

# GENERATIVE MODELING OF CONVOLUTIONAL NEURAL NETWORKS

**Jifeng Dai**

Microsoft Research

jifdai@microsoft.com

**Ying-Nian Wu**

University of California, Los Angeles

ywu@stat.ucla.edu

## ABSTRACT

The convolutional neural networks (CNNs) have proven to be a powerful tool for discriminative learning. Recently researchers have also started to show interest in the generative aspects of CNNs in order to gain a deeper understanding of what they have learned and how to further improve them. This paper investigates generative modeling of CNNs. The main contributions include: (1) We construct a generative model for the CNN in the form of exponential tilting of a reference distribution. (2) We propose a generative gradient for pre-training CNNs by a non-parametric importance sampling scheme, which is fundamentally different from the commonly used discriminative gradient, and yet has the same computational architecture and cost as the latter. (3) We propose a generative visualization method for the CNNs by sampling from an explicit parametric image distribution. The proposed visualization method can directly draw synthetic samples for any given node in a trained CNN by the Hamiltonian Monte Carlo (HMC) algorithm, without resorting to any extra hold-out images. Experiments on the challenging ImageNet benchmark show that the proposed generative gradient pre-training consistently helps improve the performances of CNNs, and the proposed generative visualization method generates meaningful and varied samples of synthetic images from a large-scale deep CNN.

## 1 INTRODUCTION

Recent years have witnessed the triumphant return of the feedforward neural networks, especially the convolutional neural networks (CNNs) (LeCun et al., 1989). Fueled by the availabilities of large labeled data sets, the increased computing power, and the incorporation of new types of non-linear units, CNNs have proven to be a powerful tool for discriminative learning (Krizhevsky et al., 2012; Girshick et al., 2013). Despite the successes of the discriminative learning of CNNs, the generative aspect of CNNs has not been thoroughly investigated. But it can be very useful for the following reasons: (1) The generative pre-training has the potential to lead the network to a better local optimum; (2) Samples can be drawn from the generative model to reveal the knowledge learned by the CNN. Although many generative models and learning algorithms have been proposed (Hinton et al., 2006a;b; Rifai et al., 2011; Salakhutdinov & Hinton, 2009), most of them do not scale well and have not been applied to learning large scale deep CNNs.

In this paper, we study the generative modeling of the CNNs. We start from defining probability distributions of images given the underlying object categories or class labels, such that the CNN with a final logistic regression layer serves as the corresponding conditional distribution of the class labels given the images. These distributions are in the form of exponential tilting of a reference distribution, i.e., exponential family models or energy-based models relative to a reference distribution.

With such a generative model, we proceed to study it along two related themes, which differ in how to handle the reference distribution or the null model. In the first scheme, we propose a non-parametric generative gradient for pre-training the CNN, where the CNN is learned by the stochastic

gradient algorithm that seeks to minimize the log-likelihood of the generative model. The gradient of the log-likelihood is approximated by the importance sampling method that keeps reweighing the images that are sampled from a non-parametric implicit reference distribution, such as the marginal distribution of all the training images. The generative gradient is fundamentally different from the commonly used discriminative gradient, and yet in batch training, it shares the same computational architecture as well as computational cost as the discriminative gradient. This generative learning scheme can be used in a pre-training stage that is to be followed by the usual discriminative training. The generative log-likelihood provides stronger driving force than the discriminative criteria for stochastic gradient by requiring the learned parameters to explain the images instead of their labels. Experiments on the MNIST (LeCun et al., 1998) and the challenging ImageNet (Deng et al., 2009) classification benchmarks show that this generative pre-training scheme consistently helps improve the performance of CNNs.

The second theme in our study of generative modeling is to assume an explicit parametric form of the reference distribution, such as the Gaussian white noise model, so that we can draw synthetic images from the resulting probability distributions of images. The sampling can be accomplished by the Hamiltonian Monte Carlo (HMC) algorithm (Neal, 2011), which iterates between a bottom-up convolution step and a top-down deconvolution step. The proposed visualization method can directly draw samples of synthetic images for any given node in a trained CNN, without resorting to any extra hold-out images. Experiments show that vividly meaningful and diversely varied synthetic images can be generated for nodes of a large-scale deep CNN discriminatively trained on ImageNet.

## 2 PAST WORK

The generative model that we study is an energy-based model. Such models include field of experts (Roth & Black, 2009), product of experts (Hinton, 2002), Boltzmann machines (Hinton et al., 2006a), model based on neural networks (Hinton et al., 2006b), etc. However, most of these generative models and learning algorithms have not been applied to learning deep CNNs. The proposed non-parametric generative gradient as well as the generative model based on deep CNNs have not been studied in the literature to the best of our knowledge.

The relationship between the generative models and the discriminative approaches has been extensively studied, perhaps starting from Efron (Efron, 1975), and more recently by (Jordan, 2002; Liang & Jordan, 2008), etc. Moreover, the usefulness of generative pre-training for deep learning has been studied by (Erhan et al., 2010) etc. However, this issue has not been thoroughly investigated for CNNs. Moreover, unlike previous pre-training methods, our non-parametric generative gradient shares the same computational architecture as the discriminative gradient.

