Flow Contrastive Estimation of Energy-Based Models

Ruiqi Gao
UCLA
ruiqigao@ucla.edu

Erik Nijkamp
UCLA
enijkamp@ucla.edu

Diederik P. Kingma
Google
durk@google.com

Zhen Xu
Google
zhenxu@google.com

Andrew M. Dai
Google
adai@google.com

Ying Nian Wu
UCLA
ywu@stat.ucla.edu

Abstract

This paper studies a training method to jointly estimate an energy-based model and a flow-based model, in which the two models are iteratively updated based on a shared adversarial value function. This joint training method has the following traits. (1) The update of the energy-based model is based on noise contrastive estimation, with the flow model serving as a strong noise distribution. (2) The update of the flow model approximately minimizes the Jensen-Shannon divergence between the flow model and the data distribution. (3) Unlike generative adversarial networks (GAN) which estimates an implicit probability distribution defined by a generator model, our method estimates two explicit probabilistic distributions on the data. Using the proposed method we demonstrate a significant improvement on the synthesis quality of the flow model, and show the effectiveness of unsupervised feature learning by the learned energy-based model. Furthermore, the proposed training method can be easily adapted to semi-supervised learning. We achieve competitive results to the state-of-the-art semi-supervised learning methods.

1 Introduction

Recently, *flow-based models* (henceforth simply called *flow models*) have gained popularity as a type of deep generative model [6, 7, 25, 13, 3, 30, 59] and for use in variational inference [27, 52, 26].

Flow models have two properties that set them apart from other types of deep generative models: (1) they allow for efficient evaluation of the density function, and (2) they allow for efficient sampling from the model. Efficient evaluation of the log-density allows flow models to be directly optimized towards the log-likelihood objective, unlike variational autoencoders (VAEs) [27, 53], which are optimized towards a *bound* on the log-likelihood, and generative adversarial networks (GANs) [12]. Auto-regressive models [14, 49, 55], on the other hand, are (in principle) inefficient to sample from, since synthesis requires computation that is proportional to the dimensionality of the data.

These properties of efficient density evaluation and efficient sampling are typically viewed as advantageous. However, they have a potential downside: these properties also acts as *assumptions* on the true data distribution that they are trying to model. By choosing a flow model, one is making the assumption that the true data distribution is one that is in principle simple to sample from, and is computationally efficient to normalize. In addition, flow models assume that the data is generated by a finite sequence of invertible functions. If these assumptions do not hold, flow-based models can result in a poor fit.

On the other end of the spectrum of deep generative models lies the family of energy-based models (EBMs) [35, 47, 22, 65, 63, 11, 31, 48, 8, 9]. Energy-based models define an unnormalized density that is the exponential of the negative *energy function*. The energy function is directly defined as

Work submitted.

Interesting comment.

a (learned) scalar function of the input, and is often parameterized by a neural network, such as a convolutional network [34, 29]. Evaluation of the density function for a given datapoint involves calculating a normalizing constant, which requires an intractable integral. Sampling from EBMs is expensive and requires approximation as well, such as computationally expensive Markov Chain Monte Carlo (MCMC) sampling. EBMs therefore do not make any of the two assumptions above: they do not assume that the density of data is easily normalized, and they do not assume efficient synthesis. Moreover, they do not constrain the data distribution by invertible functions.

Contrasting an EBM with a flow model, the former is on the side of representation where different layers represent features of different complexities, whereas the latter is on the side of learned computation, where each layer, or each transformation is like a step in the computation. The EBM is like an objective function or a target distribution whereas the flow model is like a finite step iterative algorithm or a sampler. The EBM can be simpler and more flexible in form than the flow model which is highly constrained, and thus the EBM may capture the modes of the data distribution more accurately than the flow model.

In contrast, the flow model is capable of direct generation via ancestral sampling, which is sorely lacking in an EBM. It may thus be desirable to train the two models jointly, combining the best of both worlds. This is the goal of this paper.

Our joint training method is inspired by the noise contrastive estimation (NCE) of [15], where an EBM is learned discriminatively by classifying the real data and the data generated by a noise model. In NCE, the noise model must have an explicit normalized density function. Moreover, it is desirable for the noise distribution to be close to the data distribution for accurate estimation of the EBM. However, the noise distribution can be far away from the data distribution. The flow model can potentially transform or transport the noise distribution to a distribution closer to the data distribution. With the advent of strong flow-based generative models [6, 7, 25], it is natural to recruit the flow model as the contrast distribution for noise contrastive estimation of the EBM.

However, even with the flow-based model pre-trained by maximum likelihood estimation (MLE) on the data distribution, it may still not be strong enough as a contrast distribution, in the sense that the synthesized examples generated by the pre-trained flow model may still be distinguished from the real examples by a classifier based on an EBM. Thus, we want the flow model to be a stronger contrast or a stronger training opponent for EBM. To achieve this goal, we can simply use the same objective function of NCE, which is the log-likelihood of the logistic regression for classification. While NCE updates the EBM by maximizing this objective function, we can also update the flow model by minimizing the same objective function to make the classification task harder for the EBM. Such update of flow model combines MLE and variational approximation, and helps correct the over-dispersion of MLE. If the EBM is close to the data distribution, this amounts to minimizing the Jensen-Shannon divergence (JSD) [12] between the data distribution and the flow model. In this sense, the learning scheme relates closely to GANs [12]. However, unlike GANs, which learns a generator model that defines an implicit probability density function via a low-dimensional latent vector, our method learns two probabilistic models with explicit probability densities (a normalized one and an unnormalized one).

