COOPERATIVE TRAINING OF DESCRIPTOR AND GENERATOR NETWORKS

Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu & Ying Nian Wu
Department of Statistics
University of California, Los Angeles
{jianwen, yanglv, ruiqigao}@ucla.edu, {sczhu, ywu}@stat.ucla.edu

ABSTRACT

This paper studies the cooperative training of two probabilistic models of signals such as images. Both models are parametrized by convolutional neural networks (ConvNets). The first network is a descriptor network, which is an exponential family model or an energy-based model, whose feature statistics or energy function are defined by a bottom-up ConvNet, which maps the observed signal to the feature statistics. The second network is a generator network, which is a non-linear version of factor analysis. It is defined by a top-down ConvNet, which maps the latent factors to the observed signal. The maximum likelihood training algorithms of both the descriptor net and the generator net are in the form of alternating back-propagation, and both algorithms involve Langevin sampling. We observe that the two training algorithms can cooperate with each other by jump-starting each other's Langevin sampling, and they can be seamlessly interwoven into a CoopNets algorithm that can train both nets simultaneously.

1 INTRODUCTION

1.1 TWO CONVNETS OF OPPOSITE DIRECTIONS

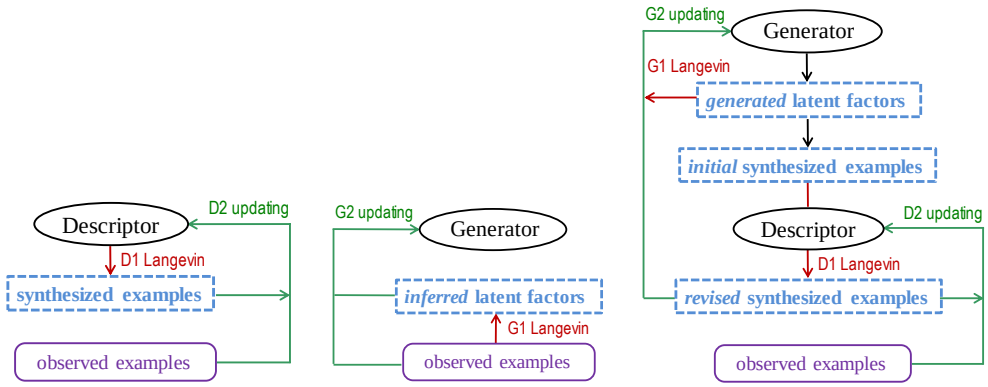
We begin with a story that the reader of this paper can readily relate to. A student writes up an initial draft of a paper. His advisor then revises it. After that they submit the revised paper for review. The student then learns from his advisor's revision, while the advisor learns from the outside review. In this story, the advisor guides the student, but the student does most of the work.

This paper is about two probabilistic models of signals such as images, and they play the roles of student and advisor as described above. Both models are parametrized by convolutional neural networks (ConvNets or CNNs) (LeCun et al., 1998; Krizhevsky et al., 2012). The two nets take two opposite directions. One is bottom-up, and the other is top-down, as illustrate by the following diagram:

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

The simultaneous training of such two nets was first studied by the recent work of Kim & Bengio (2016). These two nets belong to two major classes of probabilistic models. (a) The exponential family models or the energy-based models (LeCun et al., 2006) or the Markov random field models (Zhu et al., 1997), where the probability distribution is defined by feature statistics or energy function computed from the signal by a bottom-up process. (b) The latent variable models or the directed graphical models, where the signal is assumed to be a transformation of the latent factors that follow a known prior distribution. The latent factors generate the signal by a top-down process. A classical example is factor analysis.

The two classes of models have been contrasted by Zhu (2003); Teh et al. (2003); Ngiam et al. (2011). Zhu (2003) called the two classes of models the descriptive models and the generative



(a) Algorithm D (b) Algorithm G (c) CoopNets algorithm

Figure 1: (a) Algorithm D involves sampling from the current model by Langevin dynamics. (b) Algorithm G involves sampling from the posterior distribution of the latent factors by Langevin dynamics. (c) CoopNets algorithm. The part of the flowchart for training the descriptor is similar to Algorithm D, except that the D1 Langevin sampling is initialized from the initial synthesized examples supplied by the generator. The part of the flowchart for training the generator can also be mapped to Algorithm G, except that the revised synthesized examples play the role of the observed data, and the known generated latent factors can be used as inferred latent factors (or be used to initialize the G1 Langevin sampling of the latent factors).

models respectively. Both classes of models can benefit from the high capacity of the multi-layer ConvNets. (a) In the exponential family models or the energy-based models, the feature statistics or the energy function can be defined by a bottom-up ConvNet that maps the signal to the features and the energy function (Ngiam et al., 2011; Xie et al., 2016). We call the resulting model a descriptive network or a descriptor net following Zhu (2003), because it is built on descriptive feature statistics. (b) In the latent variable models or the directed graphical models, the transformation from the latent factors to the signal can be defined by a top-down ConvNet (Dosovitskiy et al., 2015), which maps the latent factors to the signal. We call the resulting model a generative network or generator net following Goodfellow et al. (2014), who proposed such a model in their work on the generative adversarial networks (GAN).

1.2 Two training algorithms and their cooperation

Fig. 1(a) and (b) display the flowcharts of the maximum likelihood learning algorithms for training the descriptor and generator nets . We call the two algorithms Algorithm D and Algorithm G respectively. Algorithm D (Xie et al., 2016) iterates two steps: Step D1 synthesizes examples by sampling from the current model by Langevin dynamics. Step D2 updates the parameters to shift the density from the synthesized examples towards the observed examples. Algorithm G (Han et al., 2017) also iterates two steps. Step G1 infers latent factors for each observed example by sampling from their posterior distribution by Langevin dynamics. Step G2 updates the parameters by a non-linear regression of the observed examples on their corresponding latent factors. We use Langevin dynamics for Markov chain Monte Carlo (MCMC) sampling because the gradient term of Langevin dynamics can be readily computed via back-propagation. Thus all the steps D1, D2 and G1, G2 are powered by back-propagation, and both Algorithms D and G are alternating back-propagation algorithms.

In this article, we propose to couple Algorithms D and G into a cooperative training algorithm that interweaves the steps of the two algorithms seamlessly. We call the resulting algorithm the CoopNets algorithm, and we show that it can train both nets simultaneously.

Figure 1(c) displays the flowchart of the CoopNets algorithm. The generator is like the student. It generates the initial draft of the synthesized examples. The descriptor is like the advisor. It revises the initial draft by initializing its Langevin dynamics from the initial draft in Step D1, which produces the revised draft of the synthesized examples. The descriptor learns from the outside review in Step D2, which is in the form of the difference between the observed examples and the revised

synthesized examples. The generator learns from how the descriptor revises the initial draft by reconstructing the revised draft in Step G2. For each synthesized example, the generator knows the latent factors that generate the initial draft, so that Step G1 can infer the latent factors by initializing its Langevin dynamics from their known values.

In the CoopNets algorithm, the generator fuels the MCMC of the descriptor by supplying initial synthesized examples, which can be obtained by direct ancestral sampling. The generator then learns from the revised synthesized examples with virtually known latent factors. The cooperation is thus beneficial to both nets.

2 RELATED WORK

Our work is inspired by the generative adversarial networks (GAN) (Goodfellow et al., 2014; Denton et al., 2015; Radford et al., 2015). In GAN, the generator net is paired with a discriminator net. The two nets play adversarial roles. In our work, the generator net and the descriptor net play cooperative roles, and they feed each other the initial, revised and reconstructed synthesized data. The learning of both nets is based on maximum likelihood, and the learning process is quite stable because of the cooperative nature and the consistent directions of the two maximum likelihood training algorithms.

Another method to train the generator network is variational auto-encoder (VAE) (Kingma & Welling, 2014; Rezende et al., 2014; Mnih & Gregor, 2014), which learns an inferential or recognition network to approximate the posterior distribution of the latent factors.

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 is most similar to the recent work of Kim & Bengio (2016). In fact, the settings of the two nets are the same. In their work, 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 two nets interact with each other via synthesized data, and the generator learns from the descriptor by reconstructing the revised draft of synthesized examples. Our method does not need to approximate the intractable entropy term.

Our work is related to the contrastive divergence algorithm (Hinton, 2002) for training the descriptor net. The contrastive divergence initializes the MCMC sampling from the observed examples. The CoopNets algorithm initializes the MCMC sampling from the examples supplied by the generator.

3 TWO NETS AND TWO TRAINING ALGORITHMS

3.1 DESCRIPTOR NET AND TRAINING ALGORITHM

Let Y be the D-dimensional signal, such as an image. The descriptor model is in the form of exponential tilting of a reference distribution (Xie et al., 2016):

$$P_{\mathcal{D}}(Y; W_{\mathcal{D}}) = \frac{1}{Z(W_{\mathcal{D}})} \exp\left[f(Y; W_{\mathcal{D}})\right] q(Y), \tag{2}$$

where $q(Y)$ is the reference distribution such as Gaussian white noise $q(Y) \propto \exp\left(-\|Y\|^2/2s^2\right)$, $f(Y; W_{\mathcal{D}})$ (f stands for features) is the feature statistics or energy function, defined by a ConvNet whose parameters are denoted by $W_{\mathcal{D}}$. This ConvNet is bottom-up because it maps the signal Y to a number. See the diagram in (1). $Z(W_{\mathcal{D}}) = \int \exp\left[f(Y; W_{\mathcal{D}})\right] q(Y) dY = \mathrm{E}_q\{\exp[f(Y; W_{\mathcal{D}})]\}$ is the normalizing constant, where E_q is the expectation with respect to q.

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

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

where $\mathrm{E}_{W_\mathcal{D}}$ denotes the expectation with respect to $P_\mathcal{D}(Y; W_\mathcal{D})$.

The expectation in equation (3) is analytically intractable and has to be approximated by MCMC, such as Langevin dynamics, which iterates the following step:

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

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.