As to visualization, our work is related to (Erhan et al., 2009; Le et al., 2012; Girshick et al., 2013; Zeiler & Fergus, 2013; Long et al., 2014). In Girshick et al. (2013); Long et al. (2014), the high-scoring image patches are directly presented. In Zeiler & Fergus (2013), a top-down deconvolution step is used to understand what contents are emphasized in the high-scoring input image patches. Visualization of nodes in neural networks has also been studied in Erhan et al. (2009); Le et al. (2012), where images are synthesized by maximizing the responses of the nodes. In our approach, a generative model is formally defined. We sample from the well-defined probability distribution by the HMC algorithm, generating meaningful and varying synthetic images, without resorting to a large collection of hold-out images (Girshick et al., 2013; Zeiler & Fergus, 2013; Long et al., 2014).

## 3 GENERATIVE MODEL BASED ON CNN

### 3.1 PROBABILITY DISTRIBUTIONS ON IMAGES

Suppose we observe images from many different object categories. Let  $x$  be an observed image from an object category  $y$ . Consider the following probability distribution on  $x$ ,

$$p_y(x; w) = \frac{1}{Z_y(w)} \exp(f_y(x; w)) q(x), \quad (1)$$

where  $q(x)$  is a reference distribution common to all the categories,  $f_y(x; w)$  is a scoring function for class  $y$ ,  $w$  collects the unknown parameters to be learned from the data, and  $Z_y(w) =$

$E_q[\exp(f_y(x; w))] = \int \exp(f_y(x; w))q(x)dx$  is the normalizing constant or partition function. The distribution  $p_y(x; w)$  is in the form of an exponential tilting of the reference distribution  $q(x)$ , and can be considered an energy-based model or an exponential family model. In Model (1), the reference distribution  $q(x)$  may not be unique. If we change  $q(x)$  to  $q_1(x)$ , then we can change  $f_y(x; w)$  to  $f_y(x; w) - \log[q_1(x)/q(x)]$ , which may correspond to a  $f_y(x; w_1)$  for a different  $w_1$  if the parametrization of  $f_y(x; w)$  is flexible enough. We want to choose  $q(x)$  so that either  $q(x)$  is reasonably close to  $p_y(x; w)$  as in our non-parametric generative gradient method, or the resulting  $p_y(x; w)$  based on  $q(x)$  is easy to sample from as in our generative visualization method.

For an image  $x$ , let  $y$  be the underlying object category or class label, so that  $p(x|y; w) = p_y(x; w)$ . Suppose the prior distribution on  $y$  is  $p(y) = \rho_y$ . The posterior distribution of  $y$  given  $x$  is

$$p(y|x, w) = \frac{\exp(f_y(x; w) + \alpha_y)}{\sum_y \exp(f_y(x; w) + \alpha_y)}, \quad (2)$$

where  $\alpha_y = \log \rho_y - \log Z_y(w)$ .  $p(y|x, w)$  is in the form of a multi-class logistic regression, where  $\alpha_y$  can be treated as an intercept parameter to be estimated directly if the model is trained discriminatively. Thus for notational simplicity, we shall assume that the intercept term  $\alpha_y$  is already absorbed into  $w$  for the rest of the paper. Note that  $f_y(x; w)$  is not unique in (2). If we change  $f_y(x; w)$  to  $f_y(x; w) - g(x)$  for a  $g(x)$  that is common to all the categories, we still have the same  $p(y|x; w)$ . This non-uniqueness corresponds to the non-uniqueness of  $q(x)$  in (1) mentioned above.

Given a set of labeled data  $\{(x_i, y_i)\}$ , equations (1) and (2) suggest two different methods to estimate the parameters  $w$ . One is to maximize the generative log-likelihood  $l_G(w) = \sum_i \log p(x_i|y_i, w)$ , which is the same as maximizing the full log-likelihood  $\sum_i \log p(x_i, y_i|w)$ , where the prior probability of  $\rho_y$  can be estimated by class frequency of category  $y$ . The other is to maximize the discriminative log-likelihood  $l_D(w) = \sum_i \log p(y_i|x_i, w)$ . For the discriminative model (2), a popular choice of  $f_y(x; w)$  is multi-layer perceptron or CNN, with  $w$  being the connection weights, and the top-layer is a multi-class logistic regression. This is the choice we adopt throughout this paper.

### 3.2 GENERATIVE GRADIENT

The gradient of the discriminative log-likelihood is calculated according to

$$\frac{\partial}{\partial w} \log p(y_i|x_i, w) = \frac{\partial}{\partial w} f_{y_i}(x_i; w) - E_D \left[ \frac{\partial}{\partial w} f_y(x_i; w) \right], \quad (3)$$

where  $\alpha_y$  is absorbed into  $w$  as mentioned above, and

$$E_D \left[ \frac{\partial}{\partial w} f_y(x_i; w) \right] = \sum_y \frac{\partial}{\partial w} f_y(x_i; w) \frac{\exp(f_y(x_i; w))}{\sum_y \exp(f_y(x_i; w))}. \quad (4)$$