The contributions of our paper are as follows. We explore a parameter estimation method that couples estimation of an EBM and a flow model using a shared objective function. It improves NCE with a flow-transformed noise distribution, and it modifies MLE of the flow model to approximate JSD minimization, and helps correct the over-dispersion of MLE. Experiments on 2D synthetic data show that the learned EBM achieves accurate density estimation with a much simpler network structure than the flow model. On real image datasets, we demonstrate a significant improvement on the synthesis quality of the flow model, and the effectiveness of unsupervised feature learning by the energy-based model. Furthermore, we show that the proposed method can be easily adapted to semi-supervised learning, achieving performance comparable to state-of-the-art semi-supervised methods.

2 Related work

For learning the energy-based model by MLE, the main difficulty lies in drawing fair samples from the current model. A prominent approximation of MLE is the contrastive divergence (CD) [19] framework, requiring MCMC initialized from the data distribution. CD has been generalized to

persistent CD [58], and has more recently been generalized to modified CD [11], adversarial CD [22, 5, 16] with modern CNN structure. [48, 8] scale up sampling-based methods to large image datasets with white noise as the starting point of sampling. However, these sampling based methods may still have difficulty traversing different modes of the learned model, which may result in biased model, and may take a long time to converge. An advantage of noise contrastive estimation (NCE), and our adaptive version of it, is that it avoids MCMC sampling in estimation of the energy-based model, by turning the estimation problem into a classification problem.

Generalizing from [60], [20, 33, 36] developed an introspective parameter estimation method, where the EBM is discriminatively learned and composed of a sequence of discriminative models obtained through the learning process.

NCE and it variants has gained popularity in natural language processing (NLP) [17, 50, 2, 4]. [44, 43] applied NCE to log-bilinear models and in [61] NCE is applied to neural probabilistic language models. NCE shows effectiveness in typical NLP tasks such as word embeddings [41] and order embeddings [62].

In the context of inverse reinforcement learning, [37] proposes a guided policy search method, and [9] connects it to GAN. Our method is closely related to this method, where the energy function can be viewed as the cost function, and the flow model can be viewed as the unrolled policy.

3 Learning method

3.1 Energy-based model

Let x be the input variable, such as an image. We use $p_\theta(x)$ to denote a model's probability density function of x with parameter θ. The energy-based model (EBM) is defined as follows:

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp[f_\theta(x)], \tag{1}$$

where $f_\theta(x)$ is defined by a bottom-up convolutional neural network whose parameters are denoted by θ. The normalizing constant $Z(\theta) = \int \exp[f_\theta(x)]dx$ is intractable to compute exactly for high-dimensional x.

3.1.1 Maximum likelihood estimation

The energy-based model in eqn. 1 can be estimated from unlabeled data by maximum likelihood estimation (MLE). Suppose we observe training examples $\{x_i, i = 1, ..., n\}$ from unknown true distribution $p_{\text{data}}(x)$. We can view this dataset as forming empirical data distribution, and thus expectation with respect to $p_{\text{data}}(x)$ can be approximated by averaging over the training examples. In MLE, we seek to maximize the log-likelihood function

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(x_i). \tag{2}$$

Maximizing the log-likelihood function is equivalent to minimizing the Kullback-Leibler divergence $\text{KL}(p_{\text{data}}||p_\theta)$ for large n. Its gradient can be written as:

$$-\frac{\partial}{\partial \theta} \text{KL}(p_{\text{data}}||p_\theta) = \mathbb{E}_{p_{\text{data}}}\left[\frac{\partial}{\partial \theta} f_\theta(x)\right] - \mathbb{E}_{p_\theta}\left[\frac{\partial}{\partial \theta} f_\theta(x)\right], \tag{3}$$

which is the difference between the expectations of the gradient of $f_\theta(x)$ under p_{data} and p_θ respectively. The expectations can be approximated by averaging over the observed examples and synthesized samples generated from the current model $p_\theta(x)$ respectively. The difficulty lies in the fact that sampling from $p_\theta(x)$ requires MCMC such as Hamiltonian monte carlo or Langevin dynamics [3, 66], which may take a long time to converge, especially on high dimensional and multi-modal space such as image space.

The MLE of $p_\theta(x)$ seeks to cover all the models of $p_{\text{data}}(x)$. Given the flexibility of model form of $f_\theta(x)$, the MLE of $p_\theta(x)$ has the chance to approximate $p_{\text{data}}(x)$ reasonably well.

3.1.2 Noise contrastive estimation

Noise contrastive estimation (NCE) [15] can be used to learn the EBM, by including the normalizing constant as another learnable parameter. Specifically, for an energy-based model $p_\theta(x) = \frac{1}{Z(\theta)} \exp[f_\theta(x)]$, we define $p_\theta(x) = \exp[f_\theta(x) - c]$, where $c = \log Z(\theta)$. c is now treated as a free parameter, and is included into θ. Suppose we observe training examples $\{x_i, i = 1, ..., n\}$, and we have generated examples $\{\tilde{x}_i, i = 1, ..., n\}$ from a noise distribution $q(x)$. Then θ can be estimated by maximizing the following objective function:

$$J(\theta) = \mathbb{E}_{p_{\text{data}}}\left[\log \frac{p_\theta(x)}{p_\theta(x) + q(x)}\right] + \mathbb{E}_q\left[\log \frac{q(x)}{p_\theta(x) + q(x)}\right], \tag{4}$$

which transforms estimation of EBM into a classification problem.

