The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Cooperative Learning of Energy-Based Model and Latent Variable Model via MCMC Teaching

Jianwen Xie,[1,2] **Yang Lu,**[1,3] **Ruiqi Gao,**[1] **Ying Nian Wu**[1]
[1]Department of Statistics, University of California, Los Angeles, USA
[2]Hikvision Research America
[3]Amazon RSML (Retail System Machine Learning) Group

Abstract

This paper proposes a cooperative learning algorithm to train both the undirected energy-based model and the directed latent variable model jointly. The learning algorithm interweaves the maximum likelihood algorithms for learning the two models, and each iteration consists of the following two steps: (1) Modified contrastive divergence for energy-based model: The learning of the energy-based model is based on the contrastive divergence, but the finite-step MCMC sampling of the model is initialized from the synthesized examples generated by the latent variable model instead of being initialized from the observed examples. (2) MCMC teaching of the latent variable model: The learning of the latent variable model is based on how the MCMC in (1) changes the initial synthesized examples generated by the latent variable model, where the latent variables that generate the initial synthesized examples are known so that the learning is essentially supervised. Our experiments show that the cooperative learning algorithm can learn realistic models of images.

1 Introduction

1.1 Student and teacher nets

We begin with an analogy. A student writes up an initial draft of a paper. Her advisor then revises it. After that they submit the revised paper for review. The student then learns from her advisor's revision, while the advisor learns from the outside review. In this analogy, the advisor guides the student, but the student does most of the work.

This paper is about cooperative learning of two probabilistic generative models of signals such as images. These two models play the roles of student and teacher as described above. The first model is an energy-based model (LeCun et al. 2006) or undirected graphical model or Markov random field model (Zhu, Wu, and Mumford 1997) that plays the role of the teacher. The second model is a latent variable model or directed graphical model that plays the role of the student. In this paper, we focus on models that are parametrized by convolutional neural networks (ConvNets or CNNs)(LeCun et al., 1998; Krizhevsky, Sutskever, and Hinton, 2012), as illustrated by (1). Specifically, in the latent variable model, the mapping from the latent variables to the signal is parameterized by a top-down ConvNet (Dosovitskiy, Springenberg, and Brox, 2015). This is the well-known **generator** network (Goodfellow et al. 2014). In the energy-based model, the energy function is parametrized by a bottom-up ConvNet that maps the signal to the energy (Ngiam et al. 2011), (Xie et al. 2016). For ease of reference, we call it the **descriptor** network, following the terminology of (Zhu 2003).

$$\begin{array}{cc} \text{Bottom-up ConvNet} & \text{Top-down ConvNet} \\ \textbf{energy} & \textbf{latent variables} \\ \Uparrow & \Downarrow \\ \text{signal} & \text{signal} \\ \text{(a) Descriptor Net} & \text{(b) Generator Net} \\ \text{(teacher)} & \text{(student)} \end{array} \tag{1}$$

The likelihoods of both models involve intractable integrals, and the gradients of both log-likelihoods involve intractable expectations that have to be approximated by expensive Markov chain Monte Carlo (MCMC). We notice that the maximum likelihood algorithms for learning the two models can be interwoven into a cooperative learning algorithm in order to speed up the learning of both models. Each iteration of the cooperative learning algorithm consists of the following two steps: (1) **Modified contrastive divergence** for energy-based model (descriptor): The learning of the energy-based model is based on the contrastive divergence (Hinton 2002), but the finite-step MCMC sampling of the model is initialized from the synthesized examples generated by the latent variable model instead of being initialized from the observed examples. Within each iteration of the modified contrastive divergence, the latent variable model supplies a fresh and independent batch of synthesized examples to initialize the MCMC sampling of the energy-based model. (2) **MCMC teaching** of the latent variable model (generator): The learning of the latent variable model is based on how the MCMC in (1) changes the initial synthesized examples generated by the latent variable model. That is, the energy-based model (teacher) distills its knowledge to the latent variable model (student) via MCMC, and we call it MCMC teaching. In MCMC teaching, the latent variables that generate the initial synthesized examples are known so that the learning is essentially supervised. Our experiments show that the cooperative learning algorithm can learn realistic models of images.

1.2 Motivations, advantages and contributions

The main motivation for our work is that we find it very challenging to learn the two models separately, when the

training images are highly varied. We find it much easier for the cooperative algorithm to learn realistic models from such data. Another motivation is to develop an alternative system to the generative adversarial networks (GAN) (Goodfellow et al., 2014; Denton et al., 2015; Radford, Metz, and Chintala, 2015), where in our system both models are learned generatively by maximum likelihood. Our experiences suggest that the cooperative learning is stable and does not encounter mode collapsing issue.

The advantages of the cooperative learning versus separate learning are as follows. (1) The generator (latent variable model) jump-starts the finite-step MCMC of the descriptor (energy-based model) by supplying fresh and independent examples via ancestral sampling in each iteration. Thus the generator serves as a direct approximate sampler of the descriptor. (2) The generator learns from how the finite-step MCMC changes the synthesized examples it generates, where the values of the latent variables are known, thus they do not need to be inferred (or can be easily inferred), and the learning is much easier than learning from the observed examples, for which it is difficult to infer the latent variables. In MCMC teaching in (2), the generator model seeks to approximate the descriptor model, so that the modified contrastive divergence in (1) is close to maximum likelihood. With repeated teaching by finite-step MCMC in (2), the generator accumulates the MCMC transitions and reproduces them by direct ancestral sampling.

The contributions of our work are as follows. We propose a cooperative learning algorithm to train the energy-based model (descriptor) and the latent variable model (generator) simultaneously. Our work connects the undirected model (descriptor) and the directed model (generator). It also connects the ancestral sampling (generator) and the MCMC sampling (descriptor).

2 Related work

Our work is related to the contrastive divergence (Hinton 2002) for training the energy-based model. The contrastive divergence initializes the finite-step MCMC sampling from the observed examples. Our method initializes the MCMC sampling from the generator that seeks to approximate the descriptor, so that the learning is closer to maximum likelihood.

Our work is similar to the recent work of (Kim and Bengio 2016). In fact, the settings of the two nets are the same. In (Kim and Bengio 2016), the generator learns from the descriptor by minimizing the Kullback-Leibler divergence from the generator to the descriptor, which can be decomposed into an energy term and an entropy term. In our work, the descriptor teaches the generator via MCMC teaching. Our method does not need to approximate the intractable entropy term.

Another method for training the generator network is variational auto-encoder (VAE) (Kingma and Welling 2014; Rezende, Mohamed, and Wierstra 2014; Mnih and Gregor 2014), which learns an inferential or recognition network. The MCMC teaching in our work avoids the challenging problem of inferring the latent variables from the observed examples. Whereas the learned inferential model in VAE serves as a direct and approximate sampler of the posterior distribution of the latent variables, in the cooperative learning, the learned generator model serves as a direct and approximate sampler of the energy-based descriptor model.

