tasks, grid-like RFs are mostly used; higher-order RFs have been pursued but most are in fact conditional random fields (CRFs) [17]. Notably, CRFs can only be used for discriminative tasks, e.g. segmenting and labeling of natural language sentences or images, and usually have a reduced sample spaces of labels. Thus, the learning algorithms developed in CRFs are usually not applicable to unconditional RFs.

3 The inclusive-NRF approach

A high-level overview of our inclusive-NRF approach is shown in Figure 1. In the following, after introducing the NRF model (Section 3.1), the two new designs - introducing the inclusive-divergence minimized auxiliary generator and developing stochastic gradient sampling are elaborated in Section 3.2 and Section 3.3 respectively.

3.1 The NRF model

The general idea of neural random fields (NRFs) is to implement the potential $u_\theta(x) : \mathbb{R}^{d_x} \rightarrow \mathbb{R}$, by a neural network, which takes the multi-dimensional $x \in \mathbb{R}^{d_x}$ as input and outputting the scalar $u_\theta(x) \in \mathbb{R}$. In this manner, we can take advantage of the

1. Negating the potential function defines the energy function.

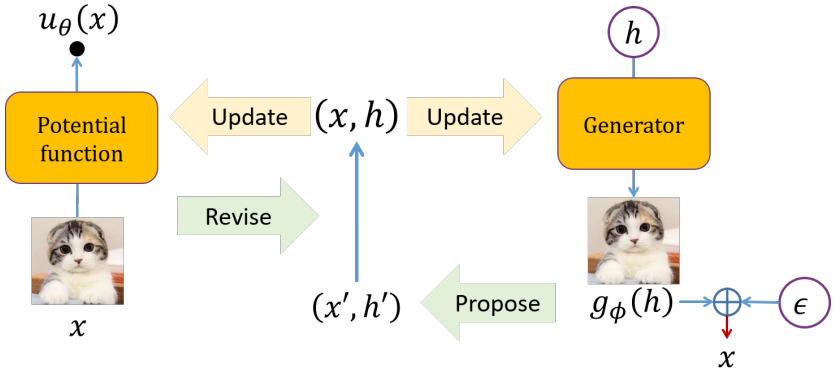


Fig. 1: Overview of the inclusive-NRF approach. Two neural networks are used to define the NRF's potential function $u_\theta(x)$ and the auxiliary generator $g_\phi(h)$ respectively. Both network parameters, θ and ϕ, are updated by using the revised samples (x, h), which are obtained by revising the samples (x', h') proposed by the auxiliary generator, according to the RF's stochastic gradients.

representation power of neural networks for RF modeling. And such a RF essentially becomes defined over a fully-connected undirected graph and captures interactions in observations to the largest order, since the neural potential function $u_\theta(x)$ involves all the components in x. Remarkably, the NRFs used in our experiments are different from similar models in previous studies [8], [11], [9], as detailed in Section 4.

3.2 Introducing inclusive-divergence minimized auxiliary generators

As shown in Eq. (2), the bottleneck in learning NRFs is that Monte Carlo sampling from the RF model is needed to approximate the model expectation for calculating the gradient. A recent idea is to introduce an auxiliary generator to approximate sampling from the target RF. In this paper, we are mainly concerned with modeling fixed-dimensional continuous observations $x \in \mathbb{R}^{d_x}$ (e.g. images). For reasons to be clear in the following, we choose a directed generative model, $q_\phi(x, h) \triangleq q(h)q_\phi(x|h)$, for the auxiliary generator, which specifically is defined as follows[2]:

$$\begin{aligned} &h \sim \mathcal{N}(0, I_h), \\ &x = g_\phi(h) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2 I_\epsilon), \end{aligned} \tag{4}$$

where $g_\phi(h) : \mathbb{R}^{d_h} \to \mathbb{R}^{d_x}$ is implemented as a neural network with parameter ϕ, which maps the latent code h to the observation space. I_h and I_ϵ denote the identity matrices, with dimensionality implied by h and ϵ respectively. Drawing samples from the generator $q_\phi(x, h)$ is simple as it is just ancestral sampling from a 2-variable directed graphical model. This is one reason for choosing Eq. (4) as the generator.

For dataset $\mathcal{D} = \{\tilde{x}_1, \cdots, \tilde{x}_n\}$, consisting of n observations, let $\tilde{p}(\tilde{x}) \triangleq \frac{1}{n}\sum_{k=1}^{n} \delta(\tilde{x} - \tilde{x}_k)$ denotes the empirical data distribution. A new design in this paper is that we perform the maximum likelihood learning of p_θ and simultaneously minimize the inclusive divergence between the target random field p_θ and the auxiliary generator q_ϕ by[3]

$$\begin{cases} \min\limits_{\theta} KL\left[\tilde{p}(\tilde{x})||p_\theta(\tilde{x})\right] \\ \min\limits_{\phi} KL\left[p_\theta(x)||q_\phi(x)\right] \end{cases} \tag{5}$$

2. Note that during training, σ^2 is absorbed into the learning rates and does not need to be estimated.

3. Such optimization using two objectives is employed in a number of familiar learning methods, such as GAN with $\log D$ trick [4], wake-sleep algorithm [18].

The first line of Eq. (5) is equivalent to maximum likelihood training of the target RF p_θ under the empirical data $\tilde{p}$, which requires sampling from p_θ. Simultaneously, the second line optimizes the generator q_ϕ to be close to p_θ so that q_ϕ becomes a good proposal for sampling from p_θ. By Proposition 1, we can derive the gradients w.r.t. θ and ϕ (to be ascended) as follows:

$$\begin{cases} \nabla_\theta = E_{\tilde{p}(\tilde{x})}\left[\nabla_\theta log p_\theta(\tilde{x})\right] \\ \quad\quad = E_{\tilde{p}(\tilde{x})}\left[\nabla_\theta u_\theta(\tilde{x})\right] - E_{p_\theta(x)}\left[\nabla_\theta u_\theta(x)\right], \\ \nabla_\phi = E_{p_\theta(x)}\left[\nabla_\phi log q_\phi(x)\right] \\ \quad\quad = E_{p_\theta(x)q_\phi(h|x)}\left[\nabla_\phi log q_\phi(x, h)\right]. \end{cases} \tag{6}$$

In practice, we apply minibatch based stochastic gradient descent (SGD) to solve the optimization problem Eq. (5), as shown in Algorithm 1. Notably, choosing Eq. (4) as the generator yields tractable gradients for the inclusive-divergence minimization.

Proposition 1. *Both lines of Eq.(6) for gradient calculations hold.*

Proof. The first line of Eq.(6) can be obtained by directly taking derivative of $KL\left[\tilde{p}(\tilde{x})||p_\theta(\tilde{x})\right]$ w.r.t. θ, as shown below,

$$\begin{aligned} \frac{\partial}{\partial\theta} KL\left[\tilde{p}(\tilde{x})||p_\theta(\tilde{x})\right] &= \frac{\partial}{\partial\theta}\int \tilde{p}(\tilde{x}) \log \frac{\tilde{p}(\tilde{x})}{p_\theta(\tilde{x})} d\tilde{x} \\ &= -\int \tilde{p}(\tilde{x}) \frac{\partial}{\partial\theta} \log p_\theta(\tilde{x}) d\tilde{x}, \end{aligned}$$

and then applying the basic formula of Eq. (2).

For the second line, by direct calculation, we first have

$$\begin{aligned} &E_{q_\phi(h|x)}\left[\nabla_\phi \log q_\phi(h|x)\right] \\ &= \int q_\phi(h|x) q_\phi(h|x)^{-1} \nabla_\phi q_\phi(h|x) dh \\ &= \int \nabla_\phi q_\phi(h|x) dh = \nabla_\phi \int q_\phi(h|x) dh = \nabla_\phi 1 = 0. \end{aligned}$$

Then combining

$$\frac{\partial}{\partial\phi} KL\left[p_\theta(x)||q_\phi(x)\right] = -E_{p_\theta(x)}\left[\nabla_\phi \log q_\phi(x)\right]$$

and

$$\begin{aligned} \nabla_\phi \log q_\phi(x) &= E_{q_\phi(h|x)}\left[\nabla_\phi \log q_\phi(x)\right] \\ &= E_{q_\phi(h|x)}\left[\nabla_\phi \log q_\phi(x, h) - \nabla_\phi \log q_\phi(h|x)\right] \\ &= E_{q_\phi(h|x)}\left[\nabla_\phi \log q_\phi(x, h)\right]. \end{aligned}$$

will give the second line of Eq.(6). □

Algorithm 1 Learning NRFs with inclusive auxiliary generators

repeat
Sampling: Draw a minibatch $\mathcal{M} = \{(\tilde{x}^i, x^i, h^i), i = 1, \cdots |\mathcal{M}|\}$ from $\tilde{p}(\tilde{x})p_\theta(x)q_\phi(h|x)$ (see Algorithm 2);
Updating:
Update θ by ascending: $\frac{1}{|\mathcal{M}|}\sum_{(\tilde{x},x,h)\sim\mathcal{M}}[\nabla_\theta u_\theta(\tilde{x}) - \nabla_\theta u_\theta(x)]$;
Update ϕ by ascending: $\frac{1}{|\mathcal{M}|}\sum_{(\tilde{x},x,h)\sim\mathcal{M}}\nabla_\phi \log q_\phi(x,h)$;
until convergence

3.3 Developing stochastic gradient samplers for NRF model sampling

In Algorithm 1, we need to draw samples $(x, h) \in \mathbb{R}^{d_x \times d_h}$ from our target distribution $p_\theta(x)q_\phi(h|x)$ given current θ and ϕ. For such continuous distribution, samplers leveraging continuous dynamics (namely continuous-time Markov processes described by stochastic differential equations), such as Langevin dynamics (LD) and Hamiltonian Monte Carlo (HMC) [19], are known to be efficient in exploring the continuous state space. Simulating the continuous dynamics leads to the target distribution as the stationary distribution. The Markov transition kernel defined by the continuous dynamical system usually involves using the gradients of the target distribution, which in our case are as follows:

$$\begin{cases} \frac{\partial}{\partial x}\log\left[p_\theta(x)q_\phi(h|x)\right] \\ \quad = \frac{\partial}{\partial x}\left[\log p_\theta(x) + \log q_\phi(h,x) - \log q_\phi(x)\right] \\ \frac{\partial}{\partial h}\log\left[p_\theta(x)q_\phi(h|x)\right] = \frac{\partial}{\partial h}\log q_\phi(h,x) \end{cases} \quad (7)$$

It can be seen that while it is straightforward to calculate the gradient w.r.t. h and the first two terms[4] in the gradient w.r.t. x, the gradient w.r.t. x consists of an intractable term $\frac{\partial}{\partial x}\log q_\phi(x)$. Therefore we are interested in developing stochastic gradient variants of continuous-dynamics samplers, which rely on using noisy estimate of $\frac{\partial}{\partial x}\log q_\phi(x)$.

Recently, stochastic gradient samplers have emerged in simulating posterior samples in large-scale Bayesian inference, such as SGLD (stochastic gradient Langevin dynamics) [14] and SGHMC (Stochastic Gradient Hamiltonian Monte Carlo) [15]. To illustrate, consider the posterior $p(\theta|\mathcal{D})$ of model parameters θ given the observed dataset $\mathcal{D}$, with abuse of notation. We have $p(\theta|\mathcal{D}) \propto \exp\left[\sum_{x\in\mathcal{D}}\log p_\theta(x) + \log p(\theta)\right]$, which is taken as the target distribution. Instead of using full-data gradients $\frac{\partial}{\partial\theta}\log p(\theta|\mathcal{D})$, which needs a sweep over the entire dataset, these samplers subsample the dataset and use stochastic gradients $\frac{\partial}{\partial\theta}\left[\frac{|\tilde{\mathcal{D}}|}{|\mathcal{D}|}\sum_{x\in\tilde{\mathcal{D}}}\log p_\theta(x) + \log p(\theta)\right]$ in the dynamic simulation, where $\tilde{\mathcal{D}} \subset \mathcal{D}$ is a subsampled data subset. In this manner, the computation cost is significantly reduced in each iteration and such Bayesian inference methods scale to large datasets.

In practice, sampling is based on a discretization of the continuous dynamics. Despite the discretization error and the noise introduced by the stochastic gradients, it can be shown that simulating the discretized dynamics with stochastic gradients also leads to the target distribution as the stationary distribution, when the step sizes are annealed to zero at a certain rate. The convergence of SGLD/SGHMC is provided in Theorem 1, which are summarized from [20], [15], [21].