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 $L'_\mathcal{D}(W_\mathcal{D})$ is

$$L'_\mathcal{D}(W_\mathcal{D}) \approx \frac{1}{n}\sum_{i=1}^{n} \frac{\partial}{\partial W_\mathcal{D}} f(Y_i; W_\mathcal{D}) - \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} \frac{\partial}{\partial W_\mathcal{D}} f(\tilde{Y}_i; W_\mathcal{D}), \tag{5}$$

which is used to update $W_\mathcal{D}$.

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

3.2 Generator net and training algorithm

The generator net (Goodfellow et al., 2014) seeks to explain the signal Y of dimension D by a vector of latent factors X of dimension d, and usually $d \ll D$. The model is of the following form:

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

$g(X; W_\mathcal{G})$ (g stands for generator) is a top-down ConvNet defined by the parameters $W_\mathcal{G}$. The ConvNet g maps the latent factors X to the signal Y. See the diagram in (1).

The joint density of model (6) is $P_\mathcal{G}(X, Y; W_\mathcal{G}) = P_\mathcal{G}(X)P_\mathcal{G}(Y|X; W_\mathcal{G})$, and

$$\log P_\mathcal{G}(X, Y; W_\mathcal{G}) = -\frac{1}{2\sigma^2}\|Y - g(X; W_\mathcal{G})\|^2 - \frac{1}{2}\|X\|^2 + \text{constant}, \tag{7}$$

where the constant term is independent of X, Y and $W_\mathcal{G}$. The marginal density is obtained by integrating out the latent factors X, i.e., $P_\mathcal{G}(Y; W_\mathcal{G}) = \int P_\mathcal{G}(X, Y; W_\mathcal{G})dX$. The inference of X given Y is based on the posterior density $P_\mathcal{G}(X|Y; W_\mathcal{G}) = P_\mathcal{G}(X, Y; W_\mathcal{G})/P_\mathcal{G}(Y; W_\mathcal{G}) \propto P_\mathcal{G}(X, Y; W_\mathcal{G})$ 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 $L_\mathcal{G}(W_\mathcal{G}) = \frac{1}{n}\sum_{i=1}^{n} \log P_\mathcal{G}(Y_i; W_\mathcal{G})$. For large sample, the learned $W_\mathcal{G}$ minimizes the Kullback-Leibler divergence $\mathrm{KL}(P_{\text{data}}|P_\mathcal{G})$ from the data distribution P_{data} to the model distribution $P_\mathcal{G}$. The gradient of $L_\mathcal{G}(W_\mathcal{G})$ is obtained according to the following identity

$$\begin{aligned}
\frac{\partial}{\partial W_\mathcal{G}} \log P_\mathcal{G}(Y; W_\mathcal{G}) &= \frac{1}{P_\mathcal{G}(Y; W_\mathcal{G})}\frac{\partial}{\partial W_\mathcal{G}}\int P_\mathcal{G}(Y, X; W_\mathcal{G})dX \\
&= \frac{1}{P_\mathcal{G}(Y; W_\mathcal{G})}\int \left[\frac{\partial}{\partial W_\mathcal{G}} \log P_\mathcal{G}(Y, X; W_\mathcal{G})\right] P_\mathcal{G}(Y, X; W_\mathcal{G})dX \\
&= \int \left[\frac{\partial}{\partial W_\mathcal{G}} \log P_\mathcal{G}(Y, X; W_\mathcal{G})\right] \frac{P_\mathcal{G}(Y, X; W_\mathcal{G})}{P_\mathcal{G}(Y; W_\mathcal{G})}dX \\
&= \mathrm{E}_{P_\mathcal{G}(X|Y; W_\mathcal{G})}\left[\frac{\partial}{\partial W_\mathcal{G}} \log P_\mathcal{G}(X, Y; W_\mathcal{G})\right],
\end{aligned} \tag{8}$$

which underlies the EM algorithm. In general, the expectation in (8) is analytically intractable, and has to be approximated by MCMC that samples from the posterior $P_G(X|Y; W_\mathcal{G})$, such as Langevin dynamics, which iterates

$$X_{\tau+1} = X_\tau + \frac{\delta^2}{2}\frac{\partial}{\partial X} \log P_\mathcal{G}(X_\tau, Y; W_\mathcal{G}) + \delta U_\tau, \tag{9}$$

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

$$L'_\mathcal{G}(W_\mathcal{G}) \approx \frac{1}{n}\sum_{i=1}^{n} \frac{\partial}{\partial W_\mathcal{G}} \log P_\mathcal{G}(X_i, Y_i; W_\mathcal{G}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{\sigma^2}\left(Y_i - g(X_i; W_\mathcal{G})\right) \frac{\partial}{\partial W_\mathcal{G}} g(X_i; W_\mathcal{G}). \quad (10)$$

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

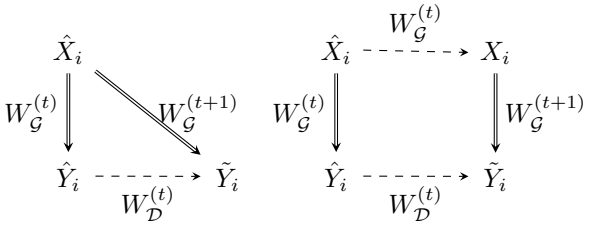
4 CoopNets algorithm: reconstructing the revision

In Algorithms D and G, both steps D1 and G1 are Langevin dynamics, which may be slow to converge. An interesting observation is that the two algorithms can cooperate with each other by jumpstarting each other's Langevin 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 $\hat{Y}_i = g(\hat{X}_i; W_\mathcal{G}) + \epsilon_i$, for $i = 1, ..., \tilde{n}$. If the current generator $P_\mathcal{G}$ is close to the current descriptor $P_\mathcal{D}$, 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 $l_\mathcal{D}$ 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 net. We can then update $W_\mathcal{D}$ according to Step D2 of Algorithm D.

In order to update $W_\mathcal{G}$ of the generator net, 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 Langevin dynamics initialized from the $\{\hat{Y}_i, i = 1, ..., \tilde{n}\}$ produced by the generator net with known latent factors $\{\hat{X}_i, i = 1, ..., \tilde{n}\}$, we can update $W_\mathcal{G}$ 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 $W_\mathcal{G}^{(t)}$, the latent factors $\hat{X}_i$ generates and thus reconstructs the initial example $\hat{Y}_i$. After updating $W_\mathcal{G}$, we want $\hat{X}_i$ to reconstruct the revised example $\tilde{Y}_i$. That is, we revise $W_\mathcal{G}$ to absorb the revision from $\hat{Y}_i$ to $\tilde{Y}_i$, so that the generator shifts its density from $\{\hat{Y}_i\}$ to $\{\tilde{Y}_i\}$. The reconstruction error can tell us whether the generator has caught up with the descriptor by fully absorbing the revision.

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 net, while the dashed-line arrows indicate Langevin dynamics for revision and inference in the two nets. The diagram on the right in (11) illustrates a more rigorous method, where we initialize the Langevin inference of $\{X_i, i = 1, ..., \tilde{n}\}$ in Step G1 from $\{\hat{X}_i\}$, and then update $W_\mathcal{G}$ in Step G2 based on $\{(\tilde{Y}_i, X_i), i = 1, ..., \tilde{n}\}$. The diagram on the right shows how the two nets jumpstart each other's Langevin dynamics.

Algorithm 1 describes the cooperative training that interweaves Algorithm D and Algorithm G. See Figure 1(c) for the flowchart of the CoopNets algorithm. In our experiments, we set $l_\mathcal{G} = 0$ and infer $X_i = \hat{X}_i$ for simplicity, i.e., we follow the left diagram in (11).

See Appendix for a theoretical understanding of the convergence of the CoopNets algorithm.

Algorithm 1 CoopNets Algorithm

Input:

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

Output:

(1) estimated parameters $W_{\mathcal{D}}$ and $W_{\mathcal{G}}$
(2) synthesized examples $\{\hat{Y}_i, \tilde{Y}_i, i = 1, ..., \tilde{n}\}$

1: Let $t \leftarrow 0$, initialize $W_{\mathcal{D}}$ and $W_{\mathcal{G}}$.
2: **repeat**
3: **Step G0**: For $i = 1, ..., \tilde{n}$, generate $\hat{X}_i \sim \mathrm{N}(0, I_d)$, and generate $\hat{Y}_i = g(\hat{X}_i; W_{\mathcal{G}}^{(t)}) + \epsilon_i$.
4: **Step D1**: For $i = 1, ..., \tilde{n}$, starting from $\hat{Y}_i$, Run $l_{\mathcal{D}}$ steps of Langevin dynamics to obtain $\tilde{Y}_i$, each step following equation (4).
5: **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 $l_{\mathcal{G}}$ steps of Langevin dynamics to update X_i, each step following equation (9).
6: **Step D2**: Update $W_{\mathcal{D}}^{(t+1)} = W_{\mathcal{D}}^{(t)} + \gamma_t L'_{\mathcal{D}}(W_{\mathcal{D}}^{(t)})$, where $L'_{\mathcal{D}}(W_{\mathcal{D}}^{(t)})$ is computed according to (5).
7: **Step G2**: Update $W_{\mathcal{G}}^{(t+1)} = W_{\mathcal{G}}^{(t)} + \gamma_t {L_{\mathcal{G}}}'(W_{\mathcal{G}}^{(t)})$, where ${L_{\mathcal{G}}}'(W_G)$ is computed according to (10), except that Y_i is replaced by $\tilde{Y}_i$, and n by $\tilde{n}$.
8: Let $t \leftarrow t + 1$
9: **until** $t = T$

5 EXPERIMENTS

We use the MatConvNet of Vedaldi & 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 et al. (2015); Dosovitskiy et al. (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 et al., 2015) to make the signals fall within $[-1, 1]$.

5.1 QUANTITATIVE EXPERIMENT ON FACE COMPLETION

We conduct an experiment on learning from complete 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 et al., 2015; Dosovitskiy et al., 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 as output. We use L=10 steps of Langevin revision 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 selected from CelebA dataset (Liu et al., 2015). We run 600 cooperative learning iterations. Figure 2 displays 144 synthesized human faces by the descriptor net.