The objective function connects to logistic regression in supervised learning in the following sense. Suppose for each training or generated examples we assign a binary class label y: $y = 1$ if x is from training dataset and $y = 0$ if x is generated from $q(x)$. In logistic regression, the posterior probabilities of classes given the data x are estimated. As the data distribution $p_{\text{data}}(x)$ is unknown, the class-conditional probability $p(\cdot|y = 1)$ is modeled with $p_\theta(x)$. And $p(\cdot|y = 0)$ is modeled by $q(x)$. Suppose we assume equal probabilities for the two class labels, i.e., $p(y = 1) = p(y = 0) = 0.5$. Then we obtain the posterior probabilities:

$$p_\theta(y = 1|x) = \frac{p_\theta(x)}{p_\theta(x) + q(x)} := u(x, \theta). \tag{5}$$

The class-labels y are Bernoulli-distributed, so that the log-likelihood of the parameter θ becomes

$$l(\theta) = \sum_{i=1}^{n} \log u(x_i; \theta) + \sum_{i=1}^{n} \log(1 - u(\tilde{x}_i; \theta)), \tag{6}$$

which is, up to a factor of $1/n$, an approximation of eqn. 4.

The choice of the noise distribution $q(x)$ is a design issue. Generally speaking, we expect $q(x)$ to satisfy the following: (1) analytically tractable expression of normalized density; (2) easy to draw samples from; (3) close to data distribution. In practice, (3) is important for learning a model over high dimensional data. If $q(x)$ is not close to the data distribution, the classification problem would be too easy and would not require p_θ to learn much about the modality of the data.

3.2 Flow-based model

A flow model is of the form

$$x = g_\alpha(z); \; z \sim q_0(z), \tag{7}$$

where q_0 is a known noise distribution. g_α is a composition of a sequence of invertible transformations where the log-determinants of the Jacobians of the transformations can be explicitly obtained. α denotes the parameters. Let $q_\alpha(x)$ be the probability density of the model given a datapoint x with parameter α. Then under the *change of variables* $q_\alpha(x)$ can be expressed as

$$q_\alpha(x) = q_0(g_\alpha^{-1}(x)) |\det(\partial g_\alpha^{-1}(x)/\partial x)|. \tag{8}$$

More specifically, suppose g_α is composed of a sequence of transformations $g_\alpha = g_{\alpha_1} \circ \cdots \circ g_{\alpha_m}$. The relation between z and x can be written as $z \leftrightarrow h_1 \leftrightarrow \cdots \leftrightarrow h_{m-1} \leftrightarrow x$. And thus we have

$$q_\alpha(x) = q_0(g_\alpha^{-1}(x)) \Pi_{i=1}^{m} |\det(\partial h_{i-1}/\partial h_i)|, \tag{9}$$

where we define $z := h_0$ and $x := h_m$ for conciseness. With carefully designed transformations, as explored in flow-based methods, the determinant of the Jacobian matrix $(\partial h_{i-1}/\partial h_i)$ can be incredibly simple to compute. The key idea is to choose transformations whose Jacobian is a triangle matrix, so that the determinant becomes

$$|\det(\partial h_{i-1}/\partial h_i)| = \Pi |\text{diag}(\partial h_{i-1}/\partial h_i)|. \tag{10}$$

The following are the two scenarios for estimating q_α:

(1) Generative modeling by MLE [6, 7, 25, 13, 3, 30, 59], based on $\min_\alpha \mathrm{KL}(p_{\mathrm{data}}\|q_\alpha)$, where again $\mathbb{E}_{p_{\mathrm{data}}}$ can be approximated by average over observed examples.

(2) Variational approximation to an unnormalized target density p [27, 52, 26, 23, 21], based on $\min_\alpha \mathrm{KL}(q_\alpha\|p)$, where

$$\begin{aligned}\mathrm{KL}(q_\alpha\|p) &= \mathbb{E}_{q_\alpha}[\log q_\alpha(x)] - \mathbb{E}_{q_\alpha}[\log p(x)] \\ &= \mathbb{E}_z[\log q_0(z) + \log|\det(g'(z))|] - \mathbb{E}_{q_\alpha}[\log p(x)].\end{aligned} \tag{11}$$

$\mathrm{KL}(q_\alpha\|p)$ is the difference between energy and entropy, i.e., we want q_α to have low energy but high entropy. $\mathrm{KL}(q_\alpha\|p)$ can be calculated without inversion of g_α.

When q_α appears on the right of KL-divergence, as in (1), it is forced to cover most of the modes of p_{data}, When q_α appears on the left of KL-divergence, as in (2), it tends to chase the major modes of p while ignoring the minor modes [45, 10]. As shown in the following section, our proposed method learns a flow model by combining (1) and (2).

3.3 Flow Contrastive Estimation

A natural improvement to NCE is to transform the noise so that the resulting distribution is closer to the data distribution. This is exactly what the flow model achieves. A flow model is of the form $x = g_\alpha(z)$, where $z \sim q_0(z)$, which is a known noise distribution. g_α is a composition of a sequence of invertible transformations, and α denotes the parameters. Let $q_\alpha(x)$ be the probability density of x. It fulfills (1) and (2) of the requirements of NCE. However, in practice, we find that a pre-trained $q_\alpha(x)$, such as learned by MLE, is not strong enough for learning an EBM $p_\theta(x)$ because the synthesized data from the MLE of $q_\alpha(x)$ can still be easily distinguished from the real data by an EBM. Thus, we propose to iteratively train the EBM and flow model, in which case the flow model is adaptively adjusted to become a stronger contrast distribution or a stronger training opponent for EBM. This is achieved by a parameter estimation scheme similar to GAN, where $p_\theta(x)$ and $q_\alpha(x)$ play a minimax game with a unified value function: $\min_\alpha \max_\theta V(\theta, \alpha)$,

$$\begin{aligned}V(\theta,\alpha) = \mathbb{E}_{p_{\mathrm{data}}}&\left[\log \frac{p_\theta(x)}{p_\theta(x)+q_\alpha(x)}\right] \\ &+ \mathbb{E}_z\left[\log \frac{q_\alpha(g_\alpha(z))}{p_\theta(g_\alpha(z))+q_\alpha(g_\alpha(z))}\right],\end{aligned} \tag{12}$$

where $\mathbb{E}_{p_{\mathrm{data}}}$ is approximated by averaging over observed samples $\{x_i, i = 1, ..., n\}$, while $\mathbb{E}_z$ is approximated by averaging over negative samples $\{\tilde{x}_i, i = 1, ..., n\}$ drawn from $q_\alpha(x)$, with $z_i \sim q_0(z)$ independently for $i = 1, ..., n$. In the experiments, we choose Glow [25] as the flow-based model. The algorithm can either start from a randomly initialized Glow model or a pre-trained one by MLE. Here we assume equal prior probabilities for observed samples and negative samples. It can be easily modified to the situation where we assign a higher prior probability to the negative samples, given the fact we have access to infinite amount of free negative samples.