4. Notably, $\frac{\partial}{\partial x}\log p_\theta(x) = \frac{\partial}{\partial x}u_\theta(x)$ does not require the calculation of the normalizing constant.

Theorem 1. *Denote the target density as $p(z;\lambda)$ with given λ. Assume that one can compute a noisy, unbiased estimate $\Delta(z;\lambda)$ (a stochastic gradient) to the gradient $\frac{\partial}{\partial z}\log p(z;\lambda)$. For a sequence of asymptotically vanishing time-steps $\{\delta_l, l \geq 1\}$ (satisfying $\sum_{l=1}^\infty \delta_l = \infty$ and $\sum_{l=1}^\infty \delta_l^2 < \infty$) and an i.i.d. noise sequence $\eta^{(l)}$, the SGLD iterates as follows, starting from $z^{(0)}$:*

$$\begin{aligned} z^{(l)} =& z^{(l-1)} + \delta_l\Delta(z^{(l-1)};\lambda) + \sqrt{2\delta_l}\eta^{(l)}, \\ & \eta^{(l)} \sim \mathcal{N}(0, I), l = 1, \cdots \end{aligned} \quad (8)$$

Starting from $z^{(0)}$ and $v^{(0)} = 0$, the SGHMC iterates as follows:

$$\begin{cases} v^{(l)} =(1-\beta)v^{(l-1)} + \delta_l\Delta(z^{(l-1)};\lambda) + \sqrt{2\beta\delta_l}\eta^{(l)}, \\ \qquad \eta^{(l)} \sim \mathcal{N}(0, I) \\ z^{(l)} = z^{(l-1)} + v^{(l)}, l = 1, \cdots \end{cases} \quad (9)$$

The iterations of Eq. (8) and (9) lead to the target distribution $p(z;\lambda)$ as the stationary distribution.

By considering $z \triangleq (x, h)$, $p(z;\lambda) \triangleq p_\theta(x)q_\phi(h|x)$, $\lambda \triangleq (\theta, \phi)^T$, and Eq. (7), we can use Theorem 1 to develop the sampling step for Algorithm 1, as presented in Algorithm 2. For the gradient w.r.t. x, the intractable term $\frac{\partial}{\partial x}\log q_\phi(x)$ could be estimated by a stochastic gradient. Motivated by observing

$$\frac{\partial}{\partial x}\log q_\phi(x) = E_{h^*\sim q_\phi(h^*|x)}\left[\frac{\partial}{\partial x}\log q_\phi(h^*, x)\right], \quad (10)$$

as proved in Proposition 2, ideally we draw $h^* \sim q_\phi(h^*|x)$ and then use $\frac{\partial}{\partial x}\log q_\phi(h^*, x)$ as an unbiased estimator of $\frac{\partial}{\partial x}\log q_\phi(x)$. In practice, at step l, given $x^{(l-1)}$ and starting from $h^{(l-1)}$, we run one step of LD sampling over h targeting $q_\phi(h|x^{(l-1)})$, to obtain $h^{(l-1)*}$ and calculate $\frac{\partial}{\partial x^{(l-1)}}\log q_\phi(h^{(l-1)*}, x^{(l-1)})$. This gives a biased but tractable estimator to $\frac{\partial}{\partial x}\log q_\phi(x)$. It is empirically found in our experiments that more steps of this inner LD sampling do not significantly improve the performance for NRF learning.

Proposition 2. *Eq. (10) holds.*

Proof.

$$\begin{aligned} \frac{\partial}{\partial x}\log q_\phi(x) &= E_{q_\phi(h^*|x)}\left[\frac{\partial}{\partial x}\log q_\phi(x)\right] \\ &= E_{q_\phi(h^*|x)}\left[\frac{\partial}{\partial x}\log q_\phi(x, h^*) - \frac{\partial}{\partial x}\log q_\phi(h^*|x)\right] \\ &= E_{q_\phi(h^*|x)}\left[\frac{\partial}{\partial x}\log q_\phi(x, h^*)\right]. \end{aligned}$$

□

Algorithm 2 Sampling from $p_\theta(x)q_\phi(h|x)$

1. Do ancestral sampling by the generator, namely first drawing $h' \sim p(h')$, and then drawing $x' \sim q_\phi(x'|h')$;
2. Starting from $(x', h') = z^{(0)}$, run finite steps of SGLD/SGHMC ($l = 1, \cdots, L$) to obtain $(x, h) = z^{(L)}$, which we call *sample revision*, according to Eq. (8) or (9).

In particular, the SGLD recursions are conducted as follows:

$$\begin{cases} x^{(l)} = x^{(l-1)} + \delta_l \dfrac{\partial}{\partial x^{(l-1)}} \left[\log p_\theta(x^{(l-1)}) + \log q_\phi(h^{(l-1)}, x^{(l-1)}) - \log q_\phi(h^{(l-1)*}, x^{(l-1)})\right] + \sqrt{2\delta_l}\eta_x^{(l)}, \\ h^{(l)} = h^{(l-1)} + \delta_l \dfrac{\partial}{\partial h^{(l-1)}} \log q_\phi(h^{(l-1)}, x^{(l-1)}) + \sqrt{2\delta_l}\eta_h^{(l)}, \quad \eta^{(l)} \triangleq (\eta_x^{(l)}, \eta_h^{(l)})^T \sim \mathcal{N}(0, I) \end{cases} \tag{11}$$

The SGHMC recursions are conducted as follows:

$$\begin{cases} v_x^{(l)} = (1-\beta)v_x^{(l-1)} + \delta_l \dfrac{\partial}{\partial x^{(l-1)}} \left[\log p_\theta(x^{(l-1)}) + \log q_\phi(h^{(l-1)}, x^{(l-1)}) - \log q_\phi(h^{(l-1)*}, x^{(l-1)})\right] + \sqrt{2\beta\delta_l}\eta_x^{(l)}, \\ v_h^{(l)} = (1-\beta)v_h^{(l-1)} + \delta_l \dfrac{\partial}{\partial h^{(l-1)}} \log q_\phi(h^{(l-1)}, x^{(l-1)}) + \sqrt{2\beta\delta_l}\eta_h^{(l)}, \\ x^{(l)} = x^{(l-1)} + v_x^{(l)}, \quad h^{(l)} = h^{(l-1)} + v_h^{(l)}, \quad \eta^{(l)} \triangleq (\eta_x^{(l)}, \eta_h^{(l)})^T \sim \mathcal{N}(0, I) \end{cases} \tag{12}$$

Given $x^{(l-1)}$ and starting from $h^{(l-1)}$, we run one step (or more steps) of LD to obtain $h^{(l-1)*}$, which could be regarded as an approximate sample from $q_\phi(h|x^{(l-1)})$:

$$h^{(l-1)*} = h^{(l-1)} + \delta_l^* \frac{\partial}{\partial h^{(l-1)}} \log q_\phi(h^{(l-1)}, x^{(l-1)}) + \sqrt{2\delta_l^*}\eta_h^{(l)*}, \quad \eta_h^{(l)*} \sim \mathcal{N}(0, I) \tag{13}$$

Return (x, h).

So instead of using the exact gradient $\frac{\partial}{\partial z} \log p(z; \lambda)$ as shown in Eq. (7) in our case, we develop a tractable biased stochastic gradient $\Delta(z; \lambda)$ as follows:

$$\Delta(z; \lambda) \triangleq \begin{pmatrix} \frac{\partial}{\partial x}\left[\log p_\theta(x) + \log q_\phi(h, x) - \log q_\phi(h^*, x)\right] \\ \frac{\partial}{\partial h} \log q_\phi(h, x) \end{pmatrix}, \tag{14}$$

where h^* is an approximate sample from $q_\phi(h^*|x)$ obtained by one or more steps of LD from (h, x). Remarkably, as we show in Algorithm 2, the starting point $(h^{(0)}, x^{(0)})$ for the SGLD/SGHMC recursions is obtained from an ancestral sampling from $q_\phi(h, x)$. Thus at step $l = 1$, $h^{(0)}$ is already a sample from $q_\phi(h|x^{(0)})$ given $x^{(0)}$, and we can directly use $h^{(0)}$ as $h^{(0)*}$ without running the inner LD sampling. Afterwards, for $l > 1$, the conditional distribution of $h^{(l-1)}$ given $x^{(l-1)}$ is close to $q_\phi(h|x^{(l-1)})$, though strictly not. We could run one or more steps of LD to obtain $h^{(l-1)*}$ to reduce the bias in the stochastic gradient estimator.

With the above stochastic gradients in Eq. (14), the sampling step in Algorithm 1 can be performed by running $|\mathcal{M}|$ parallel chains, each chain being executed by running finite steps of SGLD/SGHMC with tractable gradients w.r.t. both x and h, as shown in Algorithm 2. Intuitively, the generator first gives a proposal (x', h'), and then the system follows the gradients of $p_\theta(x)$ and $q_\phi(h, x)$ (w.r.t. x and h respectively) to revise (x', h') to (x, h). The gradient terms pull samples moving to low energy region of the random field and adjust the latent code of the generator, while the noise term brings randomness. In this manner, we obtain Markov chain samples from $p_\theta(x)q_\phi(h|x)$.

To examine the sampling performance of the developed SGLD/SGHMC samplers, we conduct a synthetic experiment and the results are shown in Figure 2. The $p_\theta(x)$ and $q_\phi(x, h)$ are 50D and 100D Gaussians respectively with randomly generated covariance matrices (i.e. both x and h are of 50D). For evaluation, we simulate $K = 500$ parallel chains for $T = 2000$ steps. We follow [21] to evaluate the sampler's performance measured by the KL divergence from the empirical Gaussian (estimated by the samples) to the ground truth $p_\theta(x)q_\phi(h|x)$. We use the stepsize schedule of

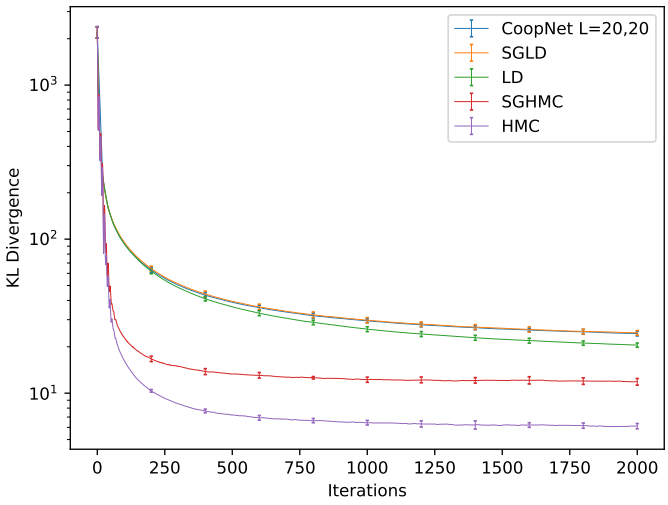


Fig. 2: Sampler's performance measured by the KL divergence with 10 independent runs to obtain standard deviations. "CoopNet $L = 20, 20$" denotes the sampling method in [9] with ($L_x = 20, L_h = 20$). "LD" or "HMC" means the Langevin Dynamics or Hamiltonian Monte Carlo sampling of the target distribution $p_\theta(x)q_\phi(h|x)$ with exact gradients. "SGLD" or "SGHMC" are our developed samplers with stochastic gradients (Algorithm 2). We fix the total iterations of x and h to be the same for each sampling method. Thus one iteration of "CoopNet $L = 20, 20$" would be regarded as 20 iterations of other methods in the figure.

$\delta_t = \left(a \cdot \left(1 + \frac{t}{b}\right)\right)^{-c}$ like in [21] with $(a = 10, b = 1000, c = 2)$ for all methods, and $\beta = 0.1$ for SGHMC, and we find that these hyperparameters perform well for each method during the experiment. The main observations are as follows. First, SGLD and SGHMC converge, though worse than their counterparts using exact gradients. Second, HMC samplers, whether using exact gradients or using stochastic gradients, outperform the corresponding LD samplers, since HMC dynamics, also referred to as second-order Langevin dynamics, exploit an additional momentum term. Third, interestingly, the SGHMC sampler outperforms the LD sampler with exact gradients. This reveals the benefit of our systematic development of the stochastic gradient samplers, including but not limited to SGLD and SGHMC. Although the CoopNet sampler in [9] (to be described in related work) performs close to our SGLD sampler, its performance is much worse than our SGHMC sampler. SGHMC is a new development, which cannot be obtained from simply extending the CoopNet sampler.