To quantitatively test whether we have learned a good generator net $g(X; W_{\mathcal{G}})$ 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 factors X. The only change is with respect to the term $\|Y - g(X; W_{\mathcal{G}})\|^2$, where the sum of squares is over all the observed pixels of Y in back-propagation computation. We run 1000 Langevin steps, initializing X from $\mathrm{N}(0, I_d)$. After inferring X, the completed image $g(X; W_{\mathcal{G}})$ is automatically obtained. We design 3 experiments, where we randomly place a 20×20, 30×30, or 40×40 mask on each 64×64 testing image. These 3 experiments are denoted by M20 M30, and M40 respectively (M for mask). We report the recovery errors and compare our method with 8 different image inpainting methods as well as the DCGAN of Radford et al. (2015).

Figure 2: Generating human face pattern. The synthesized images are generated by the CoopNets algorithm that learns from $10,000$ images.

For DCGAN, we use the parameter setting in Radford et al. (2015) except changing the number of learning iterations 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 latent factors X, and recover the unobserved pixels. In 8 inpainting methods, Methods 1 and 2 are based on Markov random field prior where the nearest neighbor potential terms are ℓ_2 and ℓ_1 differences respectively. Methods 3 to 8 are interpolation methods. Please refer to D'Errico (2004) for more details. Table 1 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 100 testing images. Fig. 3 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 generator net trained by the CoopNets algorithm on the 10,000 training images.

Table 1: Comparison of recovery errors among different inpainting methods in 3 experiments

Exp	Ours	GAN	1	2	3	4	5	6	7	8
M20	**.0966**	.2535	.1545	.1506	.1277	.1123	.2493	.1123	.1126	.1277
M30	**.1112**	.2606	.1820	.1792	.1679	.1321	.3367	.1310	.1312	.1679
M40	**.1184**	.2618	.2055	.2032	.1894	.1544	.3809	.1525	.1526	.1894

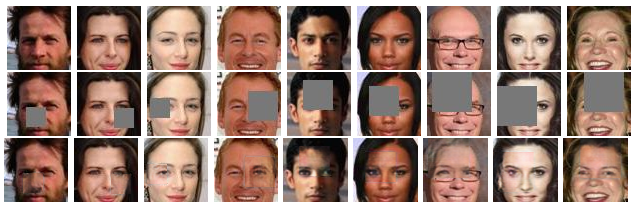


Figure 3: Row 1: ground-truth images. Row 2: testing images with occluded pixels. Row 3: recovered images by our method.

5.2 QUALITATIVE EXPERIMENT ON SYNTHESIS

We conduct an experiment on synthesizing images of categories from Imagenet ILSVRC2012 dataset (Deng et al., 2009) and MIT places205 dataset (Zhou et al., 2014). We adopt a 4-layer descriptor net. The first layer has 64 5×5 filters with sub-sampling of 2, the second layers has 128 3×3 filters with sub-sampling of 2, the third layer has 256 3×3 filters with sub-sampling of 1, and the final layer is a fully connected layer with 100 channels as output. We set the number of Langevin dynamics steps in each learning iteration to 10 and the step size to 0.002. The learning rate is 0.07. For each category, we randomly choose 1,000 images as training data and resize the images to 64×64. We run $1,000$ cooperative learning iterations to train the model. Figures 4 and 5 display the results for two categories, where for each category, we show 144 original images sampled from the training set, and 144 synthesized images generated by our method. The appendix contains more synthesis results.

As a comparison, we apply the Algorithm G alone and GAN code on the same 1,000 hotel room training images to learn the generator of the same structure as in CoopNets. Figure 6 displays the synthesis results.

We also try to synthesize images at high resolution (224×224). We adopt a 4-layer descriptor net. The first layer has 128 15×15 filters with sub-sampling of 3, the second layer has 256 3×3 filters with sub-sampling of 2, the third layer has 512 3×3 filters with sub-sampling of 1, and the final layer is a fully connected layer with 100 channels as output. We enlarge the filters of the final layer of generator net to 14×14 to generate 224×224 images. The learning rate is 0.05. We run 1000 cooperative learning iterations to train the model. Figures 7 and 8 show the synthesized images of two categories from MIT places205 dataset.

6 CONCLUSION

The most unique feature of our work 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 of our work is that the learning process interweaves the existing maximum likelihood learning algorithms for the two networks.

A third unique feature of our work is that the MCMC for the descriptor keeps rejuvenating the chains by refreshing the samples by independent replacements supplied by the generator, so that a single chain effectively amounts to an infinite number of chains or the evolution of the whole marginal distribution modeled by the generator.

CODE AND DATA

`http://www.stat.ucla.edu/~ywu/CoopNets/main.html`

7 APPENDIX: CONVERGENCE

7.1 GENERATOR OF INFINITE CAPACITY

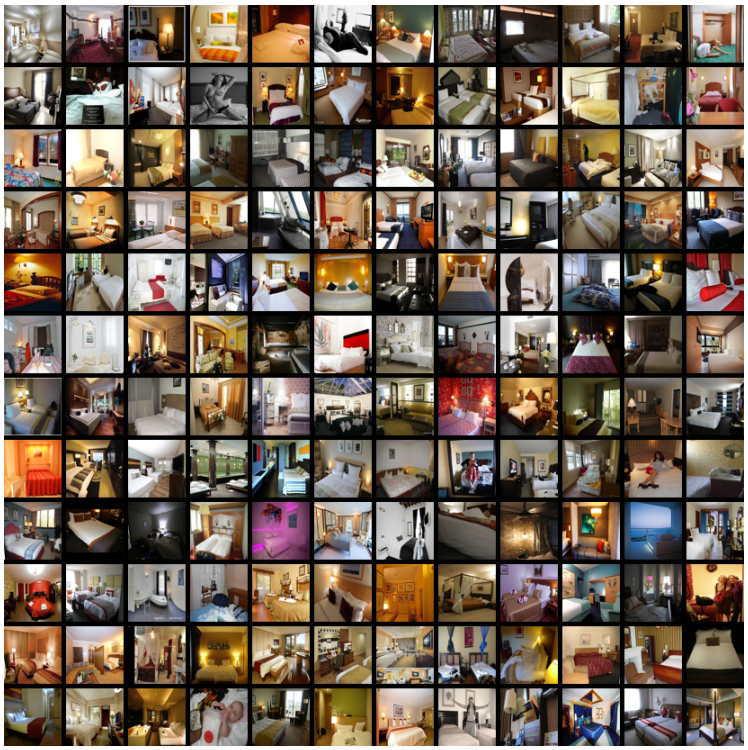
In the CoopNets algorithm, the descriptor learns from the observed examples, while the generator learns from the descriptor through the synthesized examples. Therefore, the descriptor is the driving force in terms of learning, although the generator is the driving force in terms of synthesis. In order to understand the convergence of learning, we can start from Algorithm D for learning the descriptor.

Algorithm D is a stochastic approximation algorithm (Robbins & Monro, 1951), except that the samples are generated by finite step MCMC transitions. According to Younes (1999), Algorithm D converges to the maximum likelihood estimate under suitable regularity conditions on the mixing of the transition kernel of the MCMC and the schedule of the learning rate γ_t, even if the number of Langevin steps $l_{\mathcal{D}}$ is finite or small (e.g., $l_{\mathcal{D}} = 1$), and even if the number of parallel chains $\tilde{n}$ is finite or small (e.g., $\tilde{n} = 1$). The reason is that the random fluctuations caused by the finite number of chains, $\tilde{n}$, and the limited mixing caused by the finite steps of MCMC, $l_{\mathcal{D}}$, are mitigated if the

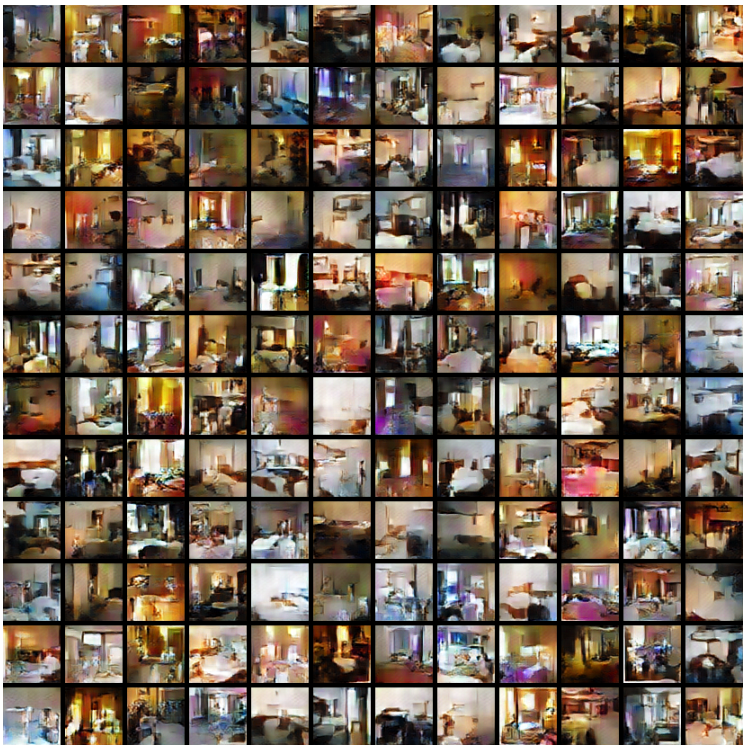
(a) Original images

(b) Synthesized images

Figure 4: Generating forest road images. The category is from MIT places205 dataset.