The objective function can be interpreted from the following perspectives:

(1) Noise contrastive estimation for EBM. The update of θ can be seen as noise contrastive estimation of $p_\theta(x)$, but with a flow-transformed noise distribution $q_\alpha(x)$ which is adaptively updated. The training is essentially a logistic regression. However, unlike regular logistic regression for classification, for each x_i or $\tilde{x}_i$, we must include $\log q_\alpha(x_i)$ or $\log q_\alpha(\tilde{x}_i)$ as an example-dependent bias term. This forces $p_\theta(x)$ to replicate $q_\alpha(x)$ in addition to distinguishing between $p_{\mathrm{data}}(x)$ and $q_\alpha(x)$, so that $p_\theta(x_i)$ is in general larger than $q_\alpha(x_i)$, and $p_\theta(\tilde{x}_i)$ is in general smaller than $q_\alpha(\tilde{x}_i)$.

(2) Minimization of Jensen-Shannon divergence for the flow model. If $p_\theta(x)$ is close to the data distribution, then the update of α is approximately minimizing the Jensen-Shannon divergence between the flow model q_α and data distribution p_{data}:

$$\begin{aligned}\mathrm{JSD}(q_\alpha\|p_{\mathrm{data}}) = &\mathrm{KL}(p_{\mathrm{data}}\|(p_{\mathrm{data}}+q_\alpha)/2) \\ &+ \mathrm{KL}(q_\alpha\|(p_{\mathrm{data}}+q_\alpha)/2).\end{aligned} \tag{13}$$

Its gradient w.r.t. α equals the gradient of $-\mathbb{E}_{p_{\mathrm{data}}}[\log((p_\theta+q_\alpha)/2)] + \mathrm{KL}(q_\alpha\|(p_\theta+q_\alpha)/2)$. The gradient of the first term resembles MLE, which forces q_α to cover the modes of data distribution, and tends to lead to an over-dispersed model, which is also pointed out in [25]. The gradient of

the second term is similar to reverse Kullback-Leibler divergence between q_α and p_θ, or variational approximation of p_θ by q_α, which forces q_α to chase the modes of p_θ [45, 10]. This may help correct the over-dispersion of MLE, and combines the two scenarios of estimating the flow-based model q_α as described in section 3.2.

(3) Connection with GAN. Our parameter estimation scheme is closely related to GAN. In GAN, the discriminator D and generator G play a minimax game: $\min_G \max_D V(G, D)$,

$$V(G, D) = \mathbb{E}_{p_{\text{data}}} [\log D(x)] + \mathbb{E}_z [\log(1 - D(G(z_i)))] . \tag{14}$$

The discriminator $D(x)$ is learning the probability ratio $p_{\text{data}}(x)/(p_{\text{data}}(x)+p_G(x))$, which is about the difference between p_{data} and p_G [9]. In the end, if the generator G learns to perfectly replicate p_{data}, then the discriminator D ends up with a random guess. However, in our method, the ratio is explicitly modeled by p_θ and q_α. p_θ must contain all the learned knowledge in q_α, in addition to the difference between p_{data} and q_α. In the end, we learn two explicit probability distributions p_θ and q_α as approximations to p_{data}.

Henceforth we simply refer to the proposed method as flow constrastive estimation, or FCE.

3.4 Semi-supervised learning

A class-conditional energy-based model can be transformed into a discriminative model in the following sense. Suppose there are K categories $k = 1, ..., K$, and the model learns a distinct density $p_{\theta_k}(x)$ for each k. The networks $f_{\theta_k}(x)$ for $k = 1, ..., K$ may share common lower layers, but with different top layers. Let ρ_k be the prior probability of category k, for $k = 1, ..., K$. Then the posterior probability for classifying x to the category k is a softmax multi-class classifier

$$P(k|x) = \frac{\exp(f_{\theta_k}(x) + b_k)}{\sum_{l=1}^K \exp(f_{\theta_l}(x) + b_l)}, \tag{15}$$

where $b_k = \log(\rho_k) - \log Z(\theta_k)$.

Given this correspondence, we can modify FCE to do semi-supervised learning. Specifically, assume $\{(x_i, y_i), i = 1, ..., m\}$ are observed examples with labels known, and $\{x_i, i = m + 1, ..., m + n\}$ are observed unlabeled examples. For each category k, we can assume that class-conditional EBM is in the form

$$p_{\theta_k}(x) = \frac{1}{Z(\theta_k)} \exp[f_{\theta_k}(x)] = \exp[f_{\theta_k}(x) - c_k], \tag{16}$$

where $f_{\theta_k}(x)$ share all the weights except for the top layer. And we assume equal prior probability for each category. Let θ denotes all the parameters from class-conditional EBMs $\{\theta_k, k = 1, ..., K\}$. For labeled examples, we can maximize the conditional posterior probability of label y, given x and the fact that x is an observed example (instead of a generated example from q_α). By Bayes rule, this leads to maximizing the following objective function over θ:

$$\begin{aligned} L_{\text{label}}(\theta) &= \mathbb{E}_{p_{\text{data}}(x,y)} [\log p_\theta(y|x, y \in \{1, ..., K\})] \\ &= \mathbb{E}_{p_{\text{data}}(x,y)} \left[\log \frac{p_{\theta_y}(x)}{\sum_{k=1}^K p_{\theta_k}(x)}\right], \end{aligned} \tag{17}$$

which is similar to a classifier in the form.