The connection between the descriptor net and the discriminator net has been explored by (Xie et al. 2016), where the descriptor can be derived from the discriminator.

Our work appears to be related to knowledge distilling (Hinton, Vinyals, and Dean 2015). In our work, the descriptor distills its knowledge to the generator through MCMC teaching, but the distillation is done in an on-line fashion.

Our work bears similarity to the co-training method of (Blum and Mitchell 1998). The two learners in our work are of different directions, and they feed each other the synthetic data, instead of class labels.

3 Two models and their maximum likelihood learning algorithms

(p_θ, q_α) notation. Let Y be the D-dimensional signal, such as an image. We use $p(Y;\theta)$ or p_θ to denote the probability distribution of the descriptor net (energy-based model), where θ denotes the parameters of the bottom-up ConvNet. We use $q(Y;\alpha)$ or q_α to denote the probability distribution of the generator net (latent variable model), where α denotes the parameters of the top-down ConvNet.

3.1 Energy-based model and maximum likelihood learning

The descriptor model is in the form of exponential tilting of a reference distribution (Xie et al. 2016):

$$p(Y;\theta) = \frac{1}{Z(\theta)} \exp\left[f(Y;\theta)\right] p_0(Y), \tag{2}$$

where $p_0(Y)$ is the reference distribution such as Gaussian white noise $p_0(Y) \propto \exp\left(-\|Y\|^2/2s^2\right)$ (or a uniform distribution), $f(Y;\theta)$ is defined by a bottom-up ConvNet whose parameters are denoted by θ. The energy function is $\mathcal{E}(Y;\theta) = \|Y\|^2/(2s^2) - f(Y;\theta)$. See the diagram in (1). $Z(\theta) = \int \exp\left[f(Y;\theta)\right] p_0(Y) dY$ is the normalizing constant that is analytically intractable.

Suppose we observe training examples $\{Y_i, i = 1, ..., n\}$ from an unknown data distribution $P_{\rm data}(Y)$. The maximum likelihood learning seeks to maximize the log-likelihood function $L_p(\theta) = \frac{1}{n}\sum_{i=1}^n \log p(Y_i;\theta)$. If the sample size n is large, the maximum likelihood estimator minimizes $\mathrm{KL}(P_{\rm data}|p_\theta)$, the Kullback-Leibler divergence from the data distribution $P_{\rm data}$ to the model distribution p_θ. The gradient of $L_p(\theta)$ is

$$L'_p(\theta) = \frac{1}{n}\sum_{i=1}^n \frac{\partial}{\partial\theta} f(Y_i;\theta) - \mathrm{E}_\theta\left[\frac{\partial}{\partial\theta} f(Y;\theta)\right], \tag{3}$$

where E_θ denotes the expectation with respect to $p(Y;\theta)$. The key to the above identity is that $\frac{\partial}{\partial\theta}\log Z(\theta) = \mathrm{E}_\theta[\frac{\partial}{\partial\theta} f(Y;\theta)]$.

The expectation in equation (3) is analytically intractable and has to be approximated by MCMC, such as Langevin

dynamics, which iterates the following step:

$$Y_{\tau+1} = Y_\tau - \frac{\delta^2}{2}\left[\frac{Y_\tau}{s^2} - \frac{\partial}{\partial Y} f(Y_\tau;\theta)\right] + \delta U_\tau, \quad (4)$$

where τ indexes the time steps of the Langevin dynamics, δ is the step size, and $U_\tau \sim \mathrm{N}(0, I_D)$ is the Gaussian white noise term. A Metropolis-Hastings step can be added to correct for the finite δ.

We can run $\tilde{n}$ parallel chains of Langevin dynamics according to (4) to obtain the synthesized examples $\{\tilde{Y}_i, i = 1, ..., \tilde{n}\}$. The Monte Carlo approximation to $L'_p(\theta)$ is

$$L'_p(\theta) \approx \frac{1}{n}\sum_{i=1}^{n} \frac{\partial}{\partial\theta} f(Y_i;\theta) - \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} \frac{\partial}{\partial\theta} f(\tilde{Y}_i;\theta), \quad (5)$$

which is used to update θ. We call the learning algorithm for the descriptor model the algorithm D.

Algorithm D (Xie et al. 2016) iterates the following two steps after initializing θ and $\{\tilde{Y}_i, i = 1, ..., \tilde{n}\}$. **Step D1**: Run l_p steps of Langevin from the current $\{\tilde{Y}_i\}$ according to (4). **Step D2**: update $\theta^{(t+1)} = \theta^{(t)} + \gamma_t L'_p(\theta^{(t)})$ with learning rate γ_t. The convergence of such an algorithm is studied by (Younes 1999).

Because the parameter θ keeps changing in the learning process, the energy landscape and the local energy minima also keep changing. This may help the Langevin dynamics avoid being trapped by the local energy minima.

Contrastive divergence. If we initialize the synthesized examples $\{\tilde{Y}_i\}$ from the observed examples $\{Y_i\}$, the learning algorithm becomes persistent contrastive divergence (Tieleman 2008).

3.2 Latent variable model and maximum likelihood learning

The generator net (Goodfellow et al. 2014) has its root in factor analysis in statistics and in latent variable model or directed graphical model in machine learning. The generator net seeks to explain the signal Y of dimension D by a vector of latent variables X of dimension d, and usually $d \ll D$. The model is of the following form:

$$X \sim \mathrm{N}(0, I_d), Y = g(X;\alpha) + \epsilon,\ \epsilon \sim \mathrm{N}(0, \sigma^2 I_D). \quad (6)$$

$g(X;\alpha)$ is a top-down ConvNet defined by the parameters α. The ConvNet g maps the latent variables X to the signal Y. See the diagram in (1).

Model (6) is a directed graphical model, where Y can be readily generated by first sampling X from its known prior distribution $\mathrm{N}(0, I_d)$ and then transforming X to Y via g. The joint density of model (6) is $q(X, Y;\alpha) = q(X)q(Y|X;\alpha)$, and

$$\log q(X, Y;\alpha) = -\frac{1}{2\sigma^2}\|Y - g(X;\alpha)\|^2 - \frac{1}{2}\|X\|^2 + \text{constant}, \quad (7)$$

where the constant term is independent of X, Y and α. The marginal density is obtained by integrating out the latent variables X, i.e., $q(Y;\alpha) = \int q(X, Y;\alpha)dX$, which is analytically intractable. The inference of X given Y is based on the posterior density $q(X|Y;\alpha) = q(X, Y;\alpha)/q(Y;\alpha) \propto q(X, Y;\alpha)$ as a function of X.