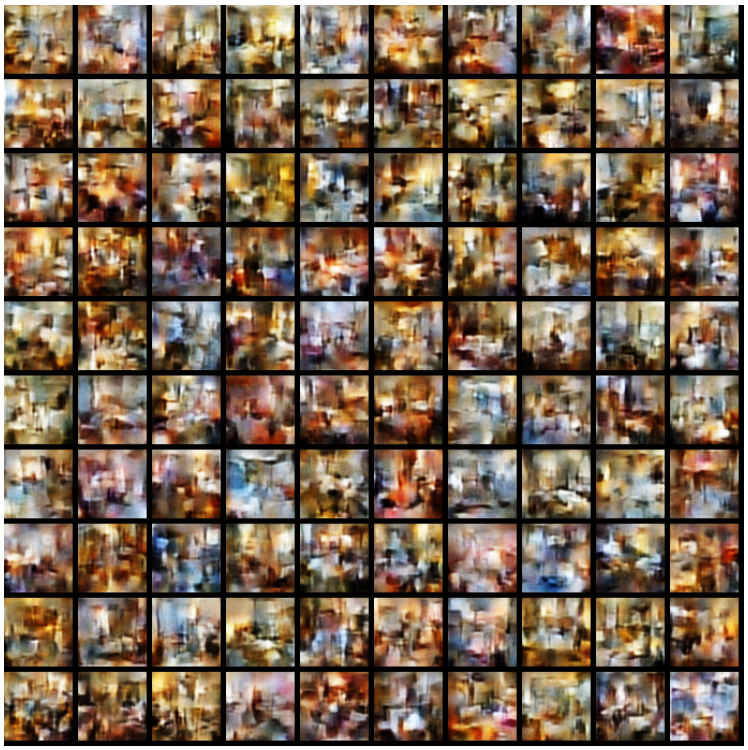


(a) Original images

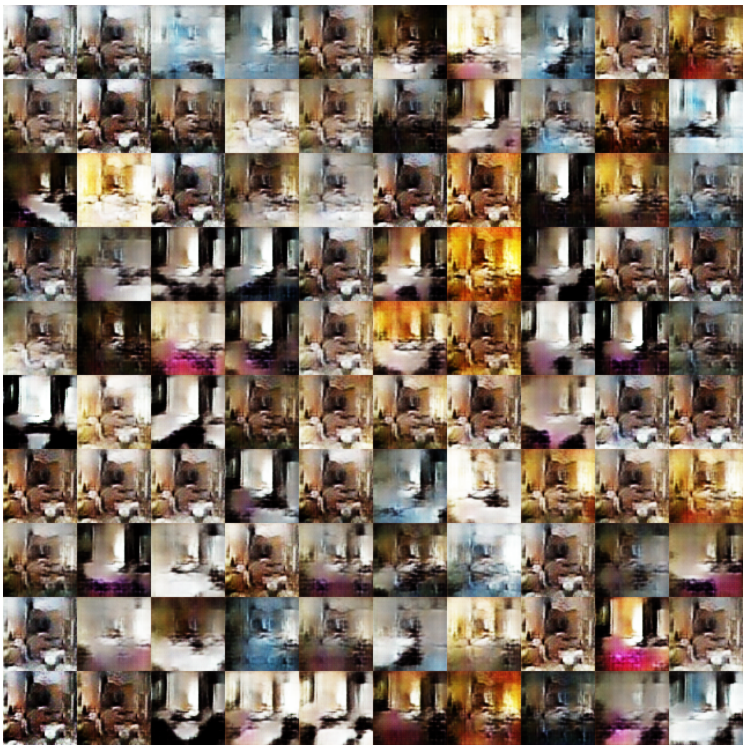


(b) Synthesized images

Figure 5: Generating hotel room images. The category is from MIT places205 dataset.



(a) Generated by Algorithm G alone.



(b) Generated by DCGAN code.

Figure 6: Generating hotel room images by Algorithm G alone and by GAN.

Figure 7: Generating forest road images at high resolution (224×224).

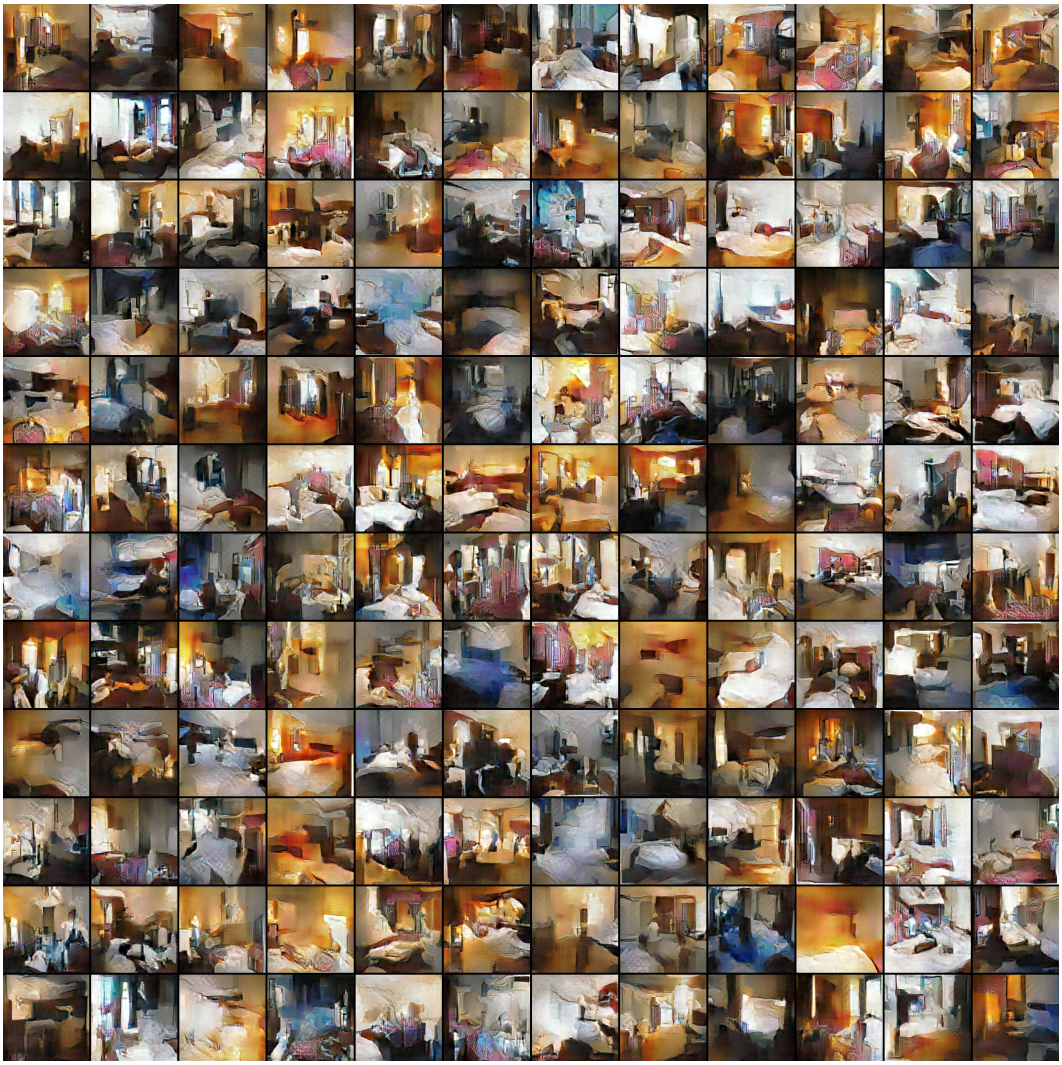


Figure 8: Generating hotel room images at high resolution (224×224).

learning rate γ_t is sufficiently small. At learning iteration t, let $W_{\mathcal{D}}^{(t)}$ be the estimated parameter of the descriptor. Let $P_{\mathcal{D}}^{(t+1)}$ be the marginal distribution of $\{\tilde{Y}_i\}$. Even though $P_{\mathcal{D}}^{(t+1)} \neq P_{\mathcal{D}}(Y; W_{\mathcal{D}}^{(t)})$ because $l_{\mathcal{D}}$ is finite ($P_{\mathcal{D}}^{(t+1)} = P_{\mathcal{D}}(Y; W_{\mathcal{D}}^{(t)})$ if $l_{\mathcal{D}} \to \infty$), we still have $W_{\mathcal{D}}^{(t)} \to \hat{W}_{\mathcal{D}}$ in probability according to Younes (1999), where $\hat{W}_{\mathcal{D}}$ is the maximum likelihood estimate of $W_{\mathcal{D}}$.

The efficiency of Algorithm D increases if the number of parallel chains $\tilde{n}$ is large because it leads to more accurate estimation of the expectation in the gradient $L'_{\mathcal{D}}(W_{\mathcal{D}})$ of equation (3), so that we can afford to use larger learning rate γ_t for faster convergence.

Now let us come back to the CoopNets algorithm. In order to understand how the descriptor net helps the training of the generator net, let us consider the idealized scenario where the number of parallel chains $\tilde{n} \to \infty$, and the generator has infinite capacity, and in each iteration it estimates $W_{\mathcal{G}}$ by maximum likelihood using the synthesized data from $P_{\mathcal{D}}^{(t+1)}$. In this idealized scenario, the learned generator $P_{\mathcal{G}}(Y; W_{\mathcal{G}}^{(t+1)})$ will reproduce $P_{\mathcal{D}}^{(t+1)}$ by minimizing $\mathrm{KL}(P_{\mathcal{D}}^{(t+1)}(Y)|P_{\mathcal{G}}(Y; W_{\mathcal{G}}))$, with $P_{\mathcal{D}}^{(t+1)}$ serving as its data distribution. Then eventually the learned generator $P_{\mathcal{G}}(Y, \hat{W}_{\mathcal{G}})$ will reproduce $P_{\mathcal{D}}(Y; \hat{W}_{\mathcal{D}})$. Thus the cooperative training helps the learning of the generator. Note that the learned generator $P_{\mathcal{G}}(Y, \hat{W}_{\mathcal{G}})$ will not reproduce the distribution of the observed data P_{data}, unless the descriptor is of infinite capacity too.