For unlabeled examples, the probability can be defined by an unconditional EBM, which is in the form of a mixture model:

$$p_\theta(x) = \sum_{i=1}^K p_\theta(x|y = k)p(y = k) = \frac{1}{K} \sum_{i=1}^K p_{\theta_k}(x), \tag{18}$$

Together with the generated examples from $q_\alpha(x)$, we can define the same value function $V(\theta, \alpha)$ as eqn. 12 for the unlabeled examples. The joint estimation algorithm alternate the following two steps: (1) update θ by $\max_\theta L_{\text{label}}(\theta) + V(\theta, \alpha)$; (2) update α by $\min_\alpha V(\theta, \alpha)$. Due to the flexibility of EBM, $f_{\theta_k}(x)$ can be defined by any existing state-of-the-art network structures designed for semi-supervised learning.

4 Experiments

For FCE, we adaptively adjust the numbers of updates for EBM and Glow: we first update EBM for a few iterations until the classification accuracy is above 0.5, and then we update Glow until the classification accuracy is below 0.5. We use *Adam* [24] with learning rate $\alpha = 0.0003$ for the EBM and *Adamax* [24] with learning rate $\alpha = 0.00001$ for the Glow model.

4.1 Density estimation on 2D synthetic data

Figure 1 demonstrates the results of FCE on several 2D distributions, where FCE starts from a randomly initialized Glow. The learned EBM can fit multi-modal distributions accurately, and forms a better fit than Glow learned by either FCE or MLE. Notably, the EBM is defined by a much simpler network structure than Glow: for Glow we use 10 affine coupling layers, which amount to 30 fully-connected layers, while the energy-based model is defined by a 4-layer fully-connected network with the same width as Glow. Another interesting finding is that the EBM can fit the distributions well, even if the flow model is not a perfect contrastive distribution.

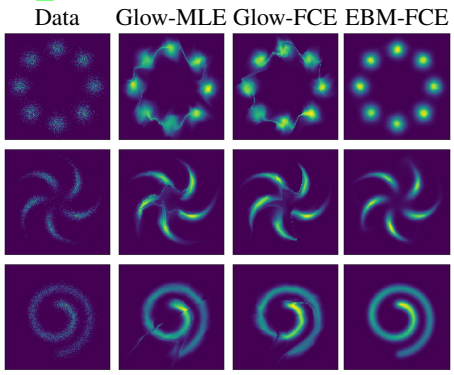

Figure 1: Comparison of trained EBM and Glow models on 2-dimensional data distributions.

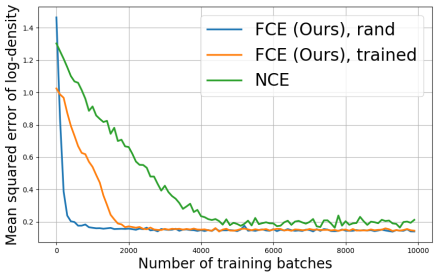

Figure 2: Density estimation accuracy in 2D examples of a mixture of 8 Gaussian distributions.

For the distribution depicted in the first row of Figure 1, which is a mixture of eight Gaussian distributions, we can compare the estimated densities by the learned models with the ground truth densities. Figure 2 shows the mean squared error of the estimated log-density over numbers of training iterations of EBMs. We show the results of FCE either starting from a randomly initialized Glow ('rand') or a Glow model pre-trained by MLE ('trained'), and compare with NCE with a Gaussian noise distribution. FCE starting from a randomly initialized Glow converges in fewer iterations. And both settings of FCE achieve a lower error rate than NCE.

4.2 Learning on real image datasets

We conduct experiments on the Street View House Numbers (SVHN) [46], CIFAR-10 [28] and CelebA [38] datasets. We resized the CelebA images to 32×32 pixels, and used $20,000$ images as a test set. We initialize FCE with a pre-trained Glow model, trained by MLE, for the sake of efficiency. We again emphasize the simplicity of the EBM model structure compared to Glow. See Supplementary A for detailed model architectures. For Glow, depth per level [25] is set as 8, 16, 32 for SVHN, CelebA and CIFAR-10 respectively. Figure 3 depicts synthesized examples from learned Glow models. To evaluate the fidelity of synthesized examples, Table 1 summarizes the Fréchet Inception Distance (FID) [18] of the synthesized examples computed with the Inception V3 [57] classifier. The fidelity is significantly improved compared to Glow trained by MLE (see Supplementary B for qualitative comparisons), and is competitive to the other generative models. In Table 2, we report the average negative log-likelihood (bits per dimension) on the testing sets. The log-likelihood of the learned EBM is based on the estimated normalizing constant (i.e., a parameter of the model) and should be taken with a grain of salt. For the learned Glow model, the log-likelihood

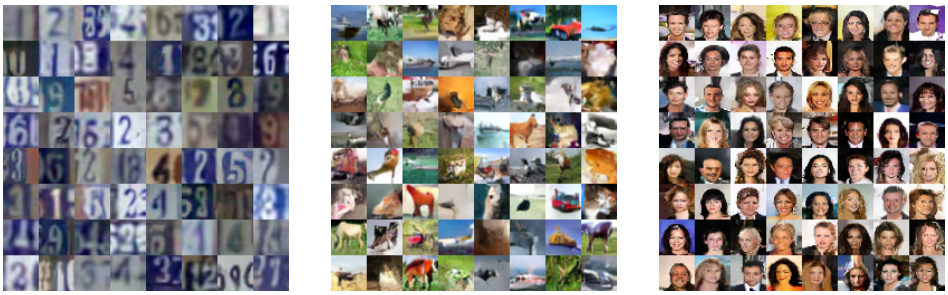

Figure 3: Synthesized examples from the Glow model learned by FCE. From left to right panels are from SVHN, CIFAR-10 and CelebA datasets, respectively. The image size is 32×32.