For the training data $\{Y_i, i = 1, ..., n\}$, the generator net can be trained by maximizing the log-likelihood $L_q(\alpha) = \frac{1}{n}\sum_{i=1}^{n} \log q(Y_i;\alpha)$. For large sample, the learned α minimizes the Kullback-Leibler divergence $\mathrm{KL}(P_{\mathrm{data}}|q_\alpha)$ from the data distribution P_{data} to the model distribution q_α. The gradient of $L_q(\alpha)$ is obtained according to the following identity

$$\begin{aligned}\frac{\partial}{\partial\alpha}\log q(Y;\alpha) &= \frac{1}{q(Y;\alpha)}\frac{\partial}{\partial\alpha}\int q(Y, X;\alpha)dX \\ &= \mathrm{E}_{q(X|Y;\alpha)}\left[\frac{\partial}{\partial\alpha}\log q(X, Y;\alpha)\right] \quad (8)\end{aligned}$$

which underlies the EM algorithm. The usefulness of identify (8) lies in the fact that the derivative of the complete-data log-likelihood $\log q(X, Y;\alpha)$ on the right hand side can be obtained in closed form. However, the expectation in (8) is analytically intractable, and has to be approximated by MCMC that samples from the posterior $q(X|Y;\alpha)$, such as Langevin dynamics, which iterates

$$X_{\tau+1} = X_\tau + \frac{\delta^2}{2}\frac{\partial}{\partial X}\log q(X_\tau, Y;\alpha) + \delta U_\tau, \quad (9)$$

where $U_\tau \sim \mathrm{N}(0, I_d)$. With X_i sampled from $q(X_i \mid Y_i, \alpha)$ for each observation Y_i, the Monte Carlo approximation to $L'_q(\alpha)$ is

$$\begin{aligned}L'_q(\alpha) &\approx \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial\alpha}\log q(X_i, Y_i;\alpha) \\ &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\sigma^2}(Y_i - g(X_i;\alpha))\frac{\partial}{\partial\alpha}g(X_i;\alpha), \quad (10)\end{aligned}$$

which is used to update α. We call the learning algorithm for the generator model the algorithm G.

Algorithm G (Han et al. 2017) iterates the following two steps after initializing α and $\{X_i, i = 1, ..., n\}$. **Step G1**: run l_q steps of Langevin from the current $\{X_i\}$ according to (9). **Step G2**: update $\alpha^{(t+1)} = \alpha^{(t)} + \gamma_t L'_q(\alpha^{(t)})$ with learning rate γ_t. The convergence of such an algorithm is studied by (Younes 1999).

4 Cooperative learning

4.1 Learning algorithm

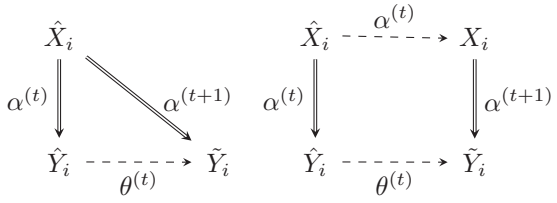
In Algorithms D and G, both steps D1 and G1 require MCMC, which can be time consuming. We notice that the two algorithms can cooperate with each other to speed up the learning of both models by jump-starting each other's MCMC sampling. In the resulting cooperative learning algorithm, the generator serves as an approximated sampler of the descriptor. Meanwhile the generator seeks to absorb and accumulate the MCMC transitions for sampling the descriptor in order to reproduce them by one step direct ancestral sampling.

Specifically, in Step D1, we can initialize the synthesized examples by generating examples from the generator

net. We first generate $\hat{X}_i \sim \mathrm{N}(0, I_d)$, and then generate $\hat{Y}_i = g(\hat{X}_i; \alpha) + \epsilon_i$, for $i = 1, ..., \tilde{n}$. If the current generator q_α is close to the current descriptor p_θ, then the generated $\{\hat{Y}_i\}$ should be a good initialization for sampling from the descriptor net, i.e., starting from the $\{\hat{Y}_i, i = 1, ..., \tilde{n}\}$, we run Langevin dynamics in Step D1 for l_p steps to get $\{\tilde{Y}_i, i = 1, ..., \tilde{n}\}$, which are revised versions of $\{\hat{Y}_i\}$. These $\{\tilde{Y}_i\}$ can be used as the synthesized examples from the descriptor p_θ. We can then update θ according to Step D2 of Algorithm D. It is important to notice that in each learning iteration, the generator supplies a fresh independent batch of $\{\hat{Y}_i, i = 1, ..., \tilde{n}\}$ to initialize the MCMC of the descriptor. This amounts to running infinite many parallel chains of MCMC for sampling from the descriptor.

In order to update α of the generator q_α, we treat the $\{\tilde{Y}_i, i = 1, ..., \tilde{n}\}$ produced by the above Step D1 as the training data for the generator. Since these $\{\tilde{Y}_i\}$ are obtained by the finite-step MCMC initialized from the $\{\hat{Y}_i, i = 1, ..., \tilde{n}\}$ produced by the generator net with *known* latent variables $\{\hat{X}_i, i = 1, ..., \tilde{n}\}$, we can update α by learning from $\{(\tilde{Y}_i, \hat{X}_i), i = 1, ..., \tilde{n}\}$, which is a supervised learning problem, or more specifically, a non-linear regression of $\tilde{Y}_i$ on $\hat{X}_i$. At $\alpha^{(t)}$, the known $\hat{X}_i$ generates and thus reconstructs the initial example $\hat{Y}_i$. After updating α, we want $\hat{X}_i$ to reconstruct the revised example $\tilde{Y}_i$. That is, we revise α to absorb the revision from $\hat{Y}_i$ to $\tilde{Y}_i$, so that the generator q_α shifts its density from $\{\hat{Y}_i\}$ to $\{\tilde{Y}_i\}$, and reproduces the effect of all the past MCMC transitions by one step direct ancestral sampling.

The left diagram in (11) illustrates the basic idea.



(11)

In the two diagrams in (11), the double-line arrows indicate generation and reconstruction by the generator q_α, while the dashed-line arrows indicate Langevin dynamics in the two nets. The diagram on the right in (11) illustrates a more rigorous method, where we initialize the Langevin dynamics for infering $\{X_i, i = 1, ..., \tilde{n}\}$ in Step G1 from $\{\hat{X}_i\}$, and then update α in Step G2 based on $\{(\tilde{Y}_i, X_i), i = 1, ..., \tilde{n}\}$. Since $\hat{X}_i$ is expected to be close to the posterior mode of X_i, the convergence of the Langevin inference initialized from $\hat{X}_i$ should be very fast. If the residual variance σ^2 of the generator model is assumed to be small, then the Langevin inference process is close to gradient descent on the reconstruction error. The diagram on the right shows how the two nets jump-start each other's MCMC. The generator jump-starts the Langevin dynamics for sampling from the descriptor, while the descriptor jump-starts the Langevin dynamics for inferring the latent variables of the generator. It is important to notice that by letting the generator learn from the synthesized data, the latent variables are essentially known, so that the learning becomes a much simpler supervised learning problem.