The gradient of the generative log-likelihood is calculated according to

$$\frac{\partial}{\partial w} \log p_{y_i}(x_i; w) = \frac{\partial}{\partial w} f_{y_i}(x_i; w) - E_G \left[ \frac{\partial}{\partial w} f_{y_i}(x; w) \right], \quad (5)$$

where

$$E_G \left[ \frac{\partial}{\partial w} f_{y_i}(x; w) \right] = \int \frac{\partial}{\partial w} f_{y_i}(x; w) \frac{1}{Z_{y_i}(w)} \exp(f_{y_i}(x; w))q(x), \quad (6)$$

which can be approximated by importance sampling. Specifically, let  $\{\tilde{x}_j\}_{j=1}^m$  be a set of samples from  $q(x)$ , for instance,  $q(x)$  is the distribution of images from all the categories. Here we do not attempt to model  $q(x)$  parametrically, instead, we treat it as an implicit non-parametric distribution. Then by importance sampling,

$$E_G \left[ \frac{\partial}{\partial w} f_{y_i}(x; w) \right] \approx \sum_j \frac{\partial}{\partial w} f_{y_i}(\tilde{x}_j; w) W_j, \quad (7)$$

where the importance weight  $W_j \propto \exp(f_{y_i}(\tilde{x}_j; w))$  and is normalized to have sum 1. Namely,

$$\frac{\partial}{\partial w} \log p_{y_i}(x_i; w) \approx \frac{\partial}{\partial w} f_{y_i}(x_i; w) - \sum_j \frac{\partial}{\partial w} f_{y_i}(\tilde{x}_j; w) \frac{\exp(f_{y_i}(\tilde{x}_j; w))}{\sum_k \exp(f_{y_i}(\tilde{x}_k; w))}. \quad (8)$$

The discriminative gradient and the generative gradient differ subtly and yet fundamentally in calculating  $E[\partial f_y(x; w)/\partial w]$ , whose difference from the observed  $\partial f_{y_i}(x_i; w)/\partial w$  provides the driving force for updating  $w$ . In the discriminative gradient, the expectation is with respect to the posterior distribution of the class label  $y$  while the image  $x_i$  is fixed, whereas in the generative gradient, the expectation is with respect to the distribution of the images  $x$  while the class label  $y_i$  is fixed. In general, it is easier to adjust the parameters  $w$  to predict the class labels than to reproduce the features of the images. So it is expected that the generative gradient provides stronger driving force for updating  $w$ .

The non-parametric generative gradient can be especially useful in the beginning stage of training or what can be called pre-training, where  $w$  is small, so that the current  $p_y(x; w)$  for each category  $y$  is not very separated from  $q(x)$ , which is the overall marginal distribution of  $x$ . In this stage, the importance weights  $W_j$  are not very skewed and the effective sample size for importance sampling can be large. So updating  $w$  according to the generative gradient can provide useful pre-training with the potential to lead  $w$  to a good local optimum. If the importance weights  $W_j$  start to become skewed and the effective sample size starts to dwindle, then this indicates that the categories  $p_y(x; w)$  start to separate from  $q(x)$  as well as from each other, so we can switch to discriminative training to further separate the categories. More explanation of the generative gradient can be found in the supplementary materials.

### 3.3 BATCH TRAINING AND GENERATIVE LOSS LAYER

At first glance, the generative gradient appears computationally expensive due to the need to sample from  $q(x)$ . In fact, with  $q(x)$  being the collection of images from all the categories, we may use each batch of samples as an approximation to  $q(x)$  in the batch training mode.

Specifically, let  $\{(x_i, y_i)\}_{i=1}^n$  be a batch set of training examples, and we seek to maximize  $\sum_i \log p_{y_i}(x_i; w)$  via generative gradient. In the calculation of  $\partial \log p_{y_i}(x_i; w)/\partial w$ ,  $\{x_j\}_{j=1}^n$  can be used as samples from  $q(x)$ . In this way, the computational cost of the generative gradient is about the same as that of the discriminative gradient.

Moreover, the computation of the generative gradient can be induced to share the same back propagation architecture as the discriminative gradient. Specifically, the calculation of the generative gradient can be decoupled into the calculation at a new generative loss layer and the calculation at lower layers. To be more specific, by replacing  $\{\tilde{x}_j\}_{j=1}^n$  in (8) by the batch sample  $\{x_j\}_{j=1}^n$ , we can rewrite (8) in the following form:

$$\frac{\partial}{\partial w} \log p_{y_i}(x_i; w) \approx \sum_{y,j} \frac{\partial \log p_{y_i}(x_i; w)}{\partial f_y(x_j; w)} \frac{\partial f_y(x_j; w)}{\partial w}, \quad (9)$$

where  $\partial \log p_{y_i}(x_i; w)/\partial f_y(x_j; w)$  is called the generative loss layer (to be defined below, with  $f_y(x_j; w)$  being treated here as a variable in the chain rule), while the calculation of  $\partial f_y(x_j; w)/\partial w$  is exactly the same as that in the discriminative gradient. This decoupling brings simplicity to programming.

We use the notation  $\partial \log p_{y_i}(x_i; w)/\partial f_y(x_j; w)$  for the top generative layer mainly to make it conformal to the chain rule calculation. According to (8),  $\partial \log p_{y_i}(x_i; w)/\partial f_y(x_j; w)$  is defined by

$$\frac{\partial \log p_{y_i}(x_i; w)}{\partial f_y(x_j; w)} = \begin{cases} 0 & y \neq y_i; \\ 1 - \frac{\exp(f_{y_i}(x_j; w))}{\sum_k \exp(f_{y_i}(x_k; w))} & y = y_i, j = i; \\ -\frac{\exp(f_{y_i}(x_j; w))}{\sum_k \exp(f_{y_i}(x_k; w))} & y = y_i, j \neq i. \end{cases} \quad (10)$$

During training, on a batch of training examples,  $\{(x_i, y_i), i = 1, \dots, n\}$ , the generative log-likelihood is

$$l_G(w) = \sum_i \log p(x_i|y_i, w) = \sum_i \log \frac{\exp(f_{y_i}(x_i; w))}{Z_{y_i}(w)} \approx \sum_i \log \frac{\exp(f_{y_i}(x_i; w))}{\sum_i \exp(f_{y_i}(x_i; w)) / n}.$$