Table 1: FID scores for generated samples. For our method, we evaluate generative samples from the learned Glow model.

Method	SVHN	CIFAR-10	CelebA
VAE [27]	57.25	78.41	38.76
DCGAN [51]	21.40	37.70	12.50
Glow [25]	41.70	45.99	23.32
FCE (Ours)	**20.19**	**37.30**	**12.21**

of the Glow model estimated with FCE is slightly lower than the log-likelihood of the Glow model trained with MLE.

4.3 Unsupervised feature learning

To further explore the EBM learned with FCE, we perform unsupervised feature learning with features from a learned EBM. Specifically, we first conduct FCE on the entire training set of SVHN in an unsupervised way. Then, we extract the top layer feature maps from the learned EBM, and train a linear classifier on top of the extracted features using only a subset of the training images and their corresponding labels. Figure 4 shows the classification accuracy as a function of the number of labeled examples. Meanwhile, we compare our method with a supervised model with the same model structure as the EBM, and is trained only on the same subset of labeled examples each time. We observe that FCE outperforms the supervised model when the number of labeled examples is small (less than 2000).

Next we try to combine features from multiple layers together. Specifically, following the same procedure outlined in [51], the features from the top three convolutional layers are max pooled and concatenated to form a $14,336$-dimensional vector of feature. A regularized L2-SVM is then trained on these features with a subset of training examples and the corresponding labels. Table 3 summarizes the results of using $1,000$, $2,000$ and $4,000$ labeled examples from the training set. At the top part of the table, we compare with methods that estimate an EBM or a discriminative model coupled with a generator network. At the middle part of the table, we compare with methods that learn an EBM with contrastive divergence (CD) and modified versions of CD. For fair comparison, we use the same model structure for the EBMs or discriminative models used in all the methods. The results indicate that FCE outperforms these methods in terms of the effectiveness of learned features.

Table 2: Bits per dimension on testing data. † indicates that the log-likelihood is computed based on models with estimated normalizing constant, and should be taken with a grain of salt.

Model	SVHN	CIFAR-10	CelebA
Glow-MLE	2.17	3.35	3.49
Glow-FCE (Ours)	2.25	3.45	3.54
EBM-FCE (Ours)	†2.15	†3.27	†3.40

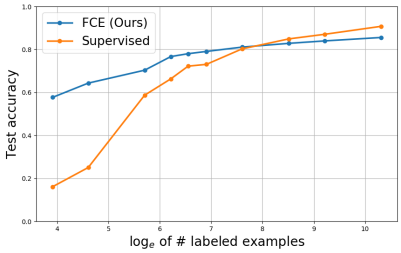

Figure 4: SVHN test-set classification accuracy as a function of number of labeled examples. The features from top layer feature maps are extracted and a linear classifier is learned on the extracted features.

Table 3: Test set classification error of L2-SVM classifier trained on the concatenated features learned from SVHN. DDGM stands for Deep Directed Generative Models. For fair comparison, all the energy-based models or discriminative models are trained with the same model structure.

Method	# of labeled data		
	1000	2000	4000
Wasserstein GAN [1]	43.15	38.00	32.56
DDGM [22]	44.99	34.26	27.44
DCGAN [51]	38.59	32.51	29.37
Persistent CD [58]	45.74	39.47	34.18
One-step CD [19]	44.38	35.87	30.45
Multigrid sampling [11]	30.23	26.54	22.83
FCE (Ours)	**27.07**	**24.12**	**22.05**

4.4 Semi-supervised learning

Recall that in section 3.4 we show that FCE can be generalized to perform semi-supervised learning. We emphasize that for semi-supervised learning, FCE not only learns a classification boundary or a posterior label distribution $p(y|x)$. Instead, the algorithm ends up with K estimated probabilistic distributions $p(x|y=k), k=1,...K$ for observed examples belonging to K categories respectively. Figure 5 illustrates this point by showing the learning process on a 2D example, where the data distribution consists of two twisted spirals belonging to two categories. Seven labeled points are provided for each category. As the training goes, the unconditional EBM $p_\theta(x)$ learns to capture all the modes of the data distribution, which is in the form of a mixture of class-conditional EBMs $p_{\theta_1}(x)$ and $p_{\theta_2}(x)$. Meanwhile, by maximizing the objective function $L_{\text{label}}(\theta)$ (eqn. 17), $p_\theta(x)$ is forced to project the learned modes into different spaces, resulting in two well-separated class-conditional

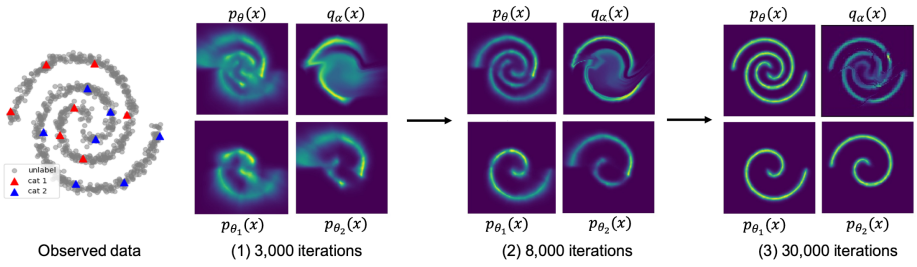

Figure 5: Illustration of FCE for semi-supervised learning on a 2D example, where the data distribution is two spirals belonging to two categories. Within each panel, the top left is the learned unconditional EBM. The top right is the learned Glow model. The bottom are two class-conditional EBMs. For observed data, seven labeled points are provided for each category.