Finally note that, as discussed before, finite steps in Eq. (11)(12) and in Eq. (13) in applying SGLD/SGHMC sampling from $p_\theta(x)q_\phi(h|x)$ will produce biased estimates of the gradients (∇_θ and ∇_ϕ) in Eq. (6) for NRF learning. We did not find this to pose problems to the SGD optimization in practice, as similarly found in [22] and [13], which work with biased gradient estimators.

3.4 Semi-supervised learning with inclusive-NRFs

In the following, we apply our inclusive-NRF approach in the SSL setting to show its flexibility. Note that different models are needed in unsupervised and semi-supervised learning, because SSL needs to additionally consider labels apart from observations.

Model definition. In semi-supervised tasks, we consider the following RF for joint modeling of observation $x \in \mathbb{R}^{d_x}$ and class label $y \in \{1, \cdots, K\}$:

$$p_\theta(x, y) = \frac{1}{Z(\theta)} \exp\left[u_\theta(x, y)\right]. \tag{15}$$

This is different from Eq. (1) for unsupervised learning which only models x without labels. To implement the potential function $u_\theta(x, y)$, we consider a neural network $\Phi_\theta(x) : \mathbb{R}^{d_x} \to \mathbb{R}^K$, with x as the input and the output size being equal to the number of class labels, K. Then we define $u_\theta(x, y) = onehot(y)^T \Phi_\theta(x)$, where $onehot(y)$ represents the one-hot encoding vector for the label y. In this manner, the conditional density $p_\theta(y|x)$ is the classifier, defined as follows:

$$p_\theta(y|x) = \frac{p_\theta(x, y)}{p_\theta(x)} = \frac{\exp\left[u_\theta(x, y)\right]}{\sum_y \exp\left[u_\theta(x, y)\right]} \tag{16}$$

which acts like multi-class logistic regression using K logits calculated from x by the neural network $\Phi_\theta(x)$. And we do not need to calculate $Z(\theta)$ for classification. The auxiliary generator is implemented the same as in Eq. 4, i.e. an unconditional generator.

With the definition the joint density in Eq. 15, it can be shown that, with abuse of notation, the marginal density $p_\theta(x) = \frac{1}{Z(\theta)} \exp\left[u_\theta(x)\right]$ where $u_\theta(x) \triangleq log \sum_y \exp\left[u_\theta(x, y)\right]$.

Model learning. Suppose that among the data $\mathcal{D} = \{\tilde{x}_1, \cdots, \tilde{x}_n\}$, only a small subset of the observations, for example the first m observations, have class labels, $m \ll n$. Denote these labeled data as $\mathcal{L} = \{(\tilde{x}_1, \tilde{y}_1), \cdots, (\tilde{x}_m, \tilde{y}_m)\}$. Then we can formulate the semi-supervised learning as jointly optimizing

$$\begin{cases} \min\limits_\theta KL\left[\tilde{p}(\tilde{x})||p_\theta(\tilde{x})\right] - \alpha_d \sum\limits_{(\tilde{x},\tilde{y})\sim\mathcal{L}} log p_\theta(\tilde{y}|\tilde{x}) \\ \min\limits_\phi KL\left[p_\theta(x)||q_\phi(x)\right] \end{cases} \tag{17}$$

which are defined by hybrids of generative and discriminative criteria, similar to [23], [24], [25]. The hyper-parameter α_d controls the relative weight between generative and discriminative criteria. Similar to deriving Eq. (6), it can be easily seen that the gradients w.r.t. θ and ϕ (to be ascended) are defined as follows:

$$\begin{cases} \nabla_\theta^{\text{semi}} = & E_{\tilde{p}(\tilde{x})}\left[\nabla_\theta log p_\theta(\tilde{x})\right] \\ & + \alpha_d \sum\limits_{(\tilde{x},\tilde{y})\sim\mathcal{L}} \nabla_\theta log p_\theta(\tilde{y}|\tilde{x}) \\ & = E_{\tilde{p}(\tilde{x})}\left[\nabla_\theta u_\theta(\tilde{x})\right] - E_{p_\theta(x)}\left[\nabla_\theta u_\theta(x)\right] \\ & + \alpha_d \sum\limits_{(\tilde{x},\tilde{y})\sim\mathcal{L}} \nabla_\theta log p_\theta(\tilde{y}|\tilde{x}) \\ \nabla_\phi^{\text{semi}} = & E_{p_\theta(x)}\left[\nabla_\phi log q_\phi(x)\right] \\ & = E_{p_\theta(x)q_\phi(h|x)}\left[\nabla_\phi log q_\phi(x, h)\right] \end{cases} \tag{18}$$

In practice, we calculate noisy gradient estimators, and apply minibatch based stochastic gradient descent (SGD) to solve the optimization problem Eq.(17), as shown in Algorithm 3 in Appendix A. Apart from the basic losses as shown in Eq. (17), there are some regularization losses that are found to be helpful to guide SSL learning and are presented in Appendix B.

To conclude, we show that the inclusive-NRF can be easily applied to SSL. To the best of our knowledge, there are no priori studies in applying random fields to SSL. **The semi-supervised inclusive-NRF model defined above is novel itself for SSL.**

4 Related work

Comparison and connection of our inclusive-NRF approach with existing studies are provided in the following from three perspectives.

Learning NRFs. These studies are most relevant to this work, which aims to learn NRFs. The classic method for learning RFs is the SML method [12], which works with the single target model p_θ. Compared to learning traditional RFs which mainly use linear potential functions, learning NRFs which use NN based nonlinear potential functions, is more challenging. A recent progress in learning NRFs as studied in [8], [9], [11], [13] is to jointly train the target random field $p_\theta(x)$ and an auxiliary generator $q_\phi(x)$. Different studies differ in the objective functions used in the joint training, and thus have different computational and statistical properties.

(1) It is shown in Proposition 3 in Appendix C that learning in [8] minimizes the exclusive-divergence $KL[q_\phi||p_\theta]$ w.r.t. ϕ, which involves the intractable entropy term $H\left[q_\phi\right]$ and tends to enforce the generator to seek modes, yielding missing modes. We refer to this approach as exclusive-NRF.

(2) Learning in [11] and in this paper minimizes the inclusive-divergence $KL[p_\theta||q_\phi]$ w.r.t. ϕ. But noticeably, this paper presents our innovation in development of NRFs for continuous data, which is fundamentally different from [11] for discrete data. The target NRF model, the generator and the sampler all require new designs. [11] mainly studies random field language models, using LSTM generators (autoregressive with no latent variables) and employing Metropolis independence sampler (MIS) - applicable for discrete data (natural sentences). In this paper, we design random field models for continuous data (e.g. images), choosing latent-variable generators and developing SGLD/SGHMC to exploit noisy gradients in the continuous space.

(3) In [9] (CoopNet), motivated by interweaving maximum likelihood training of the random field p_θ and the latent-variable generator q_ϕ, a joint training method is introduced to train NRFs. Our inclusive-NRF approach is different from [9] in two key aspects. First, this method uses LD sampling to generate samples, but two LD sampling steps are intuitively interleaved according to $\frac{\partial}{\partial x}\log p_\theta(x)$ and $\frac{\partial}{\partial h}\log q_\phi(h,x)$ separately, without aiming to draw samples from $p_\theta(x)q_\phi(h|x)$ and without awareness of stochastic gradients. Specifically, starting from the ancestral sample $(x^{(0)},h^{(0)}) \sim q_\phi(h,x)$, a first LD sampling with L_x steps are conducted to obtain $x^{(1)},\cdots,x^{(L_x)}$ according to $\frac{\partial}{\partial x^{(l)}}\log p_\theta(x^{(l)})$, and then a second LD sampling with L_h steps are used to obtain $h^{(1)},\cdots,h^{(L_h)}$ according to $\frac{\partial}{\partial h^{(l)}}\log q_\phi(h^{(l)},x^{(L_x)})$ with fixed $x^{(L_x)}$. Algorithmically, this is different from our sampling step, which moves (x,h) jointly, as systematically developed in Section 3.3. Specifically, starting from $(x^{(0)},h^{(0)})$, our SGLD recursions as shown in Eq. (11) are conducted to obtain $(x^{(1)},h^{(1)}),\cdots,(x^{(L)},h^{(L)})$. Notably, SGHMC is a new development, which cannot be obtained from simply extending the CoopNet sampler. Moreover, our development allows easy incorporation of more advanced elements in stochastic gradient samplers, such as utilizing the Riemannian geometry of the target distribution via preconditioning [21].

Second, let $r(h,x)$ denote the distribution of $(x^{(L_x)},h^{(L_h)})$, resulting from the interleaved Langevin transitions. Interpretation presented in [9] relates their method to the following joint optimization problem:

$$\begin{cases} \min\limits_{\theta} \{KL\left[\tilde{p}(\tilde{x})||p_\theta(\tilde{x})\right] - KL\left[r(h,x)||p_\theta(x)\right]\} \\ \min\limits_{\phi} KL\left[r(h,x)||q_\phi(h,x)\right] \end{cases}$$

which is also different from our learning objectives as shown in Eq. (5). Thus, learning in [9] does not aim to minimize the inclusive-divergence $KL[p_\theta||q_\phi]$ w.r.t. ϕ.

Empirically, as shown in Figure 2, the CoopNet sampler performs much worse than our SGHMC sampler. It is further shown in Table 2 that inclusive-NRF with SGLD outperforms CoopNet in image generation, and in Table 4 that utilizing SGHMC in learning inclusive-NRFs to exploit gradient information with momentum yields better performance than using SGLD.

(4) Learning in [13] minimizes the χ^2-divergence $\chi^2[q_\phi||p_\theta] \triangleq \int \frac{(p_\theta - q_\phi)^2}{q_\phi}$ w.r.t. ϕ, which also tends to drive the generator to cover modes. But this approach is severely limited by the high variance of the gradient estimator w.r.t. ϕ, and is only tested on the simpler MNIST and Omniglot.

Additionally, different NRF studies also differ in models used in the joint training. The target NRF used in this work is different from those in previous studies [8], [11], [9]. The differences are: [8] includes additional linear and squared terms in $u_\theta(x)$, [11] defines over discrete-valued sequences, and [9] defines in the form of exponential tilting of a reference distribution (Gaussian white noise). There also exist different choices for the generator, such as GAN models in [8], LSTMs in [11], or latent-variable models in [9] and this work.

To sum up, our contributions in introducing inclusive-divergence minimized auxiliary generators and developing stochastic gradient sampling enables the solid development of the new inclusive-NRF approach. Moreover, all the previous NRF studies examine unsupervised learning, and none shows application or extension of their methods or models for semi-supervised learning.

Monte Carlo sampling. One step in our inclusive-NRF approach is to apply SGLD/SGHMC to draw samples from the target density p_θ, starting from the proposal sample from the generator. Theoretically, improvements in NRF sampling methods could be potentially integrated into NRF learning algorithms. For example, it is recently studied in [26] to learn MCMC transition kernels, also parameterized by neural networks, to improve the HMC sampling from the given target distribution. Integration into learning NRFs is interesting but outside the scope of this paper.

Comparison and connection with GANs. On the one hand, there are some efforts that aim to address the inability of GANs to provide sensible energy estimates for samples. The energy-based GANs (EBGAN) [27] proposes to view the discriminator as an energy function by designing an auto-encoder discriminator. The recent work in [28] connects [27] and [8], and show another two approximations for the entropy term. However, it is known that as the generator converges to the true data distribution, the GAN discriminator converges to a degenerate uniform solution. This basically afflicts the GAN discriminator to provide density information, though there are some modifications. In contrast, our inclusive-NRFs, unlike GANs, naturally provide (unnormalized) density estimate, which is examined with GMM synthetic experiments and anomaly detection benchmarking experiments. Moreover, none of the above energy-related GAN studies examine their methods or models for SSL, except in EBGAN which performs moderately.

On the other hand, there are interesting connections between inclusive-NRFs and GANs, as elaborated in Appendix D. When interpreting the potential function $u_\theta(x)$ as the critic in Wasserstein GANs [29], inclusive-NRFs seem to be similar to Wasserstein GANs. A difference is that in optimizing θ in inclusive-NRFs, the generated samples are further revised by taking finite-step-gradient of $u_\theta(x)$ w.r.t. x. However, the critic in Wasserstein GANs can hardly be interpreted as an unnormalized log-density. Thus strictly speaking, inclusive-NRFs are not GAN-like.