Conversely, the generator net also helps the learning of the descriptor net in the CoopNets algorithm. In Algorithm D, it is impractical to make the number of parallel chains $\tilde{n}$ too large. On the other hand, it would be difficult for a small number of chains $\{\tilde{Y}_i, i = 1, ..., \tilde{n}\}$ to explore the state space. In the CoopNets algorithm, because $P_{\mathcal{G}}(Y; W_{\mathcal{G}}^{(t)})$ reproduces $P_{\mathcal{D}}^{(t)}$, we can generate a completely new batch of independent samples $\{\hat{Y}_i\}$ from $P_{\mathcal{G}}(Y; W_{\mathcal{G}}^{(t)})$, and revise $\{\hat{Y}_i\}$ to $\{\tilde{Y}_i\}$ by Langevin dynamics, instead of running Langevin dynamics from the same old batch of $\{\tilde{Y}_i\}$ as in the original Algorithm D. This is like implementing an infinite number of parallel chains, because each iteration evolves a fresh batch of examples, as if each iteration evolves a new set of chains. By updating the generator $W_{\mathcal{G}}$, it is like we are updating the infinite number of parallel chains, because $W_{\mathcal{G}}$ memorizes the whole distribution. Even if $\tilde{n}$ in the CoopNets algorithm is small, e.g., $\tilde{n} = 1$, viewed from the perspective of Algorithm D, it is as if $\tilde{n} \to \infty$. Thus the above idealization $\tilde{n} \to \infty$ is sound.

7.2 Generator of finite capacity

From an information geometry point of view, let $\mathcal{D} = \{P_{\mathcal{D}}(Y; W_{\mathcal{D}}), \forall W_{\mathcal{D}}\}$ be the manifold of the descriptor models, where each distribution $P_{\mathcal{D}}(Y; W_{\mathcal{D}})$ is a point on this manifold. Then the maximum likelihood estimate of $W_{\mathcal{D}}$ is a projection of the data distribution P_{data} onto the manifold $\mathcal{D}$. Let $\mathcal{G} = \{P_{\mathcal{G}}(Y; W_{\mathcal{G}}), \forall W_{\mathcal{G}}\}$ be the manifold of the generator models, where each distribution $P_{\mathcal{G}}(Y; W_{\mathcal{G}})$ is a point on this manifold. Then the maximum likelihood estimate of $W_{\mathcal{G}}$ is a projection of the data distribution P_{data} onto the manifold $\mathcal{G}$.

From now on, for notational simplicity and with a slight abuse of notation, we use $W_{\mathcal{D}}$ to denote the descriptor distribution $P_{\mathcal{D}}(Y; W_{\mathcal{D}})$, and use $W_{\mathcal{G}}$ to denote the generator distribution $P_{\mathcal{G}}(Y; W_{\mathcal{G}})$.

We assume both the observed data size n and the synthesized data size $\tilde{n}$ are large enough so that we shall work on distributions or populations instead of finite samples. As explained above, assuming $\tilde{n} \to \infty$ is sound because the generator net can supply unlimited number of examples.

The Langevin revision dynamics runs a Markov chain from $W_{\mathcal{G}}^{(t)}$ towards $W_{\mathcal{D}}^{(t)}$. Let $\mathbf{L}_{W_{\mathcal{D}}}$ be the Markov transition kernel of $l_{\mathcal{D}}$ steps of Langevin revisions towards $W_{\mathcal{D}}$. The distribution of the revised synthesized data is

$$P_{\mathcal{D}}^{(t+1)} = \mathbf{L}_{W_{\mathcal{D}}^{(t)}} \cdot W_{\mathcal{G}}^{(t)}, \tag{12}$$

where the notation $\mathbf{L} \cdot P$ denotes the marginal distribution obtained by running the Markov transition $\mathbf{L}$ from P. The distribution $P_{\mathcal{D}}^{(t+1)}$ is in the middle between the two nets $W_{\mathcal{G}}^{(t)}$ and $W_{\mathcal{D}}^{(t)}$, and it serves as the data distribution to train the generator, i.e., we project this distribution onto the manifold $\mathcal{G} = \{P_{\mathcal{G}}(Y; W_{\mathcal{G}}), \forall W_{\mathcal{G}}\} = \{W_{\mathcal{G}}\}$ (recall we use $W_{\mathcal{G}}$ to denote the distribution $P_{\mathcal{G}}(Y; W_{\mathcal{G}})$) in the

information geometry picture, so that

$$W_{\mathcal{G}}^{(t+1)} = \arg\min_{\mathcal{G}} \mathrm{KL}(P_{\mathcal{D}}^{(t+1)}|W_{\mathcal{G}}). \tag{13}$$

The learning process alternates between Markov transition in (12) and projection in (13), as illustrated by Figure 9.

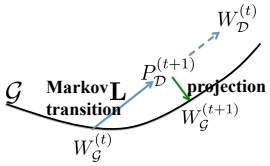


Figure 9: The learning of the generator alternates between Markov transition and projection. The family of the generator models $\mathcal{G}$ is illustrated by the black curve. Each distribution is illustrated by a point.

In the case of $l_{\mathcal{D}} \to \infty$,

$$W_{\mathcal{D}}^{(t)} \to \hat{W}_{\mathcal{D}} = \arg\min_{\mathcal{D}} \mathrm{KL}(P_{\mathrm{data}}|W_{\mathcal{D}}), \tag{14}$$

$$W_{\mathcal{G}}^{(t)} \to \hat{W}_{\mathcal{G}} = \arg\min_{\mathcal{G}} \mathrm{KL}(\hat{W}_{\mathcal{D}}|W_{\mathcal{G}}). \tag{15}$$

That is, we first project P_{data} onto $\mathcal{D}$, and from there continue to project onto $\mathcal{G}$. Therefore, $W_{\mathcal{D}}$ converges to the maximum likelihood estimate with P_{data} being the data distribution, while $W_{\mathcal{G}}$ converges to the maximum likelihood estimate with $\hat{W}_{\mathcal{D}}$ serving as the data distribution.

For finite $l_{\mathcal{D}}$, the algorithm may converge to the following fixed points. The fixed point for the generator satisfies

$$\hat{W}_{\mathcal{G}} = \arg\min_{\mathcal{G}} \mathrm{KL}(L_{\hat{W}_{\mathcal{D}}} \cdot \hat{W}_{\mathcal{G}}|W_{\mathcal{G}}). \tag{16}$$

The fixed point for the descriptor satisfies

$$\hat{W}_{\mathcal{D}} \quad = \quad \arg\min_{\mathcal{D}} \big[\mathrm{KL}(P_{\mathrm{data}}|W_{\mathcal{D}}) - \mathrm{KL}(\mathbf{L}_{\hat{W}_{\mathcal{D}}} \cdot \hat{W}_{\mathcal{G}}|W_{\mathcal{D}})\big], \tag{17}$$

which is similar to contrastive divergence (Hinton, 2002), except that $\hat{W}_G$ takes the place of P_{data} in the second Kullback-Leibler divergence. Because $\hat{W}_{\mathcal{G}}$ is supposed to be close to $\hat{W}_{\mathcal{D}}$, the second Kullback-Leibler divergence is supposed to be small, hence our algorithm is closer to maximum likelihood learning than contrastive divergence.

Kim & Bengio (2016) learned the generator by gradient descent on $\mathrm{KL}(W_{\mathcal{G}}|W_{\mathcal{D}}^{(t)})$ over $\mathcal{G}$. The objective function is $\mathrm{KL}(W_{\mathcal{G}}|W_{\mathcal{D}}^{(t)}) = \mathrm{E}_{W_{\mathcal{G}}}[\log P_G(Y; W_{\mathcal{G}})] - \mathrm{E}_{W_{\mathcal{G}}}[\log P_{\mathcal{D}}(Y; W_{\mathcal{D}}^{(t)})]$, where the first term is the negative entropy that is intractable, and the second term is the expected energy that is tractable. Our learning method for the generator is consistent with the learning objective $\mathrm{KL}(W_{\mathcal{G}}|W_{\mathcal{D}}^{(t)})$, because

$$\mathrm{KL}(P_{\mathcal{D}}^{(t+1)}|W_{\mathcal{D}}^{(t)}) \leq \mathrm{KL}(W_{\mathcal{G}}^{(t)}|W_{\mathcal{D}}^{(t)}). \tag{18}$$

In fact, $\mathrm{KL}(P_{\mathcal{D}}^{(t+1)}|W_{\mathcal{D}}^{(t)}) \to 0$ monotonically as $l_{\mathcal{D}} \to \infty$ due to the second law of thermodynamics. The reduction of the Kullback-Leibler divergence in (18) and the projection in (13) in our learning of the generator are consistent with the learning objective of reducing $\mathrm{KL}(W_{\mathcal{G}}|W_{\mathcal{D}}^{(t)})$ in Kim & Bengio (2016). But the Monte Carlo implementation of $\mathbf{L}$ in our work avoids the need to approximate the intractable entropy term.

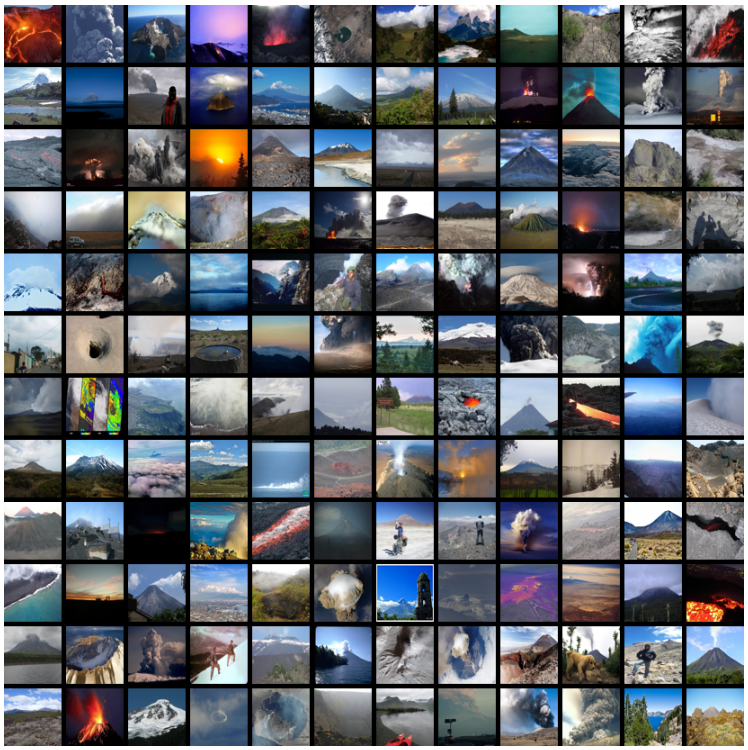
7.3 More synthesis results

We display more synthesis results at the resolution of 64×64.