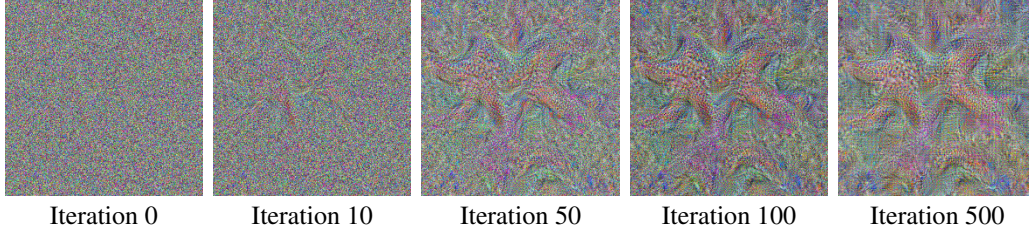


Figure 1: The sequence of images produced by HMC that samples from the “Starfish, sea star” category of the “AlexNet” network (Krizhevsky et al., 2012) discriminatively trained on ImageNet ILSVRC-2012. The figures in this paper are best viewed in color.

The gradient with respect to  $w$  is

$$l'_G(w) = \sum_i \left[ \frac{\partial}{\partial w} f_{y_i}(x_i; w) - \sum_j \frac{\partial}{\partial w} f_{y_i}(x_j; w) \frac{\exp(f_{y_i}(x_j; w))}{\sum_k \exp(f_{y_i}(x_k; w))} \right].$$

While the discriminative log-likelihood is

$$l_D(w) = \sum_i \log p(y_i | x_i, w) = \sum_i \log \frac{\exp(f_{y_i}(x_i; w))}{\sum_y \exp(f_y(x_i; w))}.$$

The gradient with respect to  $w$  is

$$l'_D(w) = \sum_i \left[ \frac{\partial}{\partial w} f_{y_i}(x_i; w) - \sum_y \frac{\partial}{\partial w} f_y(x_i; w) \frac{\exp(f_y(x_i; w))}{\sum_y \exp(f_y(x_i; w))} \right].$$

$l'_D$  and  $l'_G$  are similar in form and different in the summation operations. In  $l'_D$ , the summation is over category  $y$  while  $x_i$  is fixed, whereas in  $l'_G$ , the summation is over example  $x_j$  while  $y_i$  is fixed.

In the generative gradient, we want  $f_{y_i}$  to assign high score to  $x_i$  as well as those observations that belong to  $y_i$ , but assign low scores to those observations that do not belong to  $y_i$ . This constraint is for the *same*  $f_{y_i}$ , regardless of what other  $f_y$  do for  $y \neq y_i$ .

In the discriminative gradient, we want  $f_y(x_i)$  to work together for all *different*  $y$ , so that  $f_{y_i}$  assigns high score to  $x_i$  than other  $f_y$  for  $y \neq y_i$ .

Apparently, the discriminative constraint is weaker because it involves all  $f_y$ , and the generative constraint is stronger because it involves single  $f_y$ . After generative learning, these  $f_y$  are well behaved and then we can continue to adjust them (probably the intercepts for different  $y$ ) to satisfy the discriminative constraint.

### 3.4 GENERATIVE VISUALIZATION

Recently, people are interested in understanding what the machine learns. Suppose we care about the node at the top layer. The idea can be applied to the nodes at any layer.

We consider generating samples from  $p_y(x; w)$  with  $w$  already learned by discriminative training (or any other methods). For this purpose, we need to assume a parametric reference distribution  $q(x)$ , such as Gaussian white noise distribution, which is the maximum entropy distribution or the most featureless distribution fitted to the observed training images with normalized marginal variances. After discriminatively learning  $f_y(x; w)$  for all  $y$ , we can sample from the corresponding  $p_y(x; w)$  by Hamiltonian Monte Carlo (HMC) (Neal, 2011).

We can assume a parametric reference distribution  $q(x)$ , such as white noise distribution. We may include  $q(x)$  as a null category for discriminative learning. Then we can sample  $p_y(x; w)$  by Hamiltonian Monte Carlo. Specifically, we can write  $p_y(x; w)$  as  $p_y(x; w) \propto \exp(-U(x))$ , where  $U(x) = -f_y(x; w) + \frac{1}{2}|x|^2$  (assuming  $\sigma^2 = 1$ ). In physics context,  $x$  can be regarded as a

position vector and  $U(x)$  the potential energy function. To allow Hamiltonian dynamics to operate, we need to introduce an auxiliary momentum vector  $\phi$  and the corresponding kinetic energy function  $K(\phi) = |\phi|^2/2m$ , where  $m$  represents the mass. After that, a fictitious physical system described by the canonical coordinates  $(x, \phi)$  is defined, and its total energy is  $H(x, \phi) = U(x) + K(\phi)$ . Instead of sampling from  $p(x; w)$  directly, HMC samples from the joint canonical distribution  $p(x, \phi) \propto \exp(-H(x, \phi))$ , under which  $x \sim p(x)$  marginally and  $\phi$  follows a Gaussian distribution and is independent of  $x$ . Each time HMC draws a random sample from the marginal Gaussian distribution of  $\phi$ , and then evolve according to the Hamiltonian dynamics that conserves the total energy.