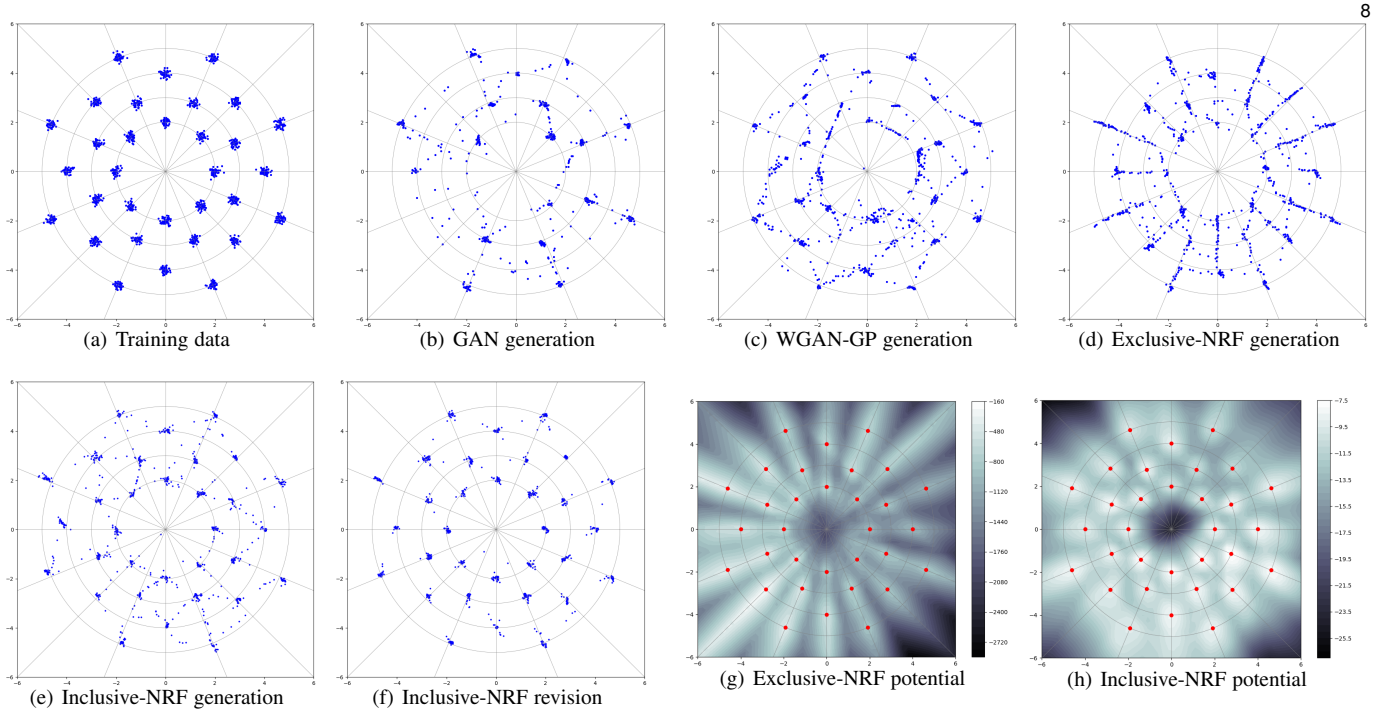
5 Experiments

As demonstrations of how the new inclusive-NRF approach can be flexibly and effectively used, specific inclusive-NRF models are developed and thoroughly evaluated for a number of tasks - unsupervised/supervised image generation, semi-supervised classification and anomaly detection.

First, we report experiments on synthetic datasets, which helps to illustrate different models and learning methods. Then, extensive experiments are conducted to evaluate the performances of our approach (inclusive-NRFs) and various existing methods on real-world datasets. We refer to Appendix E for experimental details and additional results.

5.1 GMM synthetic experiment for unsupervised learning

The synthetic data consist of 1,600 training examples generated from a 2D Gaussian mixture model (GMM) with 32 equally-weighted, low-variance ($\sigma = 0.1$) Gaussian components, uniformly laid out on four concentric circles as in Figure 3(a). The data distribution exhibits many modes separated by large low-probability regions, which makes it suitable to examine how well different learning methods can deal with multiple modes. For comparison, we experiment with GAN with logD trick [4] and WGAN-GP [30] for directed generative model, exclusive-NRF [8] and inclusive-NRF for undirected generative model.



(a) Training data (b) GAN generation (c) WGAN-GP generation (d) Exclusive-NRF generation

(e) Inclusive-NRF generation (f) Inclusive-NRF revision (g) Exclusive-NRF potential (h) Inclusive-NRF potential

Fig. 3: Comparison of different methods over GMM synthetic data. Stochastic generations from GAN with logD trick, WGAN-GP, Exclusive-NRF, **Inclusive-NRF generation** (i.e. sampling from the auxiliary generator) and **Inclusive-NRF revision** (i.e. after sample revision), are shown in (b)-(f) respectively. Each generation contains 1,000 samples. The learned potentials $u_\theta(x)$ from exclusive and inclusive NRFs are shown in (g) and (h) respectively, where the red dots indicate the mean of each Gaussian component. Inclusive NRFs are clearly superior in learning data density and sample generation.

TABLE 1: Numerical evaluations over the GMM (32 components) synthetic data. The "**covered modes**" metric is defined as the number of covered modes by a set of generated samples. The "**realistic ratio**" metric is defined as the proportion of generated samples which are close to a mode. The measurement details are presented in text. Mean and SD are from 10 independent runs.

Methods	covered modes	realistic ratio
GAN with logD trick [4]	22.25 ± 1.54	0.90 ± 0.01
WGAN-GP [30]	27.81 ± 1.40	0.74 ± 0.04
Exclusive-NRF [8]	28.14 ± 0.68	0.73 ± 0.03
Inclusive-NRF generation	29.52 ± 0.54	0.84 ± 0.01
Inclusive-NRF revision	30.75 ± 0.43	0.97 ± 0.01

The network architectures and hyperparameters are the same for all methods, as listed in Table 7 in Appendix. We use SGLD [14] for inclusive-NRFs on this synthetic dataset, with empirical revision hyperparameters $\delta_l = 0.01$

Figure 3 visually shows the generated samples from the trained models using different methods. Table 1 reports the "covered modes" and "realistic ratio" as numerical measures of how the multi-modal data are fitted, similarly as in [31]. We use the following procedure to estimate the metrics "covered modes" and "realistic ratio" for each trained model.

1) Stochastically generate 100 samples.
2) A mode is defined to be covered (not missed) if there exist generated samples located closely to the mode (with squared distance < 0.02), and those samples are said to be realistic.
3) Count how many modes are covered and calculate the proportion of realistic samples.
4) Repeat the above steps 100 times and perform averaging.

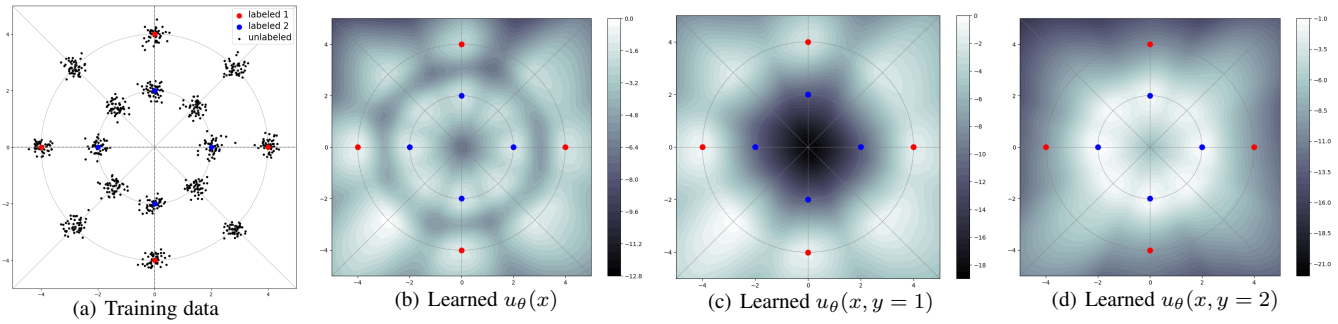
For each method, we independently train 10 models and calculate the mean and standard deviation (SD) across the 10 independent runs. The main observations are as follows:

- GAN suffers from mode missing, generating realistic but not diverse samples. WGAN-GP increases "covered modes" but decreases "realistic ratio". Inclusive-NRF performs much better than both GAN and WGAN-GP in sample generation.
- Inclusive-NRF outperforms exclusive-NRF in both sample generation and density estimation.
- After revision, samples from inclusive-NRF become more like real samples, achieving the best in both "covered modes" and "realistic ratio" metrics.

5.2 GMM synthetic experiment for semi-supervised learning

In this experiment, we present the performance of semi-supervised inclusive-NRFs for SSL on a synthetic dataset. In addition to illustrating how semi-supervised inclusive-NRF works, this experiment further emphasizes that the inclusive-NRF approach can provide (unnormalized) density estimates for $p_\theta(x)$, $p_\theta(x, y = 1)$ and $p_\theta(x, y = 2)$. In contrast, the use of GANs as general purpose probabilistic generative models has been limited by the difficulty in using them to provide density estimates or even unnormalized potential values for sample evaluation.

The dataset is a 2D GMM with 16 Gaussian components, uniformly laid out on two concentric circles. The two circles represent two different classes. There are only 4 labeled points



(a) Training data (b) Learned $u_\theta(x)$ (c) Learned $u_\theta(x, y = 1)$ (d) Learned $u_\theta(x, y = 2)$

Fig. 4: SSL toy experiment based on semi-supervised inclusive-NRFs. Each class has 4 labeled points, red dots for class 1 and blue for class 2. The learned potentials for $u_\theta(x)$, $u_\theta(x, y = 1)$ and $u_\theta(x, y = 2)$ are shown in (b)(c)(d) respectively.

TABLE 2: Inception score (IS) and FID on CIFAR-10 for unsupervised and supervised learning. "-" means the results are not reported in the original work.

Methods	Unsupervised		Supervised	
	IS	FID	IS	FID
DCGAN [35]	6.16 ± 0.07	-	6.58	-
Improved-GAN [33]	-	-	8.09 ± 0.07	-
WGAN-GP [30]	7.86 ± 0.07	-	8.42 ± 0.10	-
SGAN [36]	-	-	8.59 ± 0.12	-
DFM [37]	7.72 ± 0.13	-	-	-
CT-GAN [38]	8.12 ± 0.12	-	8.81 ± 0.13	-
Fisher-GAN [39]	7.90 ± 0.05	-	8.16 ± 0.12	-
CoopNet [9]	-	33.61	-	-
BWGAN [40]	8.26 ± 0.07	-	-	-
SNGAN [41]	8.22 ± 0.05	21.7 ± 0.21	-	-
Inclusive-NRF generation	8.28 ± 0.09	20.9 ± 0.25	9.06 ± 0.10	18.1 ± 0.23

for each class and a total of 400 unlabeled points. The network architectures are the same as in Section 5.1, except that the neural network which implement the potential function $u_\theta(x, y)$ for SSL now has two units in the output. As shown in Figure 4, the semi-supervised trained inclusive-NRFs not only captures the marginal potential effectively, but also learn the class-conditional potentials successfully. This behavior agrees with our design idea of blending unsupervised and supervised learning for the semi-supervised setting as shown in Eq. (18).

5.3 Image generation on CIFAR-10

In this experiment, we examine both unsupervised and supervised learning over the widely used real-world dataset CIFAR-10 [32] for image generation. To evaluate generation quality quantitatively, we use inception score (IS) [33] (the larger the better), and Frechet inception distance (FID) [34] (the smaller the better). Table 2 reports the inception score and FID for state of the art methods, for both unsupervised and supervised settings. The supervised learning of inclusive-NRF is conducted as a special case of semi-supervised learning over all labeled images ($m = n$), which uses unconditional generation. We use ResNet in this experiment, see Appendix E.1 for experimental details.

From the comparison results in Table 2, it can be seen that the proposed inclusive-NRF model achieves the best inception score over CIFAR-10, to the best of our knowledge, in both unsupervised and supervised settings. Some generated samples are shown in Figure 7(c)(d) in Appendix for unsupervised and supervised settings respectively. We also show the capability of inclusive-NRFs in latent space interpolation (Appendix F) and conditional generation (Appendix G).

5.4 Semi-supervised learning on MNIST, SVHN and CIFAR-10

For semi-supervised learning, we consider the three widely used benchmark datasets, namely MNIST [52], SVHN [53], and CIFAR-10 [32]. As in previous work, we randomly sample 100, 1,000, and 4,000 labeled samples from MNIST, SVHN, and CIFAR-10 respectively during training, and use the standard data split for testing. See Appendix E.2 for experimental details.