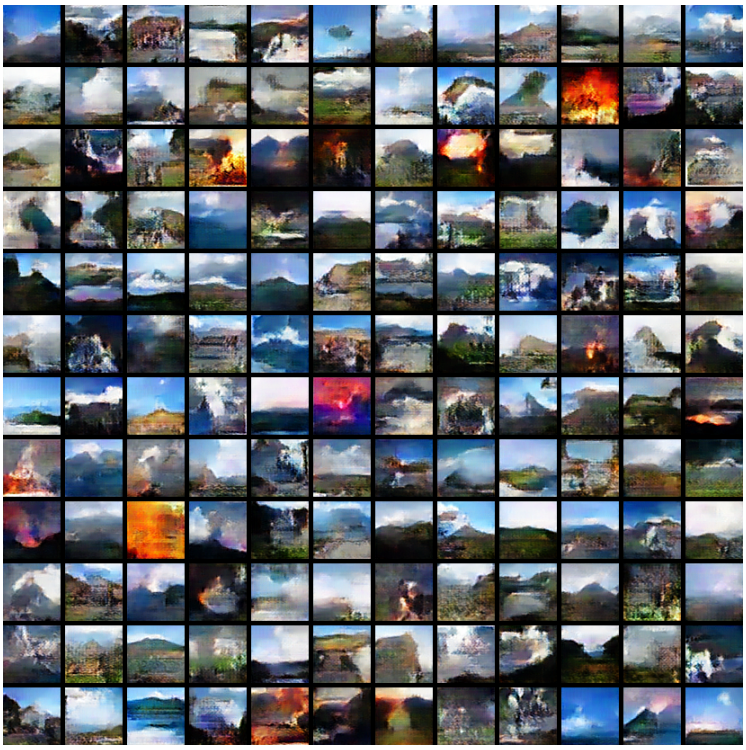
(a) Original images

(b) Synthesized images

Figure 10: Generating swimming pool images. The category is from MIT places205 dataset.

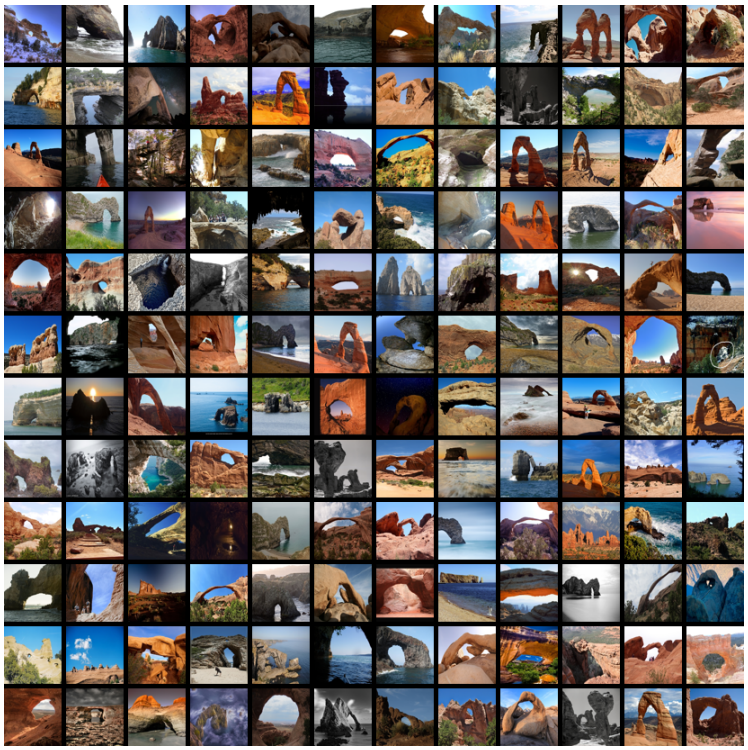


(a) Original images

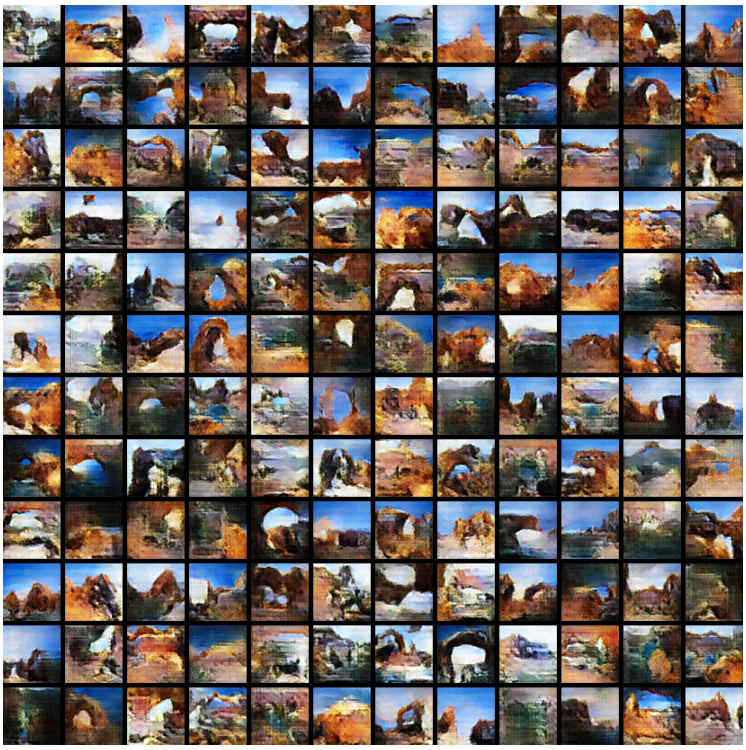


(b) Synthesized images

Figure 11: Generating volcano images. The category is from MIT places205 dataset.

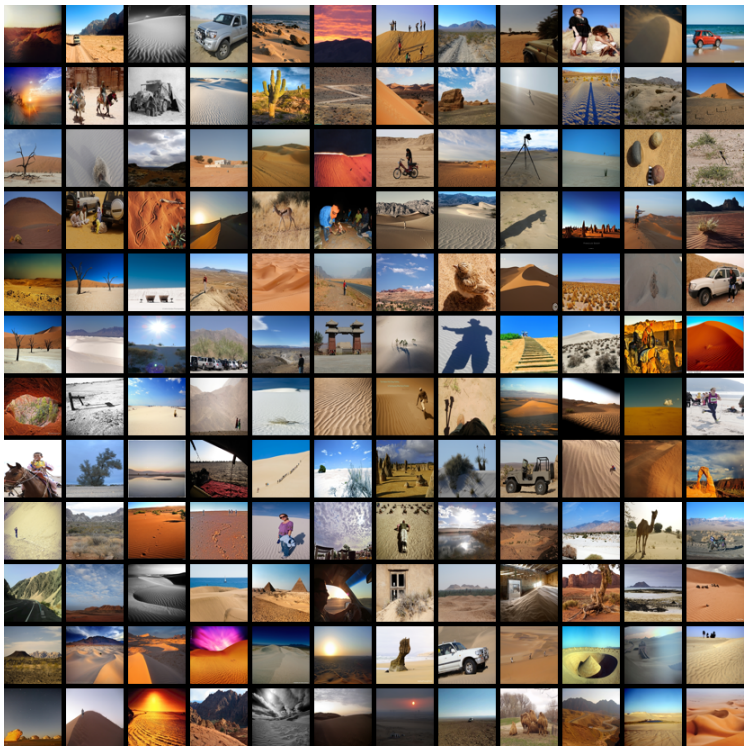


(a) Original images

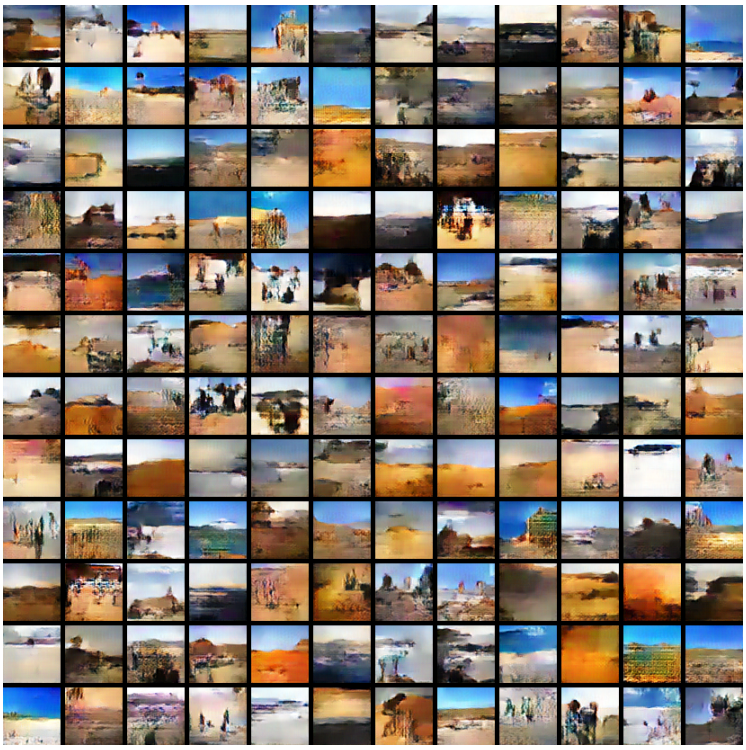


(b) Synthesized images

Figure 12: Generating rock images. The category is from MIT places205 dataset.