Algorithm 1 CoopNets Algorithm

Input:

(1) training examples $\{Y_i, i = 1, ..., n\}$, (2) numbers of Langevin steps l_p ad l_q, (3) number of learning iterations T

Output:

(1) estimated parameters θ and α, (2) synthetic examples $\{\hat{Y}_i, \tilde{Y}_i, i = 1, ..., \tilde{n}\}$

Let $t \leftarrow 0$, initialize θ and α.

repeat

  Step G0: For $i = 1, ..., \tilde{n}$, generate $\hat{X}_i \sim \mathrm{N}(0, I_d)$, and generate $\hat{Y}_i = g(\hat{X}_i; \alpha^{(t)}) + \epsilon_i$.

  Step D1: For $i = 1, ..., \tilde{n}$, starting from $\hat{Y}_i$, Run l_p steps of Langevin revision dynamics to obtain $\tilde{Y}_i$, each step following equation (4).

  Step G1: Treat the current $\{\tilde{Y}_i, i = 1, ..., \tilde{n}\}$ as the training data, for each i, infer $X_i = \hat{X}_i$. Or more rigorously, starting from $X_i = \hat{X}_i$, run l_q steps of Langevin inference dynamics to update X_i, each step following equation (9).

  Step D2: Update $\theta^{(t+1)} = \theta^{(t)} + \gamma_t L_p'(\theta^{(t)})$, where $L_p'(\theta^{(t)})$ is computed according to (5).

  Step G2: Update $\alpha^{(t+1)} = \alpha^{(t)} + \gamma_t {L_q}'(\alpha^{(t)})$, where ${L_q}'(\alpha^{(t)})$ is computed according to equation (10), except that Y_i is replaced by $\tilde{Y}_i$, and n by $\tilde{n}$. We can run multiple iterations of Step G2 to learn from and reconstruct $\{\tilde{Y}_i\}$, and to allow the generator to catch up with the descriptor.

  Let $t \leftarrow t + 1$

until $t = T$

Algorithm 1 describes the cooperative learning algorithm that we call the **CoopNets algorithm** for ease of reference. It interweaves Algorithm D and Algorithm G.

4.2 Theoretical understanding

In the CoopNets algorithm, Steps G0, D1, and D2 are modified contrastive divergence, and Steps G0, G1, and G2 are MCMC teaching.

(1) Modified contrastive divergence for the descriptor (energy-based model). In the traditional contrastive divergence (Hinton 2002), $\hat{Y}_i$ in Step D1 is taken to be the observed Y_i. In cooperative learning, $\hat{Y}_i$ is generated by $q(Y; \alpha^{(t)})$. Let $\mathcal{M}_\theta$ be the Markov transition kernel of l_p steps of Langevin dynamics that samples p_θ. For any distribution p and any Markov transition kernel $\mathcal{M}$, let $\mathcal{M}p$ be the marginal distribution obtained by running the Markov transition $\mathcal{M}$ from p. Then similar to the traditional contrastive divergence, the learning gradient of the descriptor θ at iteration t is the gradient of $\mathrm{KL}(P_{\mathrm{data}}|p_\theta) - \mathrm{KL}(\mathcal{M}_{\theta^{(t)}} q_{\alpha^{(t)}}|p_\theta)$ with respect to θ. In the traditional contrastive divergence, P_{data} takes the place of $q_{\alpha^{(t)}}$ in the second KL-divergence.

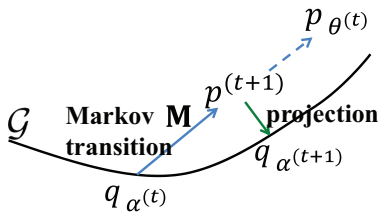


Figure 1: The MCMC teaching of the generator alternates between Markov transition and projection. The family of the generator models $\mathcal{G}$ is illustrated by the black curve. Each distribution is illustrated by a point.

(2) MCMC teaching of the generator (latent variable model). The learning gradient of the generator α in the right diagram of (11) is the gradient of $\mathrm{KL}(\mathcal{M}_{\theta^{(t)}} q_{\alpha^{(t)}} | q_\alpha)$ with respect to α. Here $p^{(t+1)} = \mathcal{M}_{\theta^{(t)}} q_{\alpha^{(t)}}$ takes the place of P_{data} as the data to train the generator model. It is much easier to minimize $\mathrm{KL}(\mathcal{M}_{\theta^{(t)}} q_{\alpha^{(t)}} | q_\alpha)$ than minimizing $\mathrm{KL}(P_{\text{data}} | q_\alpha)$ because the latent variables are essentially known in the former, so that the learning is supervised. The MCMC teaching alternates between Markov transition from $q_{\alpha^{(t)}}$ to $p^{(t+1)}$, and projection from $p^{(t+1)}$ to $q_{\alpha^{(t+1)}}$, as illustrated by Figure 1.

Assume the learning algorithm converges to a fixed point $(\hat{\theta}, \hat{\alpha})$, then

$$\hat{\theta} = \arg\min_\theta \left[\mathrm{KL}(P_{\text{data}} | p_\theta) - \mathrm{KL}(\mathcal{M}_{\hat{\theta}} q_{\hat{\alpha}} | p_\theta) \right], \quad (12)$$

$$\hat{\alpha} = \arg\min_\alpha \mathrm{KL}(\mathcal{M}_{\hat{\theta}} q_{\hat{\alpha}} | q_\alpha). \quad (13)$$

(assuming $\hat{\theta}$ and $\hat{\alpha}$ are local minima). Equation (13) tells us that $q_{\hat{\alpha}}$ seeks to be the stationary distribution of $\mathcal{M}_{\hat{\theta}}$, and the stationary distribution is nothing but $p_{\hat{\theta}}$. In the idealized scenario where the generator q_α has infinite capacity, so that $\min_\alpha \mathrm{KL}(\mathcal{M}_{\hat{\theta}} q_{\hat{\alpha}} | q_\alpha) = 0$, then $q_{\hat{\alpha}} = \mathcal{M}_{\hat{\theta}} q_{\hat{\alpha}}$, so that $q_{\hat{\alpha}}$ is the stationary distribution of $\mathcal{M}_{\hat{\theta}}$, which is $p_{\hat{\theta}}$, thus $q_{\hat{\alpha}} = p_{\hat{\theta}}$. As a consequence, the second divergence in (12) vanishes, i.e., $\mathrm{KL}(\mathcal{M}_{\hat{\theta}} q_{\hat{\alpha}} | p_{\hat{\theta}}) = 0$, so that $\hat{\theta}$ becomes the maximum likelihood estimate that minimizes the first KL-divergence $\mathrm{KL}(P_{\text{data}} | p_\theta)$.