It can be seen from Table 3 that semi-supervised inclusive-NRFs produce strong classification results on par with state-of-art DGM-based SSL methods. See Figure 7(a)(b) in Appendix for generated samples. Bad-GANs achieve better classification results, but as indicated by the low inception score, their generation is much worse than semi-supervised inclusive-NRFs. In fact, among DGM-based SSL methods, inclusive-NRFs achieve the best performance in sample generation. **This is in contrast to the conflict of good classification and good generation, as observed in GAN-based SSL** [33], [47]. It is analyzed in [47] that good GAN-based SSL requires a bad generator[5]. This is embarrassing and in fact obviates the original idea of generative SSL - successful generative training, which indicates good generation, provides regularization for finding good classifiers [23], [24]. In this sense, Bad-GANs could hardly be classified as a generative SSL method.

Finally, note that some discriminative SSL methods, as listed in the lower block in Table 3 also produce superior performances, by utilizing data augmentation and consistency regularization.

5. This analysis is based on using the $(K + 1)$-class GAN-like discriminator objective for SSL. To the best of our knowledge, the conflict does not seem to be reported in previous generative SSL methods [23], [24] which use the K-class classifier like in semi-supervised inclusive-NRFs.

TABLE 3: Comparison with state-of-the-art methods on three benchmark datasets. "CIFAR-10 IS" means the inception score for samples generated by SSL models trained on CIFAR-10. "†" is obtained by running the released code accompanied by the corresponding papers. "-" means the results are not reported in the original work and without released code. "/" means not applicable, e.g. the models cannot generate samples stochastically. "‡" uses image data augmentation which significantly helps classification performance. The upper/lower blocks show generative/discriminative SSL methods respectively.

Methods	error (%) MNIST	error (%) SVHN	error (%) CIFAR-10	IS CIFAR-10
CatGAN [42]	1.91 ± 0.10	-	19.58 ± 0.46	$3.57 \pm 0.13^{\dagger}$
SDGM [43]	1.32 ± 0.07	16.61 ± 0.24	-	-
Ladder network [44]	1.06 ± 0.37	-	20.40 ± 0.47	/
ADGM [43]	0.96 ± 0.02	22.86	-	-
Improved-GAN [33]	0.93 ± 0.07	8.11 ± 1.3	18.63 ± 2.32	3.87 ± 0.03
EBGAN [27]	1.04 ± 0.12	-	-	-
ALI [31]	-	7.42 ± 0.65	17.99 ± 1.62	-
Triple-GAN [45]	0.91 ± 0.58	5.77 ± 0.17	16.99 ± 0.36	5.08 ± 0.09
Triangle-GAN [46]	-	-	16.80 ± 0.42	-
BadGAN [47]	0.80 ± 0.10	4.25 ± 0.03	14.41 ± 0.30	$3.46 \pm 0.11^{\dagger}$
Sobolev-GAN [48]	-	-	15.77 ± 0.19	-
Semi-supervised inclusive-NRF	0.97 ± 0.10	5.84 ± 0.15	15.12 ± 0.36	7.72 ± 0.09
Results below this line cannot be directly compared to those above.				
VAT small [49]	1.36	6.83	14.87	/
Π model‡ [50]	-	4.82 ± 0.17	12.36 ± 0.31	/
Temporal Ensembling‡ [50]	-	4.42 ± 0.16	12.16 ± 0.31	/
Mean Teacher‡ [51]	-	3.95 ± 0.19	12.31 ± 0.28	/
VAT+EntMin‡ [49]	-	3.86	10.55	/
CT-GAN‡ [38]	0.89 ± 0.13	-	9.98 ± 0.21	/

TABLE 4: Ablation study of our inclusive-NRF method on CIFAR-10, regarding the effects of using SGLD or SGHMC in training and of applying sample revision in inference (generating samples). Mean and SD are from 5 independent runs for each training setting. In each training setting, for unsupervised learning, two manners to generate samples given a trained NRF are compared, as previously illustrated in Figure 3 over synthetic GMM data. We examine generated samples (i.e. directly from the generator) and revised samples (i.e. after sample revision) respectively, in term of inception scores (IS). For semi-supervised learning, we examine the classification error rates.

Training Setting	Unsupervised		Semi-supervised
	Generation IS	Revision IS	error (%)
SGLD $L = 1$	7.47 ± 0.15	7.53 ± 0.13	17.08 ± 0.39
SGLD $L = 5$	7.44 ± 0.16	7.49 ± 0.12	16.15 ± 0.44
SGLD $L = 10$	7.43 ± 0.18	7.50 ± 0.13	15.60 ± 0.31
SGHMC $L = 10$	7.46 ± 0.12	7.57 ± 0.10	15.12 ± 0.36

However, these methods are unable to generate (realistic) samples. It can be seen that discriminative SSL methods utilize different regularization from generative SSL methods and cannot be directly compared to generative SSL methods. Their combination, as an interesting future work, could yield further performance improvement.

5.5 Ablation study

We report the results of ablation study of our inclusive-NRF method on CIFAR-10 in Table 4. In this experiment, we use the standard CNN [41] for unsupervised learning and the same networks as those used in Table 3 for semi-supervised learning. See Appendix E.3 for experimental details. We analyze the effects of different settings in model training, such as using SGLD or SGHMC and the revision step $L = 1/5/10$ used. For each training setting, we also compare the two manners to generate samples - whether applying sample revision or not in inference (generating samples) given a trained NRF, as previously illustrated in Figure 3 over synthetic GMM data. The main observations are as follows.

First, given a trained NRF, after revision (i.e. following the gradient of the RF's potential $u_\theta(x)$ w.r.t. x), the quality (IS) of samples is always improved, as shown by the consistent IS improvement from the second column (generation) to the third (revision). This is in accordance with the results in the GMM synthetic experiments in Section 5.1. Moreover, noting that in revision, it is the the estimated density p_θ that guides the samples towards low energy region of the random field. **This also demonstrates one benefit of random field modeling, which, unlike GANs, can learn density estimate about the data manifold.**

Second, a row-wise reading of Table 4 reveals that with more revision steps and using SGHMC in training, the SSL classification performance is improved. Utilizing SGHMC in inclusive-NRFs to exploit gradient information with momentum yields better performance than simple SGLD. It is also found that more revision steps in model training do not significantly improve unsupervised IS. So we can use $L = 1$ in unsupervised learning for generation, which can reduce the computational cost.

5.6 Anomaly detection

One fundamental additional benefit of our inclusive-NRF approach is that, unlike GANs, it directly provides (unnormalized) density estimate, apart from superior performances in image generation and classification as already shown in previous sections. To evaluate the performance in density estimate, anomaly detection is a good real-world benchmarking task, in addition to the GMM synthetic experiments (Section 5.1 and 5.2). Anomaly detection (also known as one-class classification [54]) is a fundamental problem in machine learning, with critical applications in many areas, such as

TABLE 5: Anomaly detection results on KDDCUP dataset. Results for OC-SVM, DSEBM-e, DAGMM are taken from [58]. Inclusive-NRF results are obtained from 20 runs, each with random split of training and test sets, as in [58]. ALAD result uses a fixed split with random parameter initializations, and thus has smaller standard deviations.

Model	Precision	Recall	F1
OC-SVM [54]	0.7457	0.8523	0.7954
DSEBM-e [57]	0.7369	0.7477	0.7423
DAGMM [58]	0.9297	0.9442	0.9369
ALAD [59]	0.9427±0.0018	0.9577±0.0018	0.9501±0.0018
Inclusive-NRF	0.9452±0.0105	0.9600±0.0113	0.9525±0.0108

TABLE 6: Anomaly detection results on MNIST and CIFAR-10 measured by AUCs (%). Both datasets have ten different classes from which we create ten one-class classification setups. Results for each one-class setup are obtained from 10 runs (with random parameter initializations), and averaging the mean AUCs over the ten setups gives the "mean" result. Results for OC-SVM, KDE (Kernel density estimation), IF (Isolation Forest), DCAE (Deep Convolutional AutoEncoders), AnoGAN, DSVDD are taken from [56].

Normal Class	OC-SVM	KDE	IF	DCAE	AnoGAN	DSVDD	Inclusive-NRF
Digit 0	98.6±0.0	97.1±0.0	98.0±0.3	97.6±0.7	96.6±1.3	98.0±0.7	**98.9±0.6**
Digit 1	99.5±0.0	98.9±0.0	97.3±0.4	98.3±0.6	99.2±0.6	99.7±0.1	**99.8±0.1**
Digit 2	82.5±0.1	79.0±0.0	88.6±0.5	85.4±2.4	85.0±2.9	91.7±0.8	**91.8±4.0**
Digit 3	88.1±0.0	86.2±0.0	89.9±0.4	86.7±0.9	88.7±2.1	91.9±1.5	**93.8±2.6**
Digit 4	94.9±0.0	87.9±0.0	92.7±0.6	86.5±2.0	89.4±1.3	94.9±0.8	**95.6±1.9**
Digit 5	77.1±0.0	73.8±0.0	85.5±0.8	78.2±2.7	88.3±2.9	88.5±0.9	**94.9±1.4**
Digit 6	96.5±0.0	87.6±0.0	95.6±0.3	94.6±0.5	94.7±2.7	**98.3±0.5**	97.5±2.7
Digit 7	93.7±0.0	91.4±0.0	92.0±0.4	92.3±1.0	93.5±1.8	94.6±0.9	**96.4±1.0**
Digit 8	88.9±0.0	79.2±0.0	89.9±0.4	86.5±1.6	84.9±2.1	**93.9±1.6**	88.9±3.3
Digit 9	93.1±0.0	88.2±0.0	93.5±0.3	90.4±1.8	92.4±1.1	**96.5±0.3**	94.9±1.0
Mean	91.29	86.93	92.30	89.65	91.27	94.80	**95.26**
AIRPLANE	61.6±0.9	61.2±0.0	60.1±0.7	59.1±5.1	67.1±2.5	61.7±4.1	**78.1±2.1**
AUTOMOBILE	63.8±0.6	64.0±0.0	50.8±0.6	57.4±2.9	54.7±3.4	65.9±2.1	**71.6±2.1**
BIRD	50.0±0.5	50.1±0.0	49.2±0.4	48.9±2.4	52.9±3.0	50.8±0.8	**65.4±1.8**
CAT	55.9±1.3	56.4±0.0	55.1±0.4	58.4±1.2	54.5±1.9	59.1±1.4	**63.3±1.9**
DEER	66.0±0.7	66.2±0.0	49.8±0.4	54.0±1.3	65.1±3.2	60.9±1.1	**70.5±2.2**
DOG	62.4±0.8	62.4±0.0	58.5±0.4	62.2±1.8	60.3±2.6	**65.7±2.5**	64.1±2.7
FROG	74.7±0.3	74.9±0.0	42.9±0.6	51.2±5.2	58.5±1.4	67.7±2.6	**75.4±2.4**
HORSE	62.6±0.6	62.6±0.0	55.1±0.7	58.6±2.9	62.5±0.8	**67.3±0.9**	66.1±3.7
SHIP	74.9±0.4	75.1±0.0	74.2±0.6	**76.8±1.4**	75.8±4.1	75.9±1.2	75.6±2.8
TRUCK	75.9±0.3	**76.0±0.0**	58.9±0.7	67.3±3.0	66.5±2.8	73.1±1.2	70.1±2.0
Mean	64.78	64.89	55.46	59.39	61.79	64.81	**70.02**

cybersecurity, complex system management, medical care, and so on. At the core of anomaly detection is density estimation: given a lot of input samples, anomalies are those ones residing in low probability density areas.

Anomaly detection has been extensively studied, as reviewed in recent works [54], [55], [56], [57], [58], [59]. Classical anomaly detection methods are kernel-based, e.g. One-Class Support Vecter Machine (OC-SVM) [54] and Support Vector Data Description (SVDD) [55]. Such shallow methods typically require substantial feature engineering and also limited by bad computational scalability. Recent methods leverage feature learning by using deep neural networks. Deep SVDD (DSVDD) [56] combines a deep neural network with kernel-based SVDD. In Deep Structured Energy-based Model (DSEBM) (DSEBM) [57], the models are essentially NRFs, but the training method is score matching [60]. Deep Autoencoding Gaussian Mixture Model (DAGMM) [58] jointly train a deep autoencoder (which generates low-dimensional features) and a GMM (which operates on those low-dimensional features). Adversarially Learned Anomaly Detection (ALAD) [59] uses a bi-directional GAN which needs one more network (the inference network) in addition to the generator and the discriminator networks. In practice, the detection is usually performed by thresholding reconstruction errors (as used in DSEBM, ALAD) or density estimates (as used in DSEBM, DAGMM). Both criteria are tested for DSEBM, denoted by DSEBM-r (reconstruction) and DSEBM-e (energy). It is found that the energy score is a more accurate decision criterion than the reconstruction error [57].