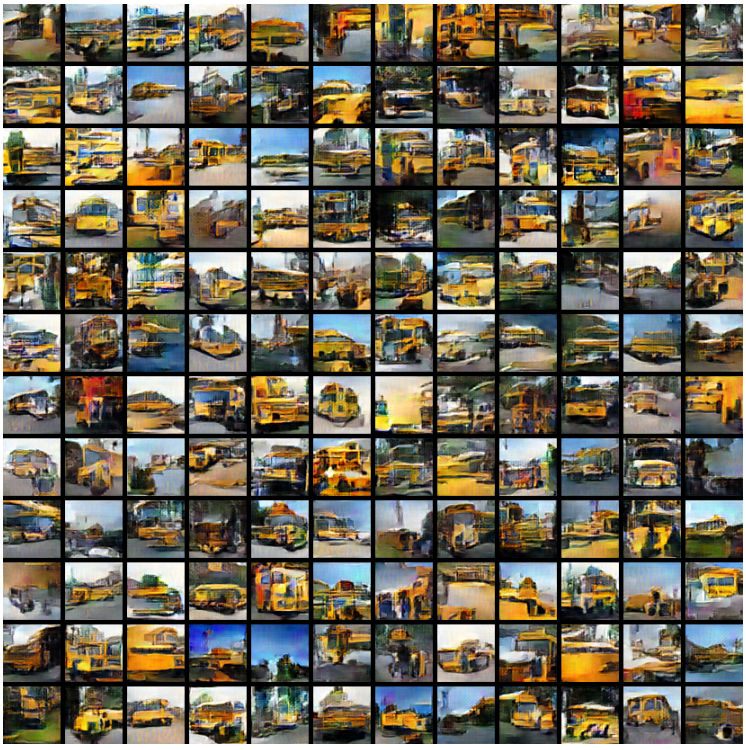
(a) Original images



(b) Synthesized images

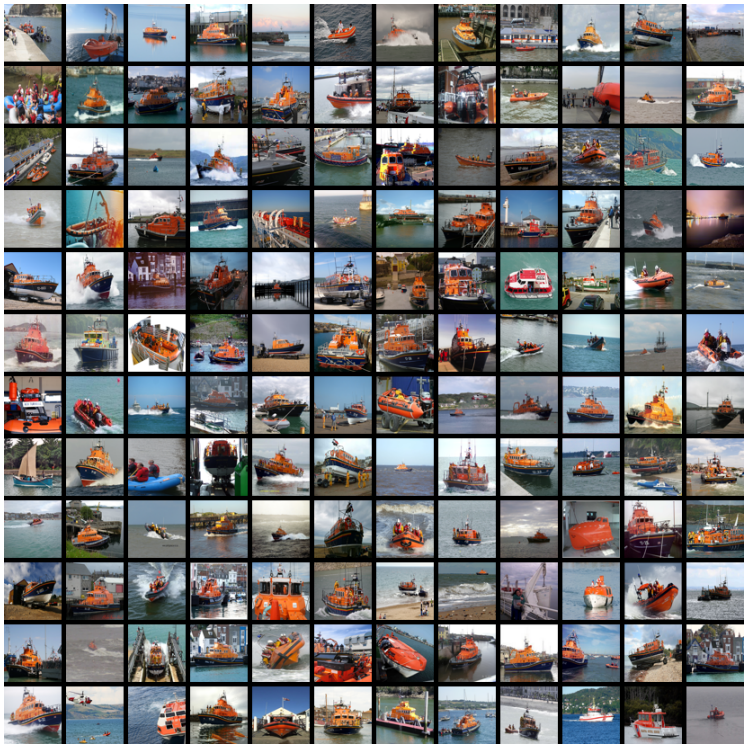
Figure 13: Generating desert images. The category is from MIT places205 dataset.

(a) Original images

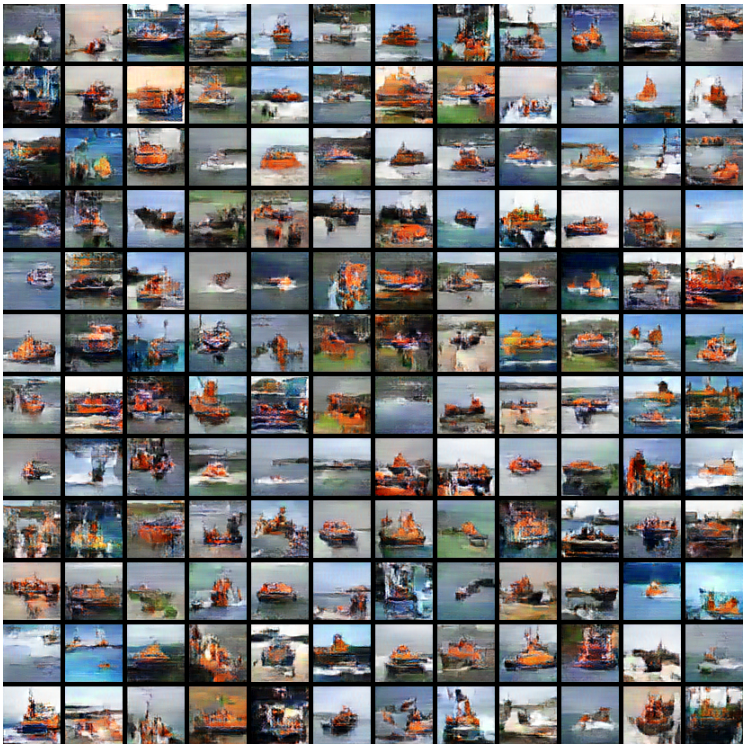


(b) Synthesized images

Figure 14: Generating schoolbus images. The category is from Imagenet ILSVRC2012 1000 object categories.



(a) Original images



(b) Synthesized images

Figure 15: Generating lifeboat images. The category is from Imagenet ILSVRC2012 1000 object categories.

(a) Original images

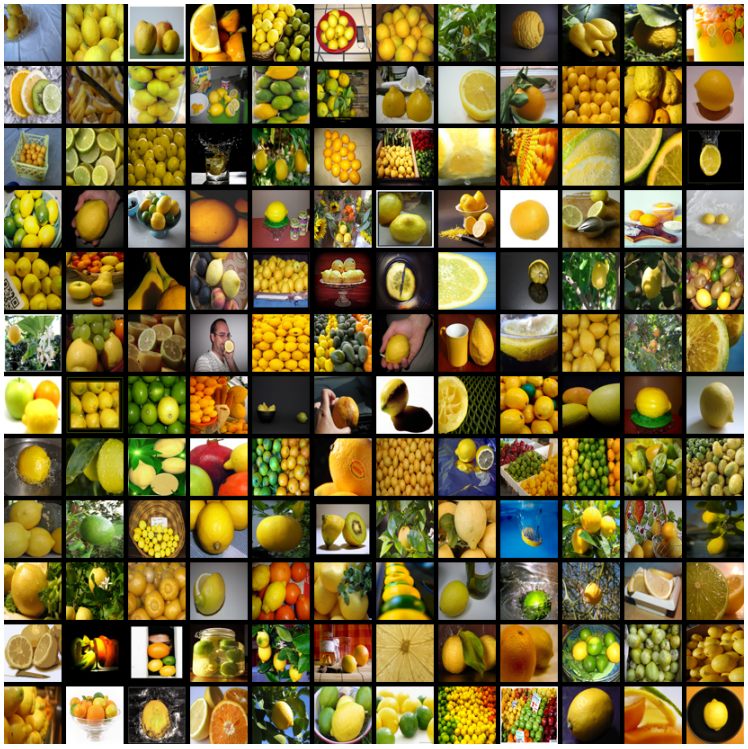
(b) Synthesized images

Figure 16: Generating zebra images. The category is from Imagenet ILSVRC2012 1000 object categories.

(a) Original images

(b) Synthesized images

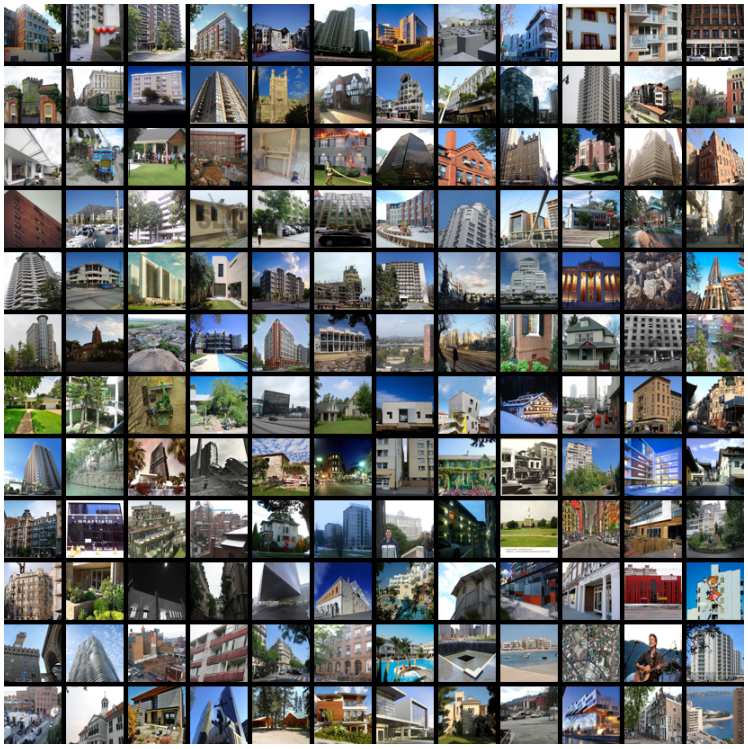
Figure 17: Generating strawberry images. The category is from Imagenet ILSVRC2012 1000 object categories.