Table 4: Semi-supervised classification error (%) on the SVHN test set, without data augmentation. † indicates that we derive the results by running the released code. The other cited results are provided by the original papers. Our results are averaged over three runs.

Method	# of labeled data	
	500	1000
SWWAE [64]		23.56
Skip DGM [39]		16.61 ($\pm$0.24)
Auxiliary DGM [39]		22.86
GAN with FM [54]	18.44 ($\pm$4.8)	8.11 ($\pm$1.3)
VAT-Conv-small [42]		6.83 ($\pm$0.24)
on Conv-small used in [54, 42]		
FCE-init	9.42 ($\pm$0.24)	8.50 ($\pm$0.26)
FCE	**7.05** ($\pm$0.28)	**6.35** ($\pm$0.12)
Π model [32]	7.05 ($\pm$0.30)	5.43 ($\pm$0.25)
VAT-Conv-large [42]	†8.98 ($\pm$0.26)	5.77 ($\pm$0.32)
on Conv-large used in [32, 42]		
FCE-init	8.86 ($\pm$0.26)	7.60 ($\pm$0.23)
FCE	6.86 ($\pm$0.18)	5.54 ($\pm$0.18)
FCE + VAT	**4.47** ($\pm$0.23)	**3.87** ($\pm$0.14)

EBMs. As shown in Figure 5, within a single mode of one category, the EBM tends to learn a smoothly connected cluster, which is often what we desire in semi-supervised learning.

Then we test the proposed method on a dataset of real images. Following the setting in [42], we use two types of CNN structures ('Conv-small'and 'Conv-large') for EBMs, which are commonly used in state-of-the-art semi-supervised learning methods. See Supplementary A for detailed model structures. We start FCE from a pre-trained Glow model. Before the joint training starts, EBMs are firstly trained for $50,000$ iterations with the Glow model fixed. In practice, this helps EBMs keep pace with the pre-trained Glow model, and equips EBMs with reasonable classification ability. We report the performance at this stage as 'FCE-init'. Also, since virtual adversarial training (VAT) [42] has been demonstrated as an effective regularization method for semi-supervised learning, we consider adopting it as an additional loss for learning the EBMs. More specifically, the loss is defined as the robustness of the conditional label distribution around each input data point against local purturbation. 'FCE + VAT' indicates the training with VAT.

Table 4 summarizes the results of semi-supervised learning on SVHN dataset without data augmentation. We report the mean error rates and standard deviations over three runs. All the methods listed in the table belong to the family of semi-supervised learning methods. Our method achieve competitive performance to these state-of-the-art methods. 'FCE + VAT' results show that the effectiveness of FCE does not overlap much with existing semi-supervised method, and thus they can be combined to further boost the performance.

5 Conclusion

This paper explores joint training of an energy-based model with a flow-based model, by combining the representational flexibility of the energy-based model and the computational tractability of the flow-based model. We may consider the learned energy-based model as the learned representation, while the learned flow-based model as the learned computation. This method can be considered as an adaptive version of noise contrastive estimation where the noise is transformed by a flow model to make its distribution closer to the data distribution and to make it a stronger contrast to the energy-based model. Meanwhile, the flow-based model is updated adaptively through the learning process, under the same adversarial value function.

In future work, we intend to generalize the joint training method by combining the energy-based model with other normalized probabilistic models, such as auto-regressive models. We also intend to

explore flow contrastive estimation of energy-based models with more interpretable energy functions, e.g., by incorporating sparsity constraints or latent variables into the energy function.

Acknowledgments

The work is partially supported by DARPA XAI project N66001-17-2-4029. We thank Pavel Sountsov, Alex Alemi and Srinivas Vasudevan for their helpful discussions.

References

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

[2] Paul Baltescu and Phil Blunsom. Pragmatic neural language modelling in machine translation. *arXiv preprint arXiv:1412.7119*, 2014.

[3] Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. *arXiv preprint arXiv:1811.00995*, 2018.

[4] Avishek Joey Bose, Huan Ling, and Yanshuai Cao. Adversarial contrastive estimation. *arXiv preprint arXiv:1805.03642*, 2018.

[5] Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating energy-based generative adversarial networks. *arXiv preprint arXiv:1702.01691*, 2017.

[6] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[7] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[8] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

[9] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

[10] Charles W Fox and Stephen J Roberts. A tutorial on variational bayesian inference. *Artificial intelligence review*, 38(2):85–95, 2012.

[11] Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning generative convnets via multi-grid modeling and sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9155–9164, 2018.

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

[13] Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[14] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[15] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

[16] Tian Han, Erik Nijkamp, Xiaolin Fang, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Divergence triangle for joint training of generator model, energy-based model, and inference model. *arXiv preprint arXiv:1812.10907*, 2018.

[17] Tianxing He, Yu Zhang, Jasha Droppo, and Kai Yu. On training bi-directional neural network language model with noise contrastive estimation. In *2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 1–5. IEEE, 2016.

[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[19] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[20] Long Jin, Justin Lazarow, and Zhuowen Tu. Introspective classification with convolutional nets. In *Advances in Neural Information Processing Systems*, pages 823–833, 2017.

[21] Ilyes Khemakhem, Diederik P Kingma, and Aapo Hyvärinen. Variational autoencoders and nonlinear ica: A unifying framework. *arXiv preprint arXiv:1907.04809*, 2019.

[22] Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*, 2016.

[23] Diederik Kingma and Max Welling. In *International Conference on Machine Learning*, pages 1782–1790, 2014.

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

[25] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

[26] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

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

[28] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[30] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. Videoflow: A flow-based generative model for video. *arXiv preprint arXiv:1903.01434*, 2019.