In applying inclusive-NRFs to anomaly detection, the unnormalized density estimates (as measured by potential values) provide a natural decision criterion for anomaly detection, since the normalizing constant only introduces a constant in thresholding. Specifically, after training inclusive-NRFs on data containing only the samples of the normal class, testing samples with potential values lower than a threshold are detected as anomaly. We evaluated our inclusive-NRFs for anomaly detection on publicly available tabular and image datasets - KDDCUP, MNIST and CIFAR-10.

See Appendix E.4 for experimental details.

KDDCUP. For tabular data, we test on the KDDCUP99 ten percent dataset [61] (denoted as KDDCUP) and follow the standard setup in [58]. This dataset is a network intrusion dataset, originally contains samples of 41 dimensions, where 34 of them are continuous and 7 are categorical. One-hot representation is used to encode categorical features, and eventually a dataset of 120 dimensions is obtained. As 20% of data samples are labeled as "normal" and the rest are labeled as "attack", "normal" ones are thus treated as anomalies in this task. For each run, we randomly take 50% of the whole dataset and use only data samples from the normal class for training models; the rest 50% is reserved for testing. During testing, the 20% samples with lowest potentials will be marked as anomalies. The anomaly class is regarded as positive, and precision, recall, and F1 score are calculated accordingly.

From the results shown in Table 5, it can be seen that inclusive-NRF outperforms all state-of-the-art methods (DSEBM-e, DAGMM, ALAD). Particularly, compared to the previous state-of-the-art deep energy model (DSEBM-e), inclusive-NRF outperforms by a large margin (more than +0.2 F1 score), which clearly shows the superiority of our new approach in learning NRFs.

MNIST and CIFAR-10. Both datasets have ten different classes from which we create ten one-class classification setups. The standard training and test splits of MNIST and CIFAR-10 are used, and we follow the "one-class" setup in [56]. For each setup, one of the classes is the normal class and samples from the remaining classes are treated as anomalies; only training samples from the normal class are employed for model training. Accordingly, for each setup, training set sizes are 6000 for MNIST and 5000 for CIFAR-10, and the test set consists of 1000 normal samples and 9000 abnormal samples. For comparison with existing results, AUC (area under the receiver operating curve) [62] is employed as performance metric in both datasets. From the results shown in Table 6, it can be seen that inclusive-NRF performs much better state-of-the-art method (DSVDD) which is specifically designed for anomaly detection.

To sum up, here we show that a straightforward application of the inclusive-NRF approach in anomaly detection on both tabular and image datasets achieves superior performance over state-of-the-art methods. This is a clear indication of the ability of inclusive-NRFs for density estimation.

6 Conclusion and Future Directions

Neural random fields (NRFs), referring to the class of random fields that use neural networks to implement potential functions, received less attention with slow progress, but have their own merits as demonstrated in this paper. In this paper we propose a new approach, the inclusive-NRF approach, to learning NRFs for continuous data (e.g. images), with inclusive-divergence minimized auxiliary generator and stochastic gradient sampling. Our contributions in introducing inclusive-divergence minimized auxiliary generators and developing stochastic gradient sampling enables the solid development of the new inclusive-NRF approach.

With the new approach, specific inclusive-NRF models are developed and thoroughly evaluated for a number of tasks - unsupervised/supervised image generation, semi-supervised classification and anomaly detection/one-class classification. The proposed models consistently achieve strong experimental results in all these tasks compared to state-of-the-art methods. These superior performances presumably are attributed to **the two distinctive features in inclusive-NRFs - successfully introducing the inclusive-divergence minimized auxiliary generator and performing model sampling by SGLD/SGHMC**. Intuitively, the revised samples from the RF will guide the training of the generator, and subsequently the generator will propose samples for the RF to sense the data manifold. This forms positive interactions between the random field and the generator, which enables successful joint training of both models.

There are various worthwhile directions for future research. First, the flexibility of the inclusive-NRF approach is worth emphasizing. The new approach enables us to flexibly use NRFs in unsupervised, supervised and semi-supervised settings. We anticipate the application of the inclusive-NRF approach to more machine learning applications. Second, although this work deals with fix-dimensional data, it is an important direction of extending the inclusive-NRF approach to sequential and trans-dimensional modeling tasks (e.g. speech, language, video, etc.). Third, the conditional distribution over labels given the observations in CRFs can be described by NRFs so that global interactions among labels can be utilized. A conditional version of the inclusive-NRF approach could be developed to train such CRFs. Finally, to facilitate future studies, we will release our code and scripts for reproducing the results in this paper.

7 Acknowledgments

This work was supported by NSFC Grant 61473168, China MOE-Mobile Grant MCM20170301. The authors would like to thank Zhiqiang Tan for helpful discussions.

References

[1] B. J. Frey and N. Jojic, "A comparison of algorithms for inference and learning in probabilistic graphical models," *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, vol. 27, no. 9, pp. 1392–1416, 2005.

[2] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[3] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *ICLR*, 2014.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[5] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, "A tutorial on energy-based learning," *Predicting structured data*, 2006.

[6] M. J. Wainwright, M. I. Jordan *et al.*, "Graphical models, exponential families, and variational inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2008.

[7] J. Ngiam, Z. Chen, W. K. Pang, and A. Y. Ng, "Learning deep energy models," in *ICML*, 2012.

[8] T. Kim and Y. Bengio, "Deep directed generative models with energy-based probability estimation," in *ICLR Workshop*, 2016.

[9] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu, "Cooperative training of descriptor and generator networks," *arXiv preprint arXiv:1609.09408*, 2016.

[10] J. Dai, Y. Lu, and Y.-N. Wu, "Generative modeling of convolutional neural networks," *arXiv preprint arXiv:1412.6296*, 2014.

[11] B. Wang and Z. Ou, "Language modeling with neural trans-dimensional random fields," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2017.

[12] L. Younes, "Parametric inference for imperfectly observed gibbsian fields," *Probability Theory and Related Fields*, vol. 82, pp. 625–645, 1989.

[13] V. Kuleshov and S. Ermon, "Neural variational inference and learning in undirected graphical models," in *NIPS*, 2017.

[14] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient Langevin dynamics," in *ICML*, 2011.

[15] T. Chen, E. Fox, and C. Guestrin, "Stochastic gradient Hamiltonian Monte Carlo," in *ICML*, 2014.

[16] B. Wang, Z. Ou, and Z. Tan, "Learning trans-dimensional random fields with applications to language modeling," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 876–890, 2018.

[17] C. Wang, N. Komodakis, and N. Paragios, "Markov random field modeling, inference & learning in computer vision & image understanding: A survey," *Computer Vision and Image Understanding*, vol. 117, no. 11, pp. 1610–1627, 2013.

[18] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal, "The "wake-sleep" algorithm for unsupervised neural networks." *Science*, vol. 268, no. 5214, pp. 1158–1161, 1995.

[19] R. M. Neal, "MCMC using Hamiltonian dynamics," *Handbook of Markov Chain Monte Carlo*, 2011.

[20] I. Sato and H. Nakagawa, "Approximation analysis of stochastic gradient langevin dynamics by using fokker-planck equation and ito process," in *ICML*, 2014.

[21] Y.-A. Ma, T. Chen, and E. Fox, "A complete recipe for stochastic gradient mcmc," in *NIPS*, 2015.

[22] J. Bornschein and Y. Bengio, "Reweighted wake-sleep," in *ICML*, 2015.

[23] X. Zhu, "Semi-supervised learning literature survey," *Technical report, University of Wisconsin-Madison*, 2006.

[24] H. Larochelle, M. I. Mandel, R. Pascanu, and Y. Bengio, "Learning algorithms for the classification restricted Boltzmann machine," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 643–669, 2012.

[25] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," in *NIPS*, 2014.

[26] D. Levy, M. D. Hoffman, and J. Sohl-Dickstein, "Generalizing Hamiltonian Monte Carlo with neural networks," in *ICLR*, 2018.

[27] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial networks," in *ICLR*, 2017.

[28] Z. Dai, A. Almahairi, P. Bachman, E. Hovy, and A. Courville, "Calibrating energy-based generative adversarial networks," in *ICLR*, 2017.

[29] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017.

[30] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein GANs," in *NIPS*, 2017.

[31] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, "Adversarially learned inference," in *ICLR*, 2017.

[32] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Technical report, University of Toronto*, 2009.

[33] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *NIPS*, 2016.

[34] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *NIPS*, 2017.

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

[36] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie, "Stacked generative adversarial networks," in *CVPR*, 2017.

[37] D. Warde-Farley and Y. Bengio, "Improving generative adversarial networks with denoising feature matching," in *ICLR*, 2017.

[38] X. Wei, Z. Liu, L. Wang, and B. Gong, "Improving the improved training of wasserstein GANs," in *ICLR*, 2018.

[39] Y. Mroueh and T. Sercu, "Fisher GAN," in *NIPS*, 2017.

[40] J. Adler and S. Lunz, "Banach wasserstein GAN," *arXiv preprint arXiv:1806.06621*, 2018.

[41] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *ICLR*, 2018.

[42] J. T. Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial networks," in *ICML*, 2016.

[43] L. Maaloe, C. K. Sonderby, S. K. Sonderby, and O. Winther, "Auxiliary deep generative models," in *ICML*, 2016.

[44] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko, "Semi-supervised learning with ladder networks," in *NIPS*, 2015.

[45] C. Li, T. Xu, J. Zhu, and B. Zhang, "Triple generative adversarial nets," in *NIPS*, 2017.

[46] Z. Gan, L. Chen, W. Wang, Y. Pu, Y. Zhang, H. Liu, C. Li, and L. Carin, "Triangle generative adversarial networks," in *NIPS*, 2017.

[47] Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. R. Salakhutdinov, "Good semi-supervised learning that requires a bad GAN," in *NIPS*, 2017.

[48] Y. Mroueh, C.-L. Li, T. Sercu, A. Raj, and Y. Cheng, "Sobolev GAN," in *ICLR*, 2018.

[49] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *arXiv preprint arXiv:1704.03976*, 2017.

[50] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," in *ICLR*, 2017.

[51] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *NIPS*, 2017.

[52] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[53] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

[54] Y. Chen, X. S. Zhou, and T. S. Huang, "One-class SVM for learning in image retrieval," in *International Conference on Image Processing*, 2001.

[55] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[56] L. Ruff, N. Görnitz, L. Deecke, S. A. Siddiqui, R. Vandermeulen, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *ICML*, 2018.

[57] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, "Deep structured energy based models for anomaly detection," in *ICML*, 2016.

[58] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *ICLR*, 2018.

[59] C.-S. F. B. L. Houssam Zenati, Manon Romain and V. Chandrasekhar, "Adversarially learned anomaly detection," in *International Conference on Data Mining*, 2018.

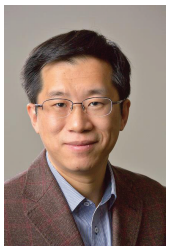
[60] A. Hyvärinen, "Estimation of non-normalized statistical models by score matching," *Journal of Machine Learning Research*, pp. 695–709, 2005.

[61] M. Lichman *et al.*, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[62] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[63] X. Liu and C.-J. Hsieh, "From adversarial training to generative adversarial networks," *arXiv preprint arXiv:1807.10454*, 2018.

Yunfu Song received the B.S. degree in Physics from Tsinghua University in 2017. Since 2017, he has been pursuing a master degree at the Department of Electronic Engineering, Tsinghua University under the supervision of Zhijian Ou. His research focuses on learning with undirected graphical models and neural networks.



Zhijian Ou received the B.S. degree (with the highest honor) in electronic engineering from Shanghai Jiao Tong University in 1998 and the Ph.D. degree in electronic engineering from Tsinghua University in 2003. Since 2003, he has been with the Department of Electronic Engineering in Tsinghua University and is currently an associate professor. From August 2014 to July 2015, he was a visiting scholar at Beckman Institute, University of Illinois at Urbana-Champaign. He has actively led research projects from NSF China (NSFC), China 863 High-tech Research and Development Program, China Ministry of Information Industry and China Ministry of Education, as well as joint-research projects with Intel, Panasonic, IBM, and Toshiba. His recent research interests include speech processing (speech recognition and understanding, speaker recognition, natural language processing) and machine intelligence (particularly with graphical models).