In practical implementation, the leapfrog algorithm (Neal, 2011) is used to discretize the continuous Hamiltonian dynamics as follows, with  $\epsilon$  being the step-size:

$$\phi^{(t+\epsilon/2)} = \phi^{(t)} - (\epsilon/2) \frac{\partial U}{\partial x}(x^{(t)}), \quad (11)$$

$$x^{(t+\epsilon)} = x^{(t)} + \epsilon \frac{\phi^{(t+\epsilon/2)}}{m}, \quad (12)$$

$$\phi^{(t+\epsilon)} = \phi^{(t+\epsilon/2)} - (\epsilon/2) \frac{\partial U}{\partial x}(x^{(t+\epsilon)}), \quad (13)$$

that is, a half-step update of  $\phi$  is performed first and then it is used to compute  $x^{(t+\epsilon)}$  and  $\phi^{(t+\epsilon)}$ . The discretization of the leapfrog algorithm cannot keep  $H(x, \phi)$  exactly constant, so a Metropolis acceptance/rejection stage is used to correct the discretization error.

A key step in the leapfrog algorithm is the computation of the derivative of the potential energy function  $\partial U/\partial x$ , which involves calculating  $\partial f_y(x; w)/\partial x$ . Note that the above derivative is with respect to  $x$ , not with respect to  $w$ . There are two types of structures in  $f_y(x; w)$ . One is rectified linear unit  $r = \max(\sum_i w_i a_i, 0)$ , where we incorporate the bias term into the sum with  $a_i = -1$ . Then  $\partial r/\partial a_i = w_i$  if  $r > 0$ , and  $\partial r/\partial a_i = 0$  if  $r = 0$ . The other structure is local max pooling unit with  $r = \max_i(a_i)$ . Then  $\partial r/\partial a_i = 1$  if  $a_i$  is the maximum, and  $\partial r/\partial a_i = 0$  otherwise. That is, the derivative is the arg-max un-pooling. Therefore the computation of  $\partial f_y(x; w)/\partial x$  involves two steps: (1) Bottom-up scoring for computing  $r$ , which consists of convolution steps and local max pooling steps. (2) Top-down derivative computation which involves deconvolution steps for taking derivatives of the rectified linear units and the arg-max un-pooling steps for taking derivatives of the local max pooling units. The derivative calculation is implemented within each leapfrog step. They are different from the scheme in Zeiler & Fergus (2013). The visualization sequence of a sample is shown in Fig. 1.

The above method can be adapted to visualize the nodes at lower layers too. We only need to use the corresponding scoring function  $f_y(x; w)$  for a node  $y$ .

## 4 EXPERIMENTS

### 4.1 GENERATIVE PRE-TRAINING

In generative pre-training experiments, three different training approaches are studied: i) discriminative gradient ( $DG$ ); ii) generative gradient ( $GG$ ); iii) generative gradient pre-training + discriminative gradient tuning ( $GG+DG$ ). We build algorithms on the code of Caffe (Jia et al., 2014) and the experiment settings are identical to Jia et al. (2014). Experiments are performed on two widely used image classification benchmarks: namely MNIST (LeCun et al., 1998) handwritten digit recognition and ImageNet ILSVRC-2012 (Deng et al., 2009) natural image classification.

**MNIST handwritten digit recognition.** We first study generative pre-training on the MNIST dataset. The ‘‘LeNet’’ network (LeCun et al., 1998) is utilized, which is default for MNIST in Caffe. Although higher accuracy can be achieved by utilizing deeper networks, random image distortion etc, here we stick to the baseline network for fair comparison and experimental efficiency. Network training and testing are performed on the *train* and *test* sets respectively. For all the three training approaches, stochastic gradient descent is performed in training with a batch size of 64, a base learning rate of 0.01, a weight decay term of 0.0005, a momentum term of 0.9, and a max epoch number of 25. For  $GG+DG$ , the pre-training stage stops after 16 epochs and the discriminative gradient tuning stage starts with a base learning rate of 0.003.

Table 1: Error rates on the MNIST *test* set of different training approaches utilizing the “LeNet” network (LeCun et al., 1998).

Training approaches	<i>DG</i>	<i>GG</i>	<i>GG+DG</i>
Error rates	1.03	0.85	<b>0.78</b>

Table 2: Top-1 classification error rates on the ImageNet ILSVRC-2012 *val* set of different training approaches.

Training approaches	<i>DG</i>	<i>GG</i>	<i>GG+DG</i>
AlexNet	40.7	45.8	<b>39.6</b>
ZeilerFergusNet (fast)	38.4	44.3	<b>37.4</b>

The experimental results are presented in Table 1. The error rate of LeNet trained by discriminative gradient is 1.03%. When trained by generative gradient, the error rate reduces to 0.85%. When generative gradient pre-training and discriminative gradient tuning are both applied, the error rate further reduces to 0.78%, which is 0.25% (24% relatively) lower than that of discriminative gradient.

**ImageNet ILSVRC-2012 natural image classification.** In experiments on ImageNet ILSVRC-2012, two networks are utilized, namely “AlexNet” (Krizhevsky et al., 2012) and “ZeilerFergusNet” (fast) (Zeiler & Fergus, 2013). Network training and testing are performed on the *train* and *val* sets respectively. In training, a single network is trained by stochastic gradient descent with a batch size of 256, a base learning rate of 0.01, a weight decay term of 0.0005, a momentum term of 0.9, and a max epoch number of 70. For *GG+DG*, the pre-training stage stops after 45 epochs and the discriminative gradient tuning stage starts with a base learning rate of 0.003. In testing, top-1 classification error rates are reported on the *val* set by classifying the center and the four corner crops of the input images.