To further understand the dynamics of MCMC teaching in this idealized scenario, suppose the descriptor p_θ is fixed, and it teaches the generator q_α by MCMC teaching such that $\alpha^{(t+1)} = \arg\min_\alpha \mathrm{KL}(\mathcal{M}_\theta q_{\alpha^{(t)}} | q_\alpha)$, then $q_{\alpha^{(t+1)}} = \mathcal{M}_\theta q_{\alpha^{(t)}}$, so that $q_{\alpha^{(t)}} = \mathcal{M}_\theta^t q_{\alpha^{(0)}} \to p_\theta$, i.e., q_α accumulates the MCMC transitions and convergences to the stationary distribution p_θ.

As is the case with the traditional contrastive divergence, the analysis of the finite capacity situation can be rather involved. We leave it to future investigation, while relying on empirical evaluations in this paper.

(Kim and Bengio 2016) learned the generator model by gradient descent on $\mathrm{KL}(q_\alpha | p_{\theta^{(t)}})$ over α. The objective function is $\mathrm{KL}(q_\alpha | p_{\theta^{(t)}}) = \mathrm{E}_{q_\alpha}[\log q(Y; \alpha)] - \mathrm{E}_{q_\alpha}[\log p(Y; \theta^{(t)})]$, where the first term is the negative entropy that is intractable, and the second term is the expected energy that is tractable.

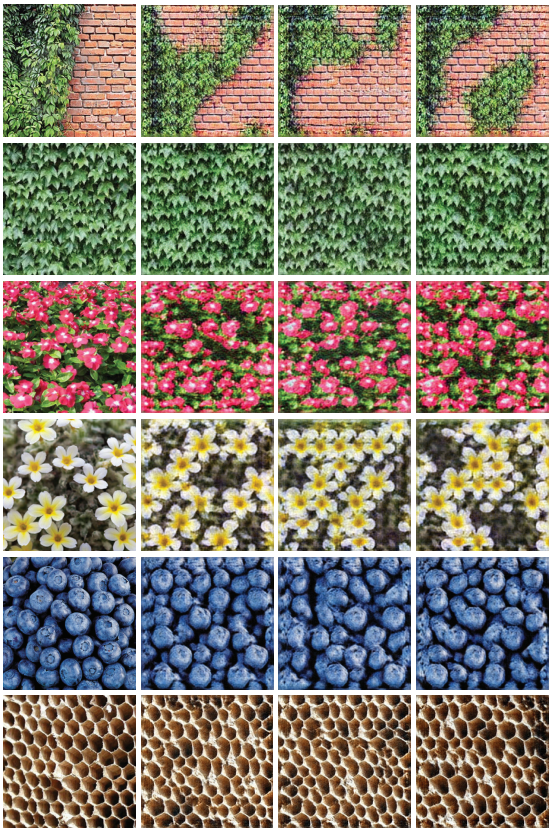


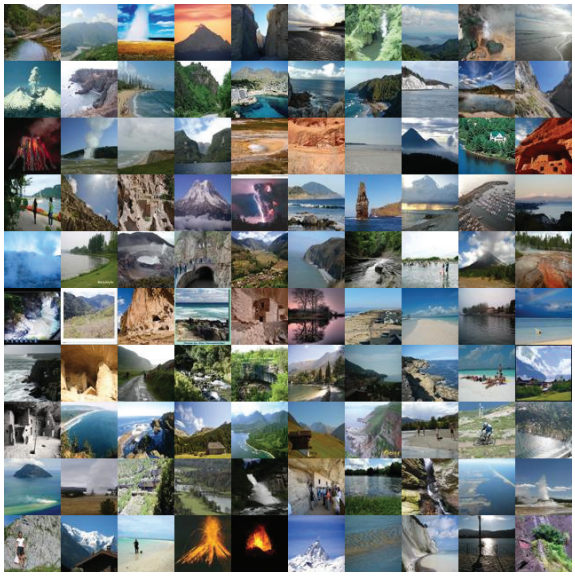
Figure 2: Generating texture patterns. Each row displays one texture experiment, where the first image is the training image, and the rest are 3 of the images generated by the CoopNets algorithm.

Our MCMC teaching of the generator is consistent with the learning objective $\mathrm{KL}(q_\alpha | p_{\theta^{(t)}})$, because

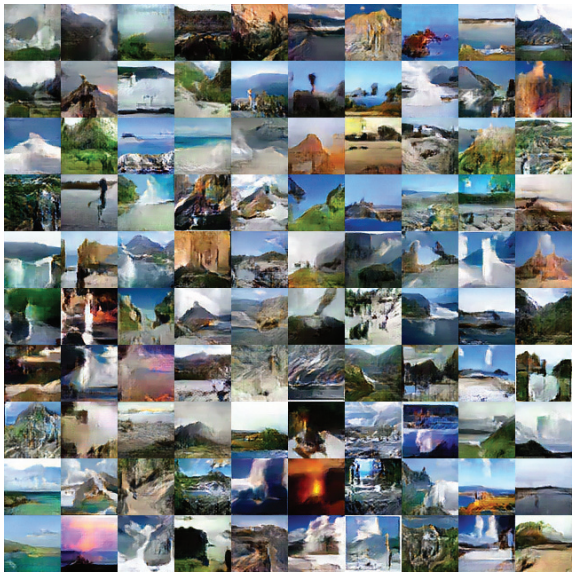
$$\mathrm{KL}(p^{(t+1)} | p_{\theta^{(t)}}) \leq \mathrm{KL}(q_{\alpha^{(t)}} | p_{\theta^{(t)}}). \quad (14)$$

In fact, $\mathrm{KL}(p^{(t+1)} | p_{\theta^{(t)}}) \to 0$ monotonically as $l_p \to \infty$ due to the second law of thermodynamics (Cover and Thomas 2012). The reduction of the Kullback-Leibler divergence in (14) in the transition from $q_{\alpha^{(t)}}$ to $p^{(t+1)}$ and the projection from $p^{(t+1)}$ to $q_{\alpha^{(t+1)}}$ in the MCMC teaching are consistent with the learning objective of reducing $\mathrm{KL}(q_\alpha | p_{\theta^{(t)}})$ in (Kim and Bengio 2016). But the Monte Carlo implementation of $\mathcal{M}$ in our work avoids the need to approximate the intractable entropy term.