APPENDIX A
SEMI-SUPERVISED LEARNING WITH INCLUSIVE-NRFS

Algorithm 3 Semi-supervised learning with inclusive-NRFs

repeat
Sampling:
Draw a unsupervised minibatch $\mathcal{U} \sim \tilde{p}(\tilde{x})p_\theta(x)q_\phi(h|x)$ and a supervised minibatch $\mathcal{S} \sim \mathcal{L}$;
Updating:
Update θ by ascending:
$\frac{1}{|\mathcal{U}|}\sum_{(\tilde{x},x,h)\sim\mathcal{U}}\left[\nabla_\theta u_\theta(\tilde{x}) - \nabla_\theta u_\theta(x)\right]$
$+\alpha_d \frac{1}{|\mathcal{S}|}\sum_{(\tilde{x},\tilde{y})\sim\mathcal{S}}\left[\nabla_\theta log p_\theta(\tilde{y}|\tilde{x})\right]$
$-\frac{1}{|\mathcal{U}|}\sum_{(\tilde{x},x,h)\sim\mathcal{U}}\left[\alpha_c\nabla_\theta H(p_\theta(y|\tilde{x})) + \alpha_p\nabla_\theta\left[u_\theta(\tilde{x})\right]^2\right]$;
Update ϕ by ascending:
$\frac{1}{|\mathcal{U}|}\sum_{(\tilde{x},x,h)\sim\mathcal{U}}\nabla_\phi log q_\phi(x,h)$;
until convergence

APPENDIX B
REGULARIZATION LOSSES

Apart from the basic losses as shown in Eq.(6) and (18) for unsupervised and semi-supervised learning respectively, there are some regularization losses that are helpful to guide the training of inclusive-NRFs.

Confidence loss. Similar to [42], [45], we add the minimization of the conditional entropy of $p_\theta(y|\tilde{x})$ averaged over training data to the loss w.r.t. θ (i.e. the first line in Eq.17) as follows:

$$\begin{aligned} L_c(\theta) =& E_{\tilde{p}(\tilde{x})}\left[H(p_\theta(y|\tilde{x}))\right] \\ =& -E_{\tilde{p}(\tilde{x})}\left[\sum\nolimits_y p_\theta(y|\tilde{x})\log p_\theta(y|\tilde{x})\right] \end{aligned}$$

In this manner, we encourage the classifier $p_\theta(y|x)$ derived from the RF to make classifications confidently. In practice, we use stochastic gradients of $L_c(\theta)$ over minibatches in optimizing θ, as shown in Algorithm 3.

Potential control loss. For random fields, the data log-likelihood $logp_\theta(\tilde{x})$ is determined relatively by the potential value $u_\theta(\tilde{x})$. To avoid the potential values not to increase unreasonably, we could control the squared potential values, by minimizing:

$$L_p(\theta) = E_{\tilde{p}(\tilde{x})}\left[u_\theta(\tilde{x})\right]^2$$

In this manner, the potential values would be attracted to zeros. In practice, we use stochastic gradients of $L_p(\theta)$ over minibatches in optimizing θ, as shown in Algorithm 3.

APPENDIX C
PROOF OF PROPOSITION 3 (EXCLUSIVE-NRF)

Proposition 3. *For the RF as defined in Eq. (1), we have the following evidence upper bound:*

$$\begin{aligned} logp_\theta(\tilde{x}) =& \mathcal{U}(\tilde{x};\theta,\phi) - KL(q_\phi(x)||p_\theta(x)) \\ \leq& \mathcal{U}(\tilde{x};\theta,\phi), \end{aligned} \tag{19}$$

where $\mathcal{U}(\tilde{x};\theta,\phi) \triangleq u_\theta(\tilde{x}) - \left(E_{q_\phi(x)}\left[u_\theta(x)\right] + H\left[q_\phi(x)\right]\right)$.

Proof. Note that (i) $logp_\theta(\tilde{x}) = u_\theta(\tilde{x}) - logZ(\theta)$, and (ii) we have the following lower bound on $Z(\theta)$: $logZ(\theta) = log\int \exp(u_\theta(x))dx = log\int q_\phi(x)\frac{\exp(u_\theta(x))}{q_\phi(x)}dx \geq \int q_\phi(x) log\frac{\exp(u_\theta(x))}{q_\phi(x)}dx$. Combining (i) and (ii) gives Eq. (19). □

Furthermore, it can be seen that learning in [8] amounts to optimizing the following evidence upper bound:

$$\max_\theta \min_\phi \mathcal{U}(\tilde{x};\theta,\phi),$$

which is unfortunately not revealed in this manner in [8].

APPENDIX D
CONNECTION BETWEEN INCLUSIVE-NRFS AND GANS

Note that for the generator as defined in Eq. 4, we have the following joint density

$$logq_\phi(x,h) = -\frac{1}{2\sigma^2}||x - g_\phi(h)||^2 + constant.$$

The generator parameter ϕ is updated according to the gradient in Eq. 6, which is rewritten as follows:

$$\nabla_\phi = E_{p_\theta(x)q_\phi(h|x)}\left[\nabla_\phi log q_\phi(x,h)\right]$$

Specifically, we draw $(h', x') \sim q_\phi$ and then perform one-step SGLD to obtain (h, x). To simply the analysis of the connection, suppose $h \approx h'$, $x' \approx g_\phi(h') \approx g_\phi(h)$. Then we have

$$\begin{aligned} x =& x' + \delta_1\left[\frac{\partial}{\partial x} log p_\theta(x)\right]\Big|_{x=x'} + \sqrt{2\delta_1}\eta^{(1)}, \\ & \eta^{(1)} \sim \mathcal{N}(0, I) \end{aligned} \tag{20}$$

which further gives

$$\begin{aligned} x - g_\phi(h) \approx& \delta_1\left[\frac{\partial}{\partial x} log p_\theta(x)\right]\Big|_{x=g_\phi(h)} \\ =& \delta_1\left[\frac{\partial}{\partial x} u_\theta(x)\right]\Big|_{x=g_\phi(h)} \end{aligned}$$

The gradient in the updating step in Algorithm 1 becomes:

$$\begin{aligned} \nabla_\phi log q_\phi(x,h) =& \frac{1}{\sigma^2}\left[\frac{\partial}{\partial\phi} g_\phi(h)\right]\left[x - g_\phi(h)\right] \\ \approx& \frac{1}{\sigma^2}\left[\frac{\partial}{\partial\phi} g_\phi(h)\right]\delta_1\left[\frac{\partial}{\partial x} u_\theta(x)\right]\Big|_{x=g_\phi(h)} \\ =& \frac{1}{\sigma^2}\delta_1\left[\frac{\partial}{\partial\phi} u_\theta(g_\phi(h))\right] \end{aligned}$$

where $\frac{\partial}{\partial\phi} g_\phi(h)$ is a matrix of size $dim(\phi) \times dim(x)$. Therefore, the inclusive-NRF Algorithm 1 can be viewed to perform the following steps:

1) Draw an empirical example $\tilde{x} \sim \tilde{p}(\tilde{x})$.
2) Draw $h \sim p(h)$, $x' = g_\phi(h)$, and generate x by one-step-gradient according to Eq. 20.
3) Update θ by ascending: $\nabla_\theta u_\theta(\tilde{x}) - \nabla_\theta u_\theta(x)$.
4) Update ϕ by descending: $-\frac{\partial}{\partial\phi} u_\theta(g_\phi(h))$.

Now suppose that we interpret the potential function $u_\theta(x)$ as the discriminator in GANs (or the critic in Wasserstein GANs), which assign high scalar scores to empirical samples $\tilde{x} \sim \tilde{p}(\tilde{x})$ and low scalar scores to generated samples x. Then, the inclusive-NRF training could be viewed as playing a two-player minimax game:

$$\min_\phi \max_\theta E_{\tilde{x}\sim p_0}\left[u_\theta(\tilde{x})\right] - E_{h\sim p(h)}\left[u_\theta(g_\phi(h))\right], \tag{21}$$

except that in optimizing θ, the generated sample are further revised by taking one-step-gradient of $u_\theta(x)$ w.r.t. x (as shown in the above Step 2). The 'discriminator' u_θ is trained to discriminate between empirical samples and generated samples, while the generator q_ϕ is trained to fool the discriminator by assigning higher scores to generated samples. From the above analysis, we find some interesting connections between inclusive-NRFs and existing studies in GANs.

- The optimization shown in Eq. (21) is in fact the same as that in Wasserstein GANs (Theorem 3 in [29]), *except that* in Wasserstein GANs, the critic $u_\theta(x)$ is constrained to be 1-Lipschitz continuous. So hopefully we can improve the inclusive-NRF training by constraining the discriminator $u_\theta(x)$ to be 1-Lipschitz continuous, e.g. by utilizing the recently developed technique of spectral normalization of weight matrices in the discriminator as in [41].
- To optimize θ, the generated sample is obtained by taking one-step-gradient of $u_\theta(x)$ w.r.t. x. The tiny perturbation guided by the gradient to increase the score for the generated sample in fact creates an adversarial example. A similar idea is presented in [63] that when feeding real samples to the discriminator, 5 steps of PGD (Projected Gradient Descent) attack is taken to decrease the score to create adversarial samples. It is shown in [63] that training the discriminator with adversarial examples significantly improves the GAN traning. Hopefully in training the discriminator in inclusive-NRFs, the adversarial attack could be increasing scores for generated samples, or decreasing scores for real samples, or a mixed one.
- The above analysis assume the use of one-step SGLD. It can be seen that running finite steps of SGLD in sample revision in fact create adversarial samples to fool the discriminator.

APPENDIX E
DETAILS OF EXPERIMENTS

E.1 Image generation on CIFAR-10

Network architectures. For convenience, we refer to the two neural networks in implementing the potential u_θ and the generator q_ϕ in NRFs as the potential network and the generator network, respectively. For comparison of different methods, we use the same network architectures as in Table 4 in [41] (ResNet using spectral normalization) for unsupervised learning of NRFs. For supervised learning, we use the semi-supervised inclusive-NRF Algorithm 3 over all labeled images. The difference in network architectures used for semi-supervised and unsupervised learning of inclusive-NRFs is that for SSL, the output layer of the potential network contains $K = 10$ scalar units, while a single scalar output unit is used for unsupervised learning.

Hyperparameters. We use Adam optimizer with the hyperparameter ($\beta_1 = 0, \beta_2 = 0.9$ and $\alpha = 0.0003$ for random fields, $\alpha = 0.0001$ for generators). For sample revision in inclusive-NRFs, we empirically choose SGLD with $L = 1$ ($\delta_l = 0.0003$). More revision steps do not significantly improve unsupervised IS, as discussed in section 5.5 Note that we use the potential control loss in both unsupervised ($\alpha_p = 0.1$) and supervised ($\alpha_d = 1, \alpha_p = 0.1$) settings, which is found beneficial for stable training.

Evaluation. Figure 7(c)(d) show the generated samples from inclusive-NRFs for unsupervised and supervised settings respectively. We calculate inception score (IS) and Frechet inception distance (FID) in the same way as in [41]. We trained 10 models with different random seeds, and then generate 5000 images 10 times and compute the average inception score and the standard deviation. We compute FID between the true distribution and the generated distribution empirically over 10000 (test set) and 5000 samples.

E.2 Semi-supervised experiment on MNIST, SVHN and CIFAR-10

The network architectures (taken from the released code from [33] and widely used in [45], [47]) and hyperparameters for semi-supervised inclusive-NRFs on MNIST, SVHN and CIFAR-10 are listed in Table 9, Table 10 and Table 11 respectively. We use SGHMC for semi-supervised inclusive-NRFs for all three datasets, with empirical revision hyperparameters ($\beta = 0.5, \delta_l = 0.003$) for MNIST and CIFAR-10, and ($\beta = 0.5, \delta_l = 0.01$) for SVHN. The confidence loss is employed for semi-supervised inclusive-NRFs on MNIST and SVHN, and the potential control loss is employed on CIFAR-10.

Figure 7(a)(b) show the generated samples from semi-supervised inclusive-NRFs trained over SVHN and CIFAR-10 respectively.

E.3 Ablation study of inclusive-NRFs on CIFAR-10

For unsupervised learning, we use the same networks as in Table 3 in [41] (standard CNN using spectral normalization). We use Adam optimizer with the hyperparameter ($\alpha = 0.0002, \beta_1 = 0, \beta_2 = 0.9$). For semi-supervised learning, the experimental setting is the same as in section E.2 including the networks, number of labels, etc. For different revision steps, we use ($\delta_l = 0.003$) for SGLD, and ($\beta = 0.5, \delta_l = 0.003$) for SGHMC. The potential control loss is employed in both unsupervised ($\alpha_p = 0.1$) and semi-supervised ($\alpha_d = 100, \alpha_p = 0.1$) learning.