As shown in Table 2, the error rates of discriminative gradient training applied on AlexNet and ZeilerFergusNet are 40.7% and 38.4% respectively. While the error rates of generative gradient are 45.8% and 44.3% respectively. Generative gradient pre-training followed by discriminative gradient tuning achieves error rates of 39.6% and 37.4% respectively, which are 1.1% and 1.0% lower than those of discriminative gradient.

Experiment results on MNIST and ImageNet ILSVRC-2012 show that generative gradient pre-training followed by discriminative gradient tuning consistently improves the classification accuracies for varying networks. At the beginning stage of training, updating network parameters according to the generative gradient provides useful pre-training, which leads the network parameters to a good local optimum.

As to the computational cost, generative gradient is on par with discriminative gradient. The computational cost of the generative loss layer itself is ignorable in the network compared to the computation at the convolutional layers and the fully-connected layers. The total epoch numbers of *GG+DG* is on par with that of *DG*. Better accuracies can be achieved by generative gradient pre-training without introducing additional computation overhead.

## 4.2 GENERATIVE VISUALIZATION

In the generative visualization experiments, we visualize the nodes of the LeNet network and the AlexNet network trained by discriminative gradient on MNIST and ImageNet ILSVRC-2012 respectively. The networks trained by generative gradient can be visualized by the same algorithm as well.

We first visualize the nodes at the final fully-connected layer of LeNet. In experiments, we delete the drop-out layer to avoid unnecessary noise for visualization. At the beginning of visualization,  $x$  is initialized from  $q(x) \propto \exp(-|x|^2/2)$ . The HMC iteration number, the leapfrog step size, the leapfrog step number, and the particle mass are set to be 100, 0.1, 50, and 0.01 respectively. The

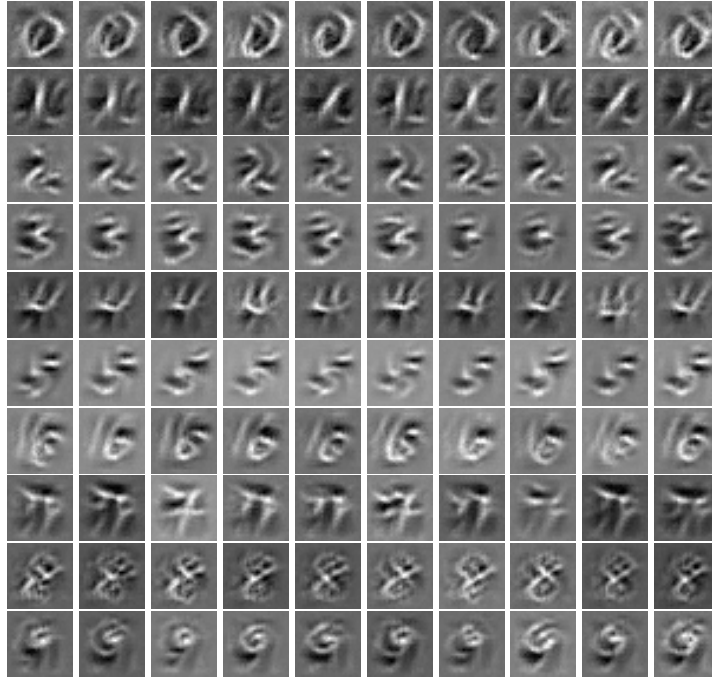


Figure 2: Samples from the nodes at the final fully-connected layer in the fully trained LeNet model, which correspond to different handwritten digits.

visualization results are shown in Fig. 2. The samples drawn from a node clearly correspond to the handwritten digit of the corresponding category, and are of varying shapes.

We further visualize the nodes in AlexNet, which is a much larger network compared to LeNet. Both nodes from the intermediate convolutional layers (conv1 to conv5) and the final fully-connected layer (fc8) are visualized. The leapfrog step size, the leapfrog step number, and the particle mass are set to be 1, 25, and 0.3 respectively. The HMC iteration numbers are 100 and 500 for nodes from the intermediate convolutional and the final fully-connected layers respectively.

The samples from the intermediate convolutional layers of AlexNet are shown in Fig. 3, while those from the final fully-connected layer are shown in Fig. 4 and Fig. 5. The HMC algorithm produces meaningful and varied samples, which vividly reveals what is learned by nodes at different hierarchies in the network. Note that such samples are generated by the HMC algorithm from the trained model directly, without using a large hold-out collection of images as in Girshick et al. (2013); Zeiler & Fergus (2013); Long et al. (2014).

As to the computational cost, it varies for nodes at different layers within different networks. On a desktop with GTX Titian, it takes about 0.4 minute to draw a sample for nodes at the final fully-connected layer of LeNet. In AlexNet, for nodes at the first convolutional layer and at the final fully-connected layer, it takes about 0.5 minute and 12 minute to draw a sample respectively.

## 5 DISCUSSION

Generative modeling is a fundamental problem of statistical learning. A good learning machine should have both strong discriminative and generative capacities. Given the recent successes of CNNs, it is worthwhile to explore their generative aspects. In this work, we show that a simple generative model can be constructed based on the CNN. The generative model helps to pre-train the CNN. It also helps to visualize the knowledge of the learned CNN.