For MCMC teaching, the right diagram in (11) leads to the update $\alpha^{(t+1)} = \arg\min_\alpha \mathrm{KL}(\mathcal{M}_\theta q_{\alpha^{(t)}} | q_\alpha)$, where $\{\tilde{Y}_i\} \sim \mathcal{M}_\theta q_{\alpha^{(t)}}$ serve as the training data, and the latent vector X_i is inferred by Langevin dynamics initialized from $\hat{X}_i$. The left diagram in (11) can be viewed as a simplified approximation to the right diagram by fixing the latent vector at $\hat{X}_i$. It minimizes a variational upper bound $\mathrm{KL}(\mathcal{M}_\theta q_{\alpha^{(t)}} | q_\alpha) + \mathrm{KL}(q(\hat{X}_i | \tilde{Y}_i, \alpha^{(t)}) | q(X_i | \tilde{Y}_i, \alpha))$. Our experiments suggest that the variational learning step in the left diagram works as well as the maximum likelihood learning



(a) Training images.



(b) Synthesized images.

Figure 3: Images generated by CoopNets learned from 10 Imagenet scene categories. The training set consists of 1100 images randomly sampled from each category. The total number of training images is 11,000.

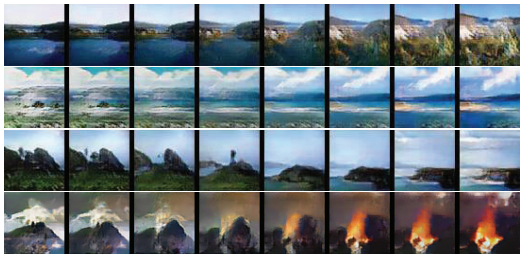


Figure 4: Interpolation between latent vectors of the images on the two ends.

step in the right diagram.

5 Experiments

We use the MatConvNet of (Vedaldi and Lenc 2015) for coding. For the descriptor net, we adopt the structure of (Xie et al. 2016), where the bottom-up network consists of multiple layers of convolution by linear filtering, ReLU non-linearity, and down-sampling. We adopt the structure of the generator network of (Radford, Metz, and Chintala, 2015; Dosovitskiy, Springenberg, and Brox, 2015), where the top-down network consists of multiple layers of deconvolution by linear superposition, ReLU non-linearity, and up-sampling, with tanh non-linearity at the bottom-layer (Radford, Metz, and Chintala 2015) to make the signals fall within $[-1, 1]$. In our experiments, we set $l_q = 0$ and infer $X_i = \hat{X}_i$, i.e., we follow the left diagram in (11). We have also experimented with $l_q > 0$, i.e., the right diagram, but did not observe significant improvement.

5.1 Experiment on texture synthesis

We first conduct qualitative experiments on generating texture patterns. We learn a separate model from each texture image. The training images are collected from the Internet, and resized to 224 $\times$ 224. The synthesized images are of the same size as the training images.

We use a 3-layer descriptor net, where the first layer has 100 15 $\times$ 15 filters with sub-sampling rate of 3 pixels, the second layer has 70 9 $\times$ 9 filters with sub-sampling of 1, and the third layer has 30 7 $\times$ 7 filters with sub-sampling of 1. We fix the standard deviation of the reference distribution of the descriptor net to be $s = 0.012$. We use $l_p = 20$ or 30 steps of Langevin revision dynamics within each learning iteration, and the Langevin step size is set at 0.003. The learning rate is 0.01. Starting from 7 $\times$ 7 latent factors, the generator net has 5 layers of deconvolution with 5 $\times$ 5 kernels (basis functions), with an up-sampling factor of 2 at each layer. The standard deviation of the noise vector is $\sigma = 0.3$. The learning rate is 10^{-6}. The number of generator learning steps is 1 at each cooperative learning iteration. We run 10^4 cooperative learning iterations to train the models. Figure 2 displays the results of generating texture patterns. For each category, the first image is the training image, and the rest are 3 of the images generated by the learning algorithm. We run $\tilde{n} = 6$ parallel chains for the first example, where images from 3 of them are presented. We run a single chain for the rest of the examples, where the synthesized images are generated at different iterations. Even though we run a single chain, it is as if we run an infinite number of chains, because in each learning iteration, we run Langevin dynamics from a new image sampled from the generator.

5.2 Experiment on scene and object synthesis

We conduct experiments on synthesizing images of categories from Imagenet ILSVRC2012 dataset (Deng et al. 2009). We adopt a 4-layer descriptor net. The first layer has 64 5 $\times$ 5 filters with sub-sampling of 2 pixels, the second layers has 128 3 $\times$ 3 filters with sub-sampling of 2, the third layer has 256 3 $\times$ 3 filters with sub-sampling of 1, and the final layer is

Table 1: Inception scores of different methods on learning from 10 Imagenet scene categories. n is the number of training images randomly sampled from each category.

	$n = 50$	$n = 100$	$n = 300$	$n = 500$	$n = 700$	$n = 900$	$n = 1100$
CoopNets	**2.66±.13**	**3.04±.13**	**3.41±.13**	**3.48±.08**	**3.59±.11**	**3.65±.07**	**3.79±.15**
DCGAN (Radford, Metz, and Chintala 2015)	2.26±.16	2.50±.15	3.16±.15	3.05±.12	3.13±.09	3.34±.05	3.47±.06
EBGAN (Zhao, Mathieu, and LeCun 2016)	2.23±.17	2.40±.14	2.62±.08	2.46±.09	2.65±.04	2.64±.04	2.75±.08
W-GAN (Arjovsky, Chintala, and Bottou 2017)	1.80±.09	2.19±.12	2.34±.06	2.62±.08	2.86±.10	2.88±.07	3.14±.06
VAE (Kingma and Welling 2014)	1.62±.09	1.63±.06	1.65±.05	1.73±.04	1.67±.03	1.72±.02	1.73±.02
InfoGAN (Chen et al. 2016)	2.21±.04	1.73±.01	2.15±.03	2.42±.05	2.47±.05	2.29±.03	2.08±.04
DDGM (Kim and Bengio 2016)	2.65±.17	1.05±.03	3.27±.14	3.42±.09	3.47±.13	3.41±.08	3.34±.11
Algorithm G (Han et al. 2017)	1.72±.07	1.94±.09	2.32±.09	2.40±.06	2.45±.05	2.54±.05	2.61±.06
Persistent CD (Tieleman 2008)	1.30±.08	1.94±.03	1.80±.02	1.53±.02	1.45±.04	1.35±.02	1.51±.02