E.4 Anomaly detection

For anomaly detection, we train inclusive-NRFs with Algorithm 1 and potential control loss. The network architectures and hyperparameters for KDDCUP, MNIST and CIFAR-10 datasets are listed in Table 8, 12 and 13. For sample revision, we use SGLD ($\delta_l = 0.003$) for KDDCUP, SGHMC ($\beta = 0.5, \delta_l = 0.003$) for MNIST and SGLD ($\delta_l = 0.03$) for CIFAR-10.

APPENDIX F
LATENT SPACE INTERPOLATION

Figure 5 shows that the auxiliary generator smoothly outputs transitional samples as the latent code h moves linearly in the latent space. The interpolated generation demonstrates that the model has indeed learned an abstract representation of the data.

APPENDIX G
CLASS-CONDITIONAL GENERATION

Figure 6 shows class-conditional generation results on MNIST with semi-supervised inclusive-NRFs. Notice that the generator does not explicitly include class labels, thus it is unable to perform class-conditional generation directly. However, the random field has modeling of $p_\theta(x, y)$, based on which we can perform class-conditional generation as follows:

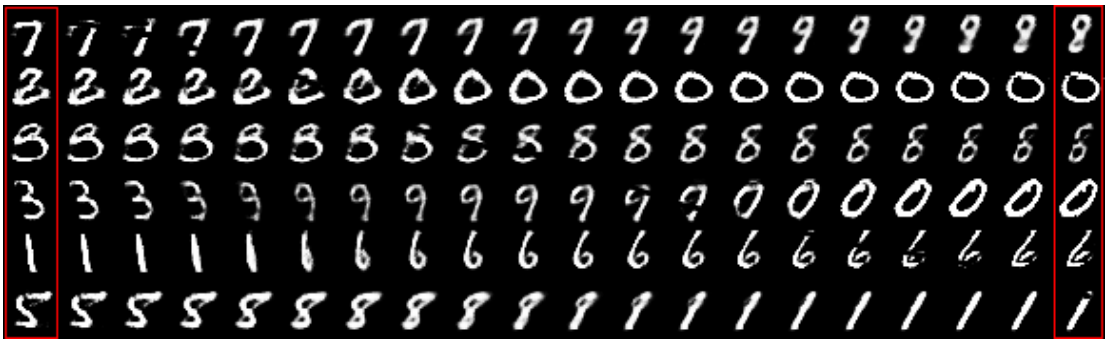


Fig. 5: Latent space interpolation with inclusive-NRFs on MNIST. The leftmost and rightmost columns are from stochastic generations x_1 with latent code h_1 and x_2 with h_2. The columns in between correspond to the generations from the latent codes interpolated linearly from h_1 to h_2.

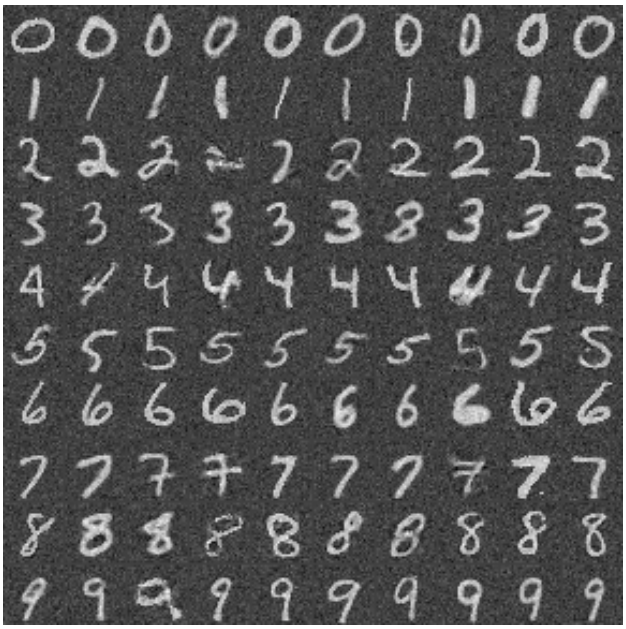


Fig. 6: Conditional generated samples from semi-supervised inclusive-NRFs trained on MNIST. Due to sample revision, the background pixels are not purely black.

1) Generate a sample x unconditionally, by ancestral sampling with the generator.
2) Predict the label y for the sample x by the random field;
3) Starting from x, running SGLD/SGHMC revision with $p_\theta(x|y)$ as the target density by fixing y. The resulting samples could be viewed as conditional generations, according to Theorem 1.

TABLE 7: Network architectures and hyperparameters for the 2D GMM data.

Random Field	**Generator**
Input 2-dim data	Noise h (2-dim)
MLP 100 units, Leaky ReLU, Weight norm	MLP 50 units, ReLU, Batch norm
MLP 100 units, Leaky ReLU, Weight norm	MLP 50 units, ReLU, Batch norm
MLP 1 unit, Linear, Weight norm	MLP 2 units, Linear
Batch size	100
Number of iterations	160,000
Leaky ReLU slope	0.2
Learning rate	0.001
Optimizer	Adam ($\beta_1 = 0.5, \beta_2 = 0.9$)
Sample revision steps	$L = 10$

TABLE 8: Network architectures and hyperparameters for anomaly detection on KDDCUP dataset.

Random Field	**Generator**
Input 120-dim data	Noise h (5-dim)
MLP 60 units, Tanh, Weight norm	MLP 10 units, Tanh, Batch Norm
MLP 30 units, Tanh, Weight norm	MLP 30 units, Tanh, Batch Norm
MLP 10 units, Tanh, Weight norm	MLP 60 units, Tanh, Batch Norm
MLP 1 unit, Linear, Weight norm	MLP 120 unit, Linear, Weight norm
Batch size	1024
Number of epochs	30
Learning rate	0.0001 fro RF, 0.0003 for G
Optimizer	Adam ($\beta_1 = 0.5, \beta_2 = 0.999$)
Sample revision steps	$L = 10$
α_p	0.1

(a) (b) (c) (d)

Fig. 7: Generated samples from semi-supervised inclusive-NRFs (i.e. trained for SSL) on SVHN and CIFAR-10 are shown in (a) and (b) respectively. Generated samples from unsupervised and supervised training of inclusive-NRFs on CIFAR-10 are shown in (c) and (d) respectively.

TABLE 9: Network architectures and hyperparameters for semi-supervised inclusive-NRFs on MNIST.

Random Field	**Generator**
Input 28×28 Gray Image	Noise h (100-dim)
MLP 1000 units, Leaky ReLU, Weight norm	MLP 500 units, Sotfplus, Batch norm
MLP 500 units, Leaky ReLU, Weight norm	MLP 500 units, Sotfplus, Batch norm
MLP 250 units, Leaky ReLU, Weight norm	MLP 784 units, Sigmoid
MLP 250 units, Leaky ReLU, Weight norm	
MLP 250 units, Leaky ReLU, Weight norm	
MLP 10 units, Linear, Weight norm	
Batch size	100
Number of epochs	200
Leaky ReLU slope	0.2
Learning rate	0.001 for RF, 0.003 for G
Optimizer	Adam ($\beta_1 = 0.0, \beta_2 = 0.9$)
Sample revision steps	$L = 20$
α in SSL	$\alpha_d = 10$, $\alpha_c = 10$, $\alpha_p = 0$

TABLE 10: Network architectures and hyperparameters for semi-supervised inclusive-NRFs on SVHN.

Random Field	**Generator**
Input 32×32 Colored Image	Noise h (100-dim)
3×3 conv. 64, Leaky ReLU, Weight norm	MLP 8192 units, ReLU, Batch norm
3×3 conv. 64, Leaky ReLU, Weight norm	Reshape $512 \times 4 \times 4$
3×3 conv. 64, stride=2, Leaky ReLU, Weight norm, dropout2d=0.5	5×5 deconv. 256, ReLU, Stride=2
3×3 conv. 128, Leaky ReLU, Weight norm	5×5 deconv. 128, ReLU, Stride=2
3×3 conv. 128, Leaky ReLU, Weight norm	5×5 deconv. 3, Tanh, Stride=2
3×3 conv. 128, stride=2, Leaky ReLU, Weight norm, dropout2d=0.5	
3×3 conv. 128, Leaky ReLU, Weight norm	
1×1 conv. 128, Leaky ReLU, Weight norm	
1×1 conv. 128, Leaky ReLU, Weight norm	
MLP 10 units, Linear, Weight norm	
Batch size	100
Number of epochs	400
Leaky ReLU slope	0.2
Learning rate	0.001
Optimizer	Adam ($\beta_1 = 0.0, \beta_2 = 0.9$)
Sample revision steps	$L = 10$
α in SSL	$\alpha_d = 10$, $\alpha_c = 10$, $\alpha_p = 0$

TABLE 11: Network architectures and hyperparameters for semi-supervised inclusive-NRFs on CIFAR-10.

Random Field	**Generator**
Input 32×32 Colored Image	Noise h (100-dim)
3×3 conv. 128, Leaky ReLU, Weight norm	MLP 8192 units, ReLU, batch norm
3×3 conv. 128, Leaky ReLU, Weight norm	Reshape $512 \times 4 \times 4$
3×3 conv. 128, Leaky ReLU, Weight norm	5×5 deconv. 256, ReLU, Stride=2
2×2 MaxPool, dropout2d=0.5	5×5 deconv. 128 ReLU, stride=2
3×3 conv. 256, Leaky ReLU, Weight norm	5×5 deconv. 3, Tanh, Stride=2
3×3 conv. 256, Leaky ReLU, Weight norm	
3×3 conv. 256, Leaky ReLU, Weight norm	
2×2 MaxPool, dropout2d=0.5	
3×3 conv. 512, Leaky ReLU, Weight norm	
1×1 conv. 256, Leaky ReLU, Weight norm	
1×1 conv. 128, Leaky ReLU, Weight norm	
MLP 10 units, Linear, Weight norm	
Batch size	100
Number of epochs	600
Leaky ReLU slope	0.2
Learning rate	0.001
Optimizer	Adam ($\beta_1 = 0.0, \beta_2 = 0.9$)
Sample revision steps	$L = 10$
α in SSL	$\alpha_d = 100$, $\alpha_c = 0$, $\alpha_p = 0.1$

TABLE 12: Network architectures and hyperparameters for anomaly detection on MNIST dataset.

Random Field	**Generator**
Input 28×28 Gray Image	Noise h (100-dim)
MLP 1000 units, Leaky ReLU, Weight norm	MLP 500 units, Sotfplus, Batch norm
MLP 500 units, Leaky ReLU, Weight norm	MLP 500 units, Sotfplus, Batch norm
MLP 250 units, Leaky ReLU, Weight norm	MLP 784 units, Sigmoid
MLP 250 units, Leaky ReLU, Weight norm	
MLP 250 units, Leaky ReLU, Weight norm	
MLP 1 units, Linear, Weight norm	
Batch size	100
Number of epochs	50
Leaky ReLU slope	0.2
Learning rate	0.003 for RF, 0.001 for G
Optimizer	Adam ($\beta_1 = 0.0, \beta_2 = 0.9$)
Sample revision steps	$L = 20$
α_p	1

TABLE 13: Network architectures and hyperparameters for anomaly detection on CIFAR-10 dataset.

Random Field	**Generator**
Input 32×32 Colored Image	Noise h (100-dim)
3×3 conv. 96, Leaky ReLU, Weight norm	MLP 8192 units, ReLU, batch norm
3×3 conv. 96, Leaky ReLU, Weight norm	Reshape $512 \times 4 \times 4$
3×3 conv. 96, stride=2, Leaky ReLU, Weight norm	5×5 deconv. 256, ReLU, Stride=2
3×3 conv. 192, Leaky ReLU, Weight norm	5×5 deconv. 128 ReLU, stride=2
3×3 conv. 192, Leaky ReLU, Weight norm	5×5 deconv. 3, Tanh, Stride=2
3×3 conv. 192, stride=2, Leaky ReLU, Weight norm	
3×3 conv. 192, Leaky ReLU, Weight norm	
1×1 conv. 192, Leaky ReLU, Weight norm	
1×1 conv. 192, Leaky ReLU, Weight norm	
MLP 1 units, Linear, Weight norm	
Batch size	64
Number of epochs	100
Leaky ReLU slope	0.2
Learning rate	0.001
Optimizer	Adam ($\beta_1 = 0.0, \beta_2 = 0.9$)
Sample revision steps	$L = 10$
α_p	0.1