The proposed scheme for visualizing a node by assuming a white noise reference distribution can be easily turned into a learning algorithm where the expectation in the generative gradient can be



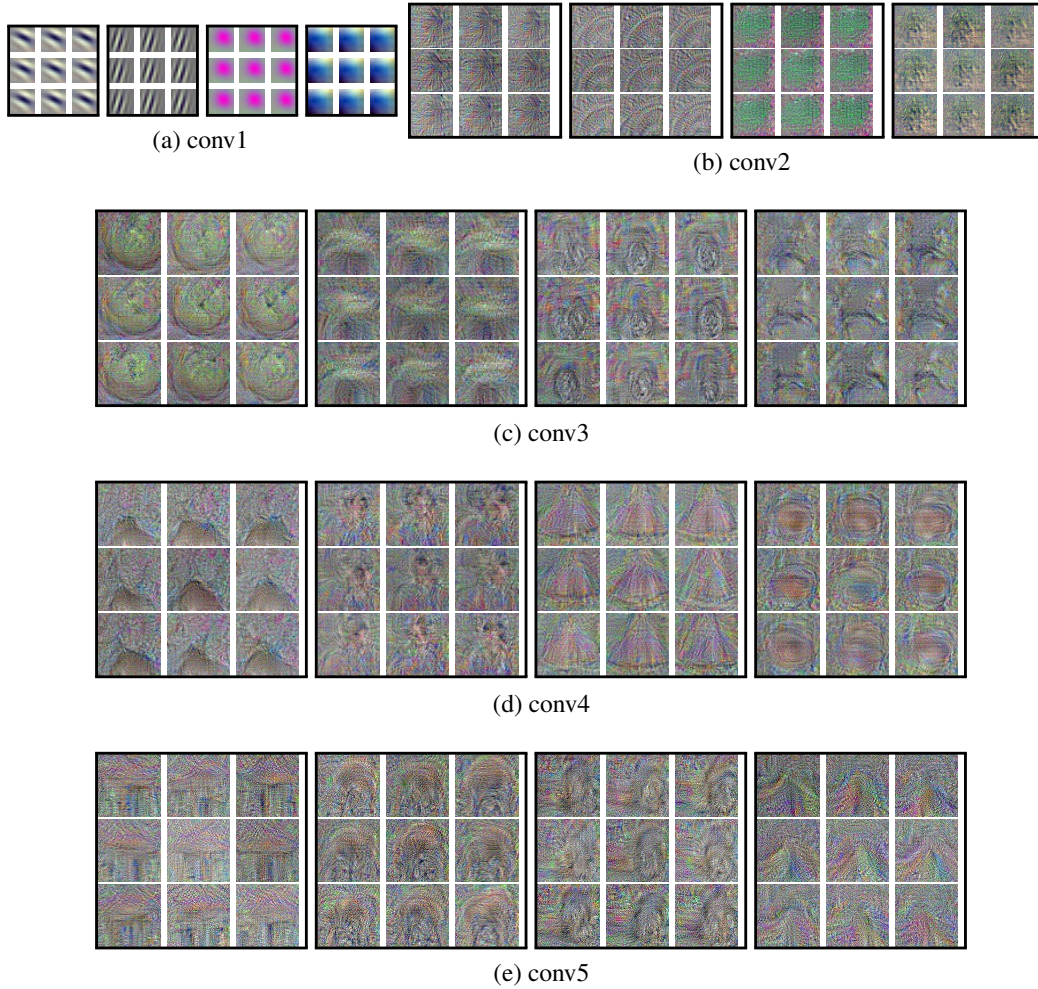


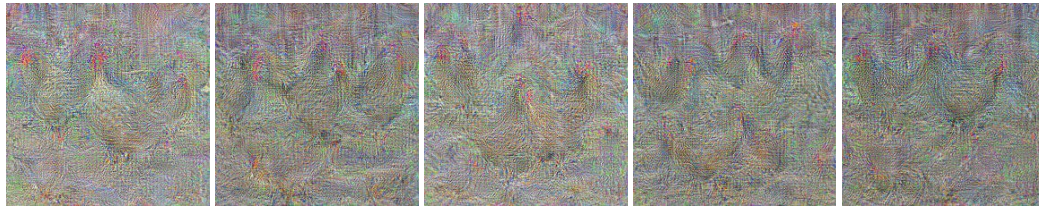
Figure 3: Samples from the nodes at the intermediate convolutional layers (conv1 to conv5) in the fully trained AlexNet model.

approximated by simple average over synthetic images generated from the current distribution, instead of the weighted average over examples from the non-parametric reference distribution. We shall explore this in further work.

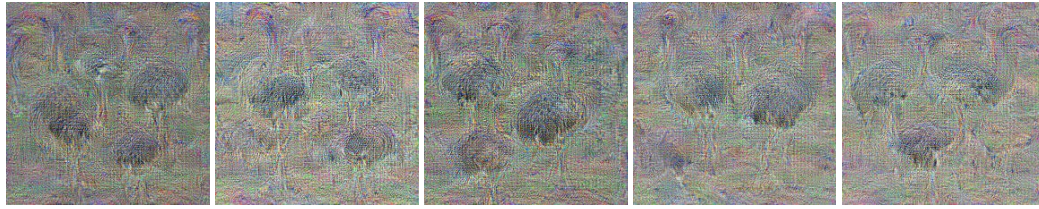
## REFERENCES

- Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
- Efron, Bradley. The efficiency of logistic regression compared to normal discriminant analysis. *Journal of the American Statistical Association*, 1975.
- Erhan, Dumitru, Bengio, Yoshua, Courville, Aaron, and Vincent, Pascal. Visualizing higher-layer features of a deep network. *Dept. IRO, Université de Montréal, Tech. Rep*, 2009.
- Erhan, Dumitru, Bengio, Yoshua, Courville, Aaron, Manzagol, Pierre-Antoine, Vincent, Pascal, and Bengio, Samy. Why does unsupervised pre-training help deep learning? *JMLR*, 11, 2010.
- Girshick, Ross, Donahue, Jeff, Darrell, Trevor, and Malik, Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2013.