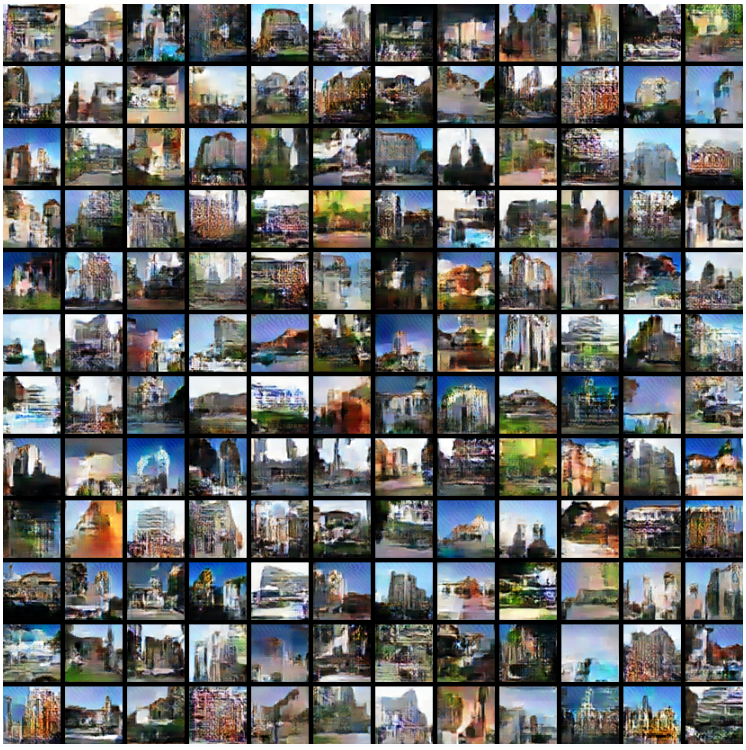
(a) Original images

(b) Synthesized images

Figure 18: Generating lemon images. The category is from Imagenet ILSVRC2012 1000 object categories.

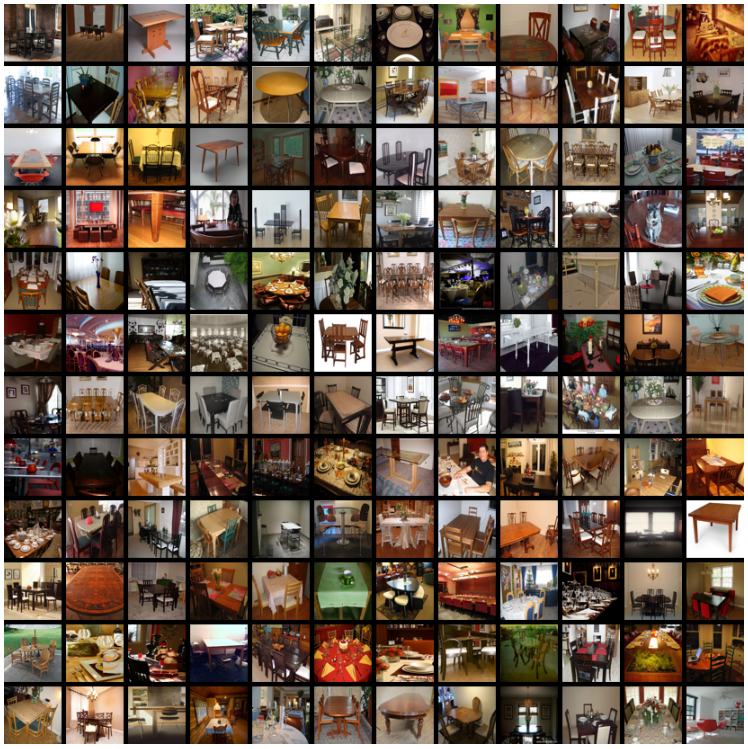


(a) Original images

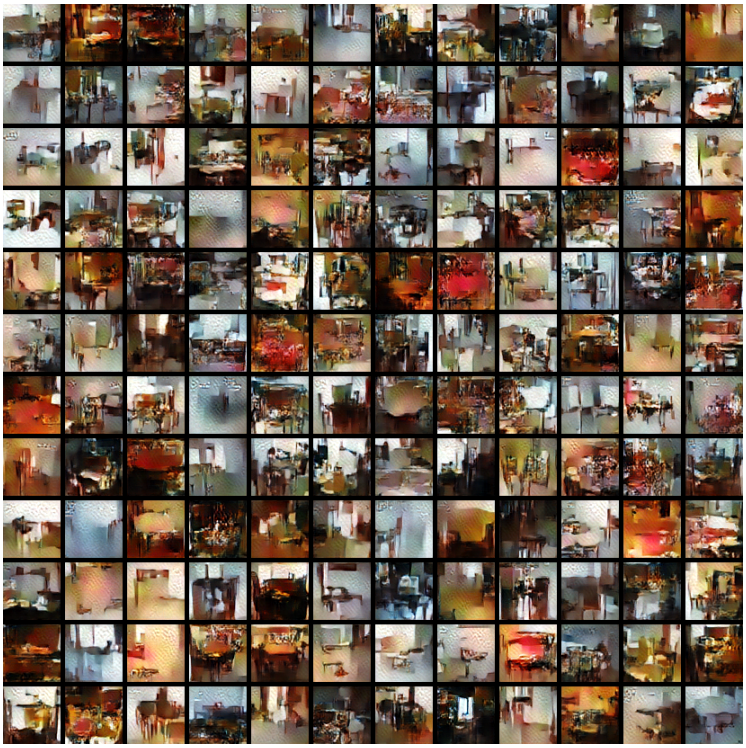


(b) Synthesized images

Figure 19: Generating apartment building images. The category is from Imagenet ILSVRC2012 1000 object categories.

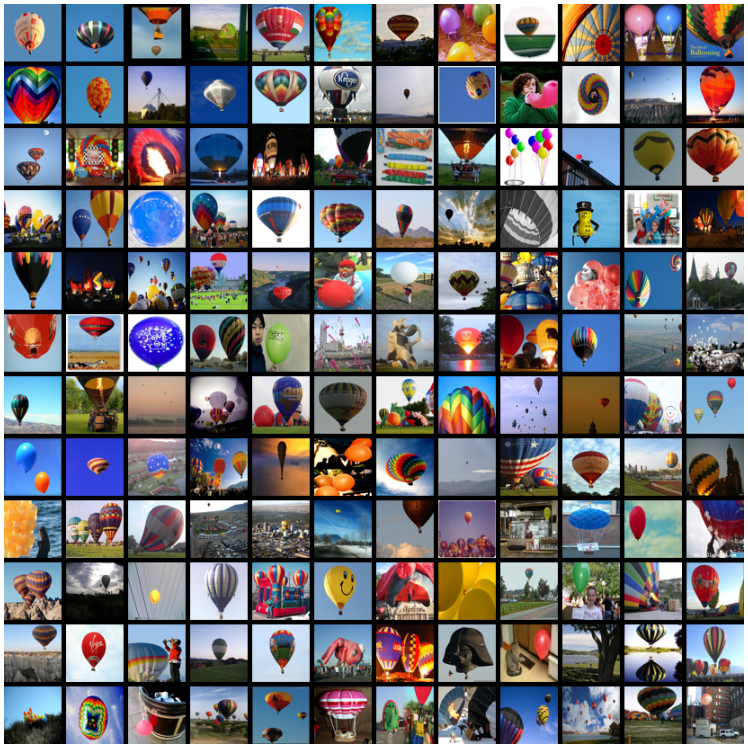


(a) Original images

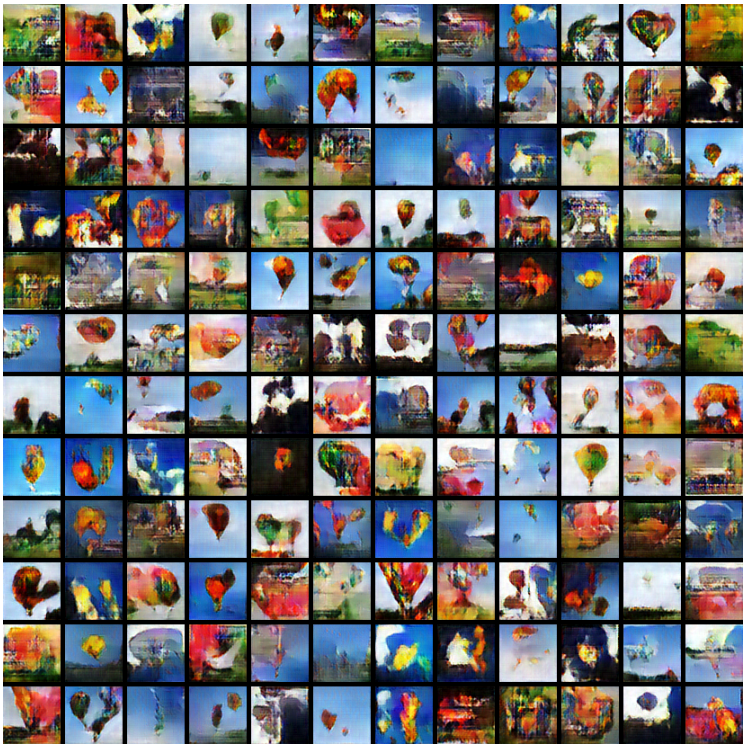


(b) Synthesized images

Figure 20: Generating dinning table images. The category is from Imagenet ILSVRC2012 1000 object categories.



(a) Original images



(b) Synthesized images

Figure 21: Generating balloon images. The category is from Imagenet ILSVRC2012 1000 object categories.

ACKNOWLEDGEMENT

We thank Hansheng Jiang for her work on this project as a summer visiting student. We thank Tian Han for sharing the code on learning the generator network, and for helpful discussions.

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

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. IEEE, 2009.

Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems*, pp. 1486–1494, 2015.

John D'Errico. Interpolation inpainting, 2004. URL `https://www.mathworks.com/matlabcentral/fileexchange/4551-inpaint-nans`.

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

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*, pp. 2672–2680, 2014.

Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Alternating back-propagation for generator network. In *31st AAAI Conference on Artificial Intelligence*, 2017.

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

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

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pp. 1097–1105, 2012.

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

Yann LeCun, Sumit Chopra, Rata Hadsell, Mare'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3730–3738, 2015.

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *ICML*, 2014.

Jiquan Ngiam, Zhenghao Chen, Pang Wei Koh, and Andrew Y. Ng. Learning deep energy models. In *International Conference on Machine Learning*, 2011.

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

Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In Tony Jebara and Eric P. Xing (eds.), *ICML*, pp. 1278–1286. JMLR Workshop and Conference Proceedings, 2014.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec):1235–1260, 2003.

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

Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. A theory of generative convnet. In *ICML*, 2016.

Laurent Younes. 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, 1999.

Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pp. 487–495, 2014.

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

Song-Chun Zhu, Ying Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9(8):1627–1660, 1997.