[31] Rithesh Kumar, Anirudh Goyal, Aaron Courville, and Yoshua Bengio. Maximum entropy generators for energy-based models. *arXiv preprint arXiv:1901.08508*, 2019.

[32] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[33] Justin Lazarow, Long Jin, and Zhuowen Tu. Introspective neural networks for generative modeling. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2774–2783, 2017.

[34] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[35] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[36] Kwonjoon Lee, Weijian Xu, Fan Fan, and Zhuowen Tu. Wasserstein introspective neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3702–3711, 2018.

[37] Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

[38] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15:2018, 2018.

[39] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.

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

[41] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[42] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

[43] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273, 2013.

[44] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

[45] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[46] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[47] Jiquan Ngiam, Zhenghao Chen, Pang W Koh, and Andrew Y Ng. Learning deep energy models. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1105–1112, 2011.

[48] Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. On learning non-convergent short-run mcmc toward energy-based model. *arXiv preprint arXiv:1904.09770*, 2019.

[49] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[50] Youssef Oualil and Dietrich Klakow. A batch noise contrastive estimation approach for training large vocabulary language models. *arXiv preprint arXiv:1708.05997*, 2017.

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

[52] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[53] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[54] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[55] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[56] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016.

[57] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[58] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM, 2008.

[59] Dustin Tran, Keyon Vafa, Kumar Krishna Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. *arXiv preprint arXiv:1905.10347*, 2019.

[60] Zhuowen Tu. Learning generative models via discriminative approaches. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[61] Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, 2013.

[62] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. *arXiv preprint arXiv:1511.06361*, 2015.

[63] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pages 2635–2644, 2016.

[64] Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015.

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

[66] Song Chun Zhu and David Mumford. Grade: Gibbs reaction and diffusion equations. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 847–854. IEEE, 1998.

A Model architectures

Table 5 summarizes the EBM architectures used in unsupervised learning (subsections 4.1-4.3). The slope of all leaky ReLU (lReLU) [40] functions are set to 0.2. For semi-supervised learning from a 2D example (subsection 4.4), we use the same EBM structure as the one used in unsupervised learning from 2D examples, except that for the top fully connect layer, we change the number of

output channels to 2, to model EBMs of two categories respectively. Table 6 summarizes the EBM architectures used in semi-supervised learning from SVHN (subsection 4.4). After each convolutional layer, a weight normalization [56] layer and a leaky ReLU layer is added. The slope of leaky ReLU functions is set to 0.2. A weight normalization layer is added after the top fully connected layer.

Table 5: EBM architectures used in unsupervised learning

2D data	SVHN / CIFAR-10
fc. 128 lReLU	4×4 conv. 64 lReLU, stride 2
fc. 128 lReLU	4×4 conv. 128 lReLU, stride 2
fc. 128 lReLU	4×4 conv. 256 lReLU, stride 2
fc. 1	4×4 conv. 1, stride 1

Table 6: EBM architectures used in semi-supervised learning from SVHN

Conv-small	Conv-large
dropout, $p = 0.2$	
3×3 conv. 64, stride 1	3×3 conv. 128, stride 1
3×3 conv. 64, stride 1	3×3 conv. 128, stride 1
3×3 conv. 64, stride 2	3×3 conv. 128, stride 2
dropout, $p = 0.5$	
3×3 conv. 128, stride 1	3×3 conv. 256, stride 1
3×3 conv. 128, stride 1	3×3 conv. 256, stride 1
3×3 conv. 128, stride 2	3×3 conv. 256, stride 2
dropout, $p = 0.5$	
3×3 conv. 128, stride 1	3×3 conv. 512, stride 1
1×1 conv. 128, stride 1	1×1 conv. 256, stride 1
1×1 conv. 128, stride 1	1×1 conv. 128, stride 1
global max pool, $6 \times 6 \rightarrow 1 \times 1$	
fc. $128 \rightarrow 10$	

For Glow model, we follow the setting of [25]. The architecture has multi-scales with levels L. Within each level, there are K flow blocks. Each block has three convolutional layers (or fully-connected layers) with a width of W channels. After the first two layers, a ReLU activation is added. Table 7 summarizes the hyperparameters for different datasets.

Table 7: Hyperparameters for Glow model architectures

Dataset	Levels L	Blocks per level K	Width W	Layer type	Coupling
2D data	1	10	128	fc	affine
SVHN	3	8	512	conv	additive
CelebA	3	16	512	conv	additive
CIFAR-10	3	32	512	conv	additive

B Synthesis comparison

In figures 6, 7 and 8, we display the synthesized examples from Glow trained by MLE and our FCE.

Figure 6: Synthesized examples from Glow models learned from SVHN. Left panel is by MLE. Right panel is by our FCE.

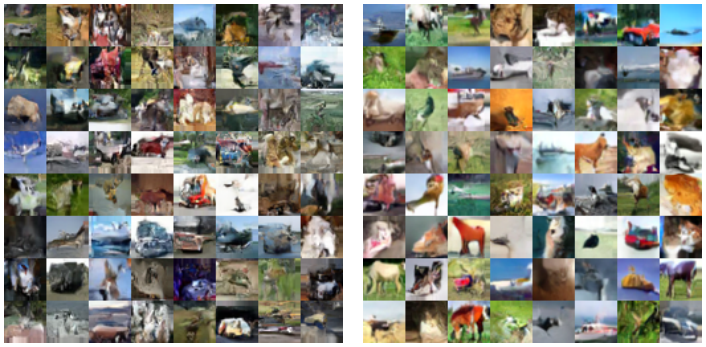

Figure 7: Synthesized examples from Glow models learned from CIFAR-10. Left panel is by MLE. Right panel is by our FCE.

Figure 8: Synthesized examples from Glow models learned from CelebA. Left panel is by MLE. Right panel is by our FCE.