Table 2: Comparison of recovery performances of different methods in 3 experiments

	task	CoopNets	DCGAN	MRF-ℓ_1	MRF-ℓ_2	inter-1	inter-2	inter-3	inter-4	inter-5
error	M30	**0.115**	0.211	0.132	0.134	0.120	0.120	0.265	0.120	0.120
	M40	**0.124**	0.212	0.148	0.149	0.135	0.135	0.314	0.135	0.135
	M50	**0.136**	0.214	0.178	0.179	0.170	0.166	0.353	0.164	0.164
PSNR	M30	**16.893**	12.116	15.739	15.692	16.203	16.635	9.524	16.665	16.648
	M40	**16.098**	11.984	14.834	14.785	15.065	15.644	8.178	15.698	15.688
	M50	**15.105**	11.890	13.313	13.309	13.220	14.009	7.327	14.164	14.161

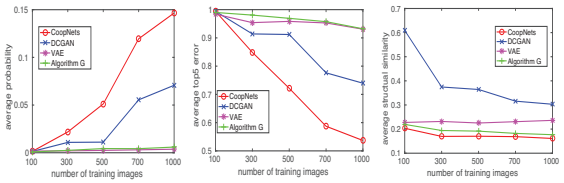


Figure 5: Left: Average softmax class probability on single Imagenet category versus the number of training images. Middle: Top 5 classification error. Right: Average pairwise structural similarity.

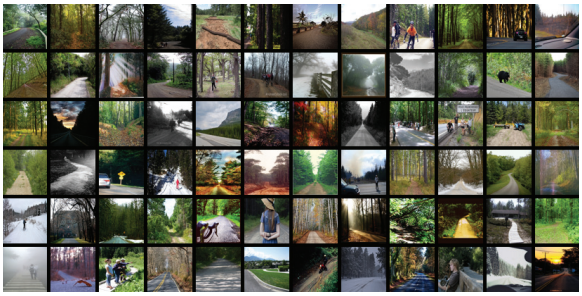
a fully connected layer with 100 channels. We set the number of Langevin dynamics steps in each learning iteration to l_p=10 and the step size to 0.002. The learning rate is 0.07. The number of learning iterations is about $1,000$. After learning the models, we synthesize images using the learned models. As in the CoopNet algorithm, we sample from the learned descriptor model by running 10 to 50 steps of Langevin dynamics initialized from the examples generated by the learned generator model.

In our first experiment, we learn from images that are randomly sampled from 10 Imagenet scene categories (alp, cliff drop, cliff dwelling, geyser, lakeside, promontory, sandbar, seashore, valley, and volcano). We conduct 7 runs. The numbers of images sampled from each category are 50, 100, 300, 500, 700, 900, and 1100 respectively in these 7 runs. Figure 3 displays the observed examples randomly sampled from the training set, and the synthesized examples generated by the CoopNets, where the number of training images from each category is 1100. The synthesized examples are randomly sampled from the learned models without cheery picking.

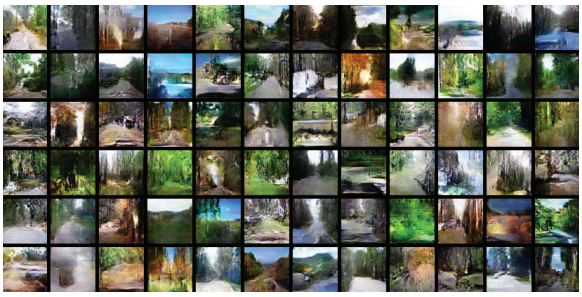
Figure 4 shows 4 examples of interpolating between two latent X vectors. For each row, the images at the two ends are generated from X vectors randomly sampled from $\mathrm{N}(0, I_d)$. Each image in the middle is obtained by first interpolating the X vectors of the two end images, and then generating the image using the generator, followed by 10 steps of Langevin dynamics. This experiment shows that we learn smooth generator model that traces the manifold of the data distribution.

We evaluate the synthesis quality by the Inception score (Salimans et al. 2016). Table 1 displays the Inception scores of the CoopNets, DCGAN (Radford, Metz, and Chintala 2015), EBGAN (Zhao, Mathieu, and LeCun 2016), Wasserstein GAN (Arjovsky, Chintala, and Bottou 2017), InfoGAN (Chen et al. 2016), VAE (Kingma and Welling 2014), DDGM (Kim and Bengio 2016), and separate training by Algorithm G and Algorithm D. For Algorithm D, we initialize the synthesized examples from the observed examples, so it is persistent contrastive divergence (Tieleman 2008).

In our second experiment, we learn from images randomly sampled from a single Imagenet object category. We then evaluate the synthesis quality using three criteria: (1) average softmax class probability that the Inception network (Szegedy et al. 2016) assigns to the synthesized images for the underlying category. (2) top-5 classification error by the Inception network, i.e., the probability that the underlying category does not belong to the categories with the top 5 softmax probabilities. (3) Average pairwise structural similarity (Wang et al. 2004) between two randomly sampled synthesized images. We conduct this experiment on 5 Imagenet categories: lemon, lifeboat, strawberry, school bus and zebra. Figure 5 displays the average results for the 5 categories. It can be seen that CoopNets generates images with higher softmax class probability, lower classification error, and higher variability than DCGAN and VAE. The advantage may be due to the fact that both models in CoopNets are learned generatively by maximum likelihood. Our experiences suggest that the CoopNets learning method is stable, and does not encounter



(a) Original images



(b) Synthesized images

Figure 6: Generating forest road images. The category is from MIT places205 dataset.

(a) Original images

(b) Synthesized images

Figure 7: Generating hotel room images. The category is from MIT places205 dataset.

mode collapsing issue.

5.3 Experiment on pattern completion

We conduct an experiment on learning from training images of human faces, and then testing the learned model on completing the occluded testing images. The structure of the generator network is the same as in (Radford, Metz, and Chintala, 2015; Dosovitskiy, Springenberg, and Brox, 2015). We adopt a 4-layer descriptor net. The first layer has 96 5×5 filters with sub-sampling of 2, the second layers has 128 5×5 filters with sub-sampling of 2, the third layer has 256 5×5 filters with sub-sampling of 2, and the final layer is a fully connected layer with 50 channels. We use l_p=10 steps of Langevin dynamics within each learning iteration, and the Langevin step size is set at 0.002. The learning rate is 0.07. The training data are $10,000$ human faces randomly sampled from CelebA dataset (Liu et al. 2015). We run 600 cooperative learning iterations.

To quantitatively test whether we have learned a good generator net $g(X;\alpha)$ even though it has never seen the training images directly in the training stage, we apply it to the task of recovering the occluded pixels of testing images. For each occluded testing image Y, we use Step G1 of Algorithm G to infer the latent vector X. The only change is with respect to the term $\|Y - g(X;\alpha)\|^2$, where the sum of squares is over all the observed pixels of Y in back-propagation computation. We run 1000 Langevin steps for inferring X, initializing X from $\mathrm{N}(0, I_d)$. After inferring X, the completed image $g(X;\alpha)$ is automatically obtained. We design 3 experiments, where we randomly place a 30×30, 40×40, or 50×50 mask on each 64×64 testing image. These 3 experiments are denoted by M30, M40, and M50 respectively (M for mask).

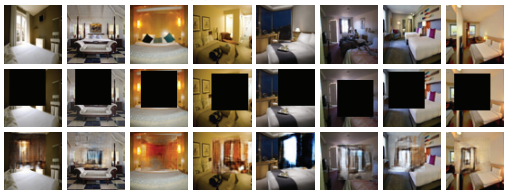
We report the recovery errors and compare our method with 7 different image inpainting methods as well as the DCGAN (Radford, Metz, and Chintala 2015). For DCGAN, we use the parameter setting in (Radford, Metz, and Chintala 2015) with the number of learning iterations increased to 600. We use the same $10,000$ training images to learn DCGAN. After the model is learned, we keep the generator and use the same method as ours to infer the latent vector X, and recover the unobserved pixels. In the 7 inpainting methods, Methods MRF-ℓ_1 and MRF-ℓ_2 are based on Markov random field prior where the nearest neighbor potential terms are ℓ_1 and ℓ_2 differences respectively. Methods inter-1 to 5 are interpolation methods. Please refer to (D'Errico 2004) for details.

Table 2 displays the recovery errors of the 3 experiments, where the error is measured by per pixel difference (relative to the range of pixel values) between the original image and the recovered image on the occluded region, averaged over 1,000 testing images. We also measure the error by PSNR. Figure 8 (a) displays some recovery results by our method. The first row shows the original images as the ground truth. The second row displays the testing images with occluded pixels. The third row displays the recovered images by the learned generator net.

We also apply the same method to MIT forest road category and hotel room category (Zhou et al. 2014). Figures 6 and 7 display randomly sampled training images and synthesized images. Figure 8 (b) and (c) display examples of pattern completion.

(a) face

(b) forest road



(c) hotel room

Figure 8: Pattern completion. First row: original images. Second row: occluded images. Third row: recovered images by CoopNets. (a) face. (b) forest road. (c) hotel room

6 Conclusion

This paper proposes a cooperative learning method to train both the energy-based model and the latent variable model simultaneously, and demonstrates its performances by a variety of experiments. Because both models in our method are learned generatively, our learning algorithm is stable, and can be statistically efficient, especially when learning from small or moderate training data.

The most unique feature of our system is that the two networks feed each other the synthesized data in the learning process, including initial, revised, and reconstructed synthesized data. Another unique feature is that the learning process interweaves the existing maximum likelihood learning algorithms for the two networks. A third unique feature is that the MCMC transitions for the descriptor are memorized and reproduced by the generator via ancestral sampling. Powering the MCMC sampling of the descriptor model in (Lu, Zhu, and Wu, 2016; Xie et al., 2016) is a main motivation of this paper, with the bonus of turning the unsupervised learning of the generator (Han et al. 2017) into supervised learning.

Project page

The code and more results can be found at http://www.stat.ucla.edu/~ywu/CoopNets/main.html

Acknowledgment

We acknowledge Dr. Song-Chun Zhu's important contributions to the work presented in this paper.

We thank a reviewer for his or her insightful comments.

We thank Hansheng Jiang for her work on this project as a summer visiting student. We thank Tengyu Liu and Zilong Zheng for assistance with the inception score comparison experiments.

The work is supported by NSF DMS 1310391, DARPA SIMPLEX N66001-15-C-4035, ONR MURI N00014-16-1-2007, and DARPA ARO W911NF-16-1-0579.

References

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Blum, A., and Mitchell, T. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, 92–100. ACM.

Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; and Abbeel, P. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2172–2180.

Cover, T. M., and Thomas, J. A. 2012. *Elements of information theory*. John Wiley & Sons.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 248–255.

Denton, E. L.; Chintala, S.; Fergus, R.; et al. 2015. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems*, 1486–1494.

D'Errico, J. 2004. Interpolation inpainting. In *https://www.mathworks.com/matlabcentral/fileexchange/4551-inpaint-nans*.

Dosovitskiy, E.; Springenberg, J. T.; and Brox, T. 2015. Learning to generate chairs with convolutional neural networks. In *IEEE International Conference on Computer Vision and Pattern Recognition*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2672–2680.

Han, T.; Lu, Y.; Zhu, S.-C.; and Wu, Y. N. 2017. Alternating back-propagation for generator network. In *31st AAAI Conference on Artificial Intelligence*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hinton, G. E. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation* 14(8):1771–1800.

Kim, T., and Bengio, Y. 2016. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*.

Kingma, D. P., and Welling, M. 2014. Auto-encoding variational bayes. *ICLR*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105.

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

LeCun, Y.; Chopra, S.; Hadsell, R.; Ranzato, M.; and Huang, F. J. 2006. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press.

Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2015. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, 3730–3738.

Lu, Y.; Zhu, S.-C.; and Wu, Y. N. 2016. Learning FRAME models using CNN filters. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Mnih, A., and Gregor, K. 2014. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*.

Ngiam, J.; Chen, Z.; Koh, P. W.; and Ng, A. Y. 2011. Learning deep energy models. In *International Conference on Machine Learning*.

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

Rezende, D. J.; Mohamed, S.; and Wierstra, D. 2014. Stochastic backpropagation and approximate inference in deep generative models. In Jebara, T., and Xing, E. P., eds., *International Conference on Machine Learning*, 1278–1286.

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2226–2234.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826.

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

Vedaldi, A., and Lenc, K. 2015. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*.

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13(4):600–612.

Xie, J.; Lu, Y.; Zhu, S.-C.; and Wu, Y. N. 2016. A theory of generative convnet. In *International Conference on Machine Learning*.

Younes, L. 1999. On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics: An International Journal of Probability and Stochastic Processes* 65(3-4):177–228.

Zhao, J.; Mathieu, M.; and LeCun, Y. 2016. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*.

Zhou, B.; Lapedriza, A.; Xiao, J.; Torralba, A.; and Oliva, A. 2014. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, 487–495.

Zhu, S.-C.; Wu, Y. N.; and Mumford, D. 1997. Minimax entropy principle and its application to texture modeling. *Neural Computation* 9(8):1627–1660.

Zhu, S.-C. 2003. Statistical modeling and conceptualization of visual patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25(6):691–712.