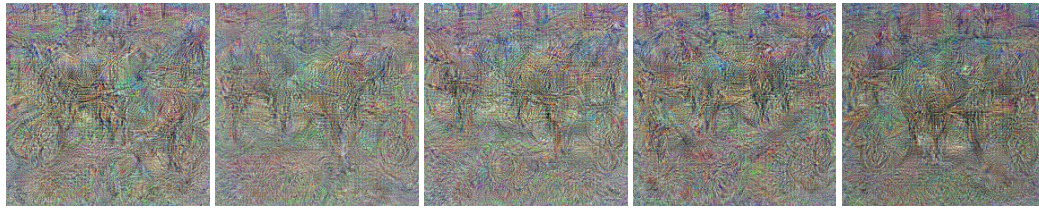




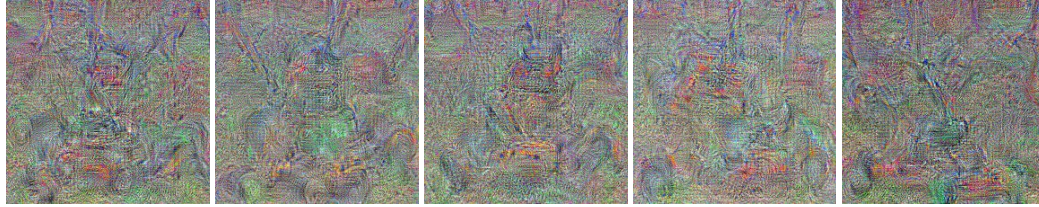
(a) Hen



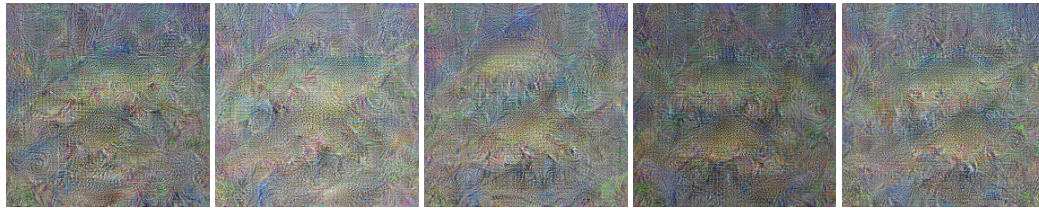
(b) Ostrich, *Struthio camelus*



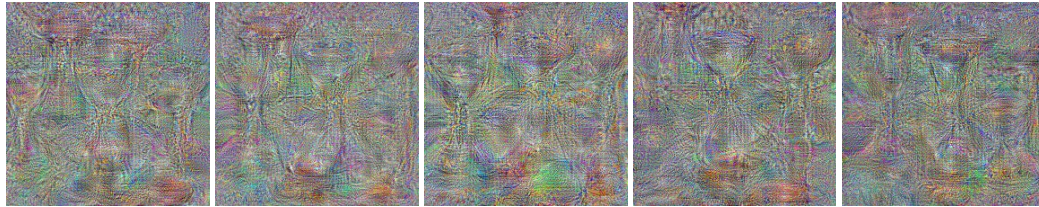
(c) Horse cart, horse-cart



(d) Lawn mower, mower



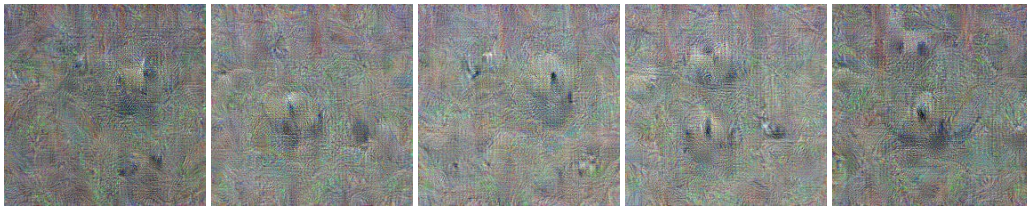
(e) Tench, *Tinca tinca*



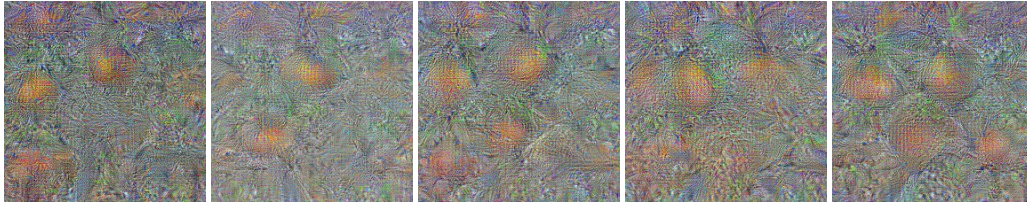
(f) Hourglass

Figure 4: Samples from the nodes at the final fully-connected layer (fc8) in the fully trained AlexNet model, which correspond to different object categories (part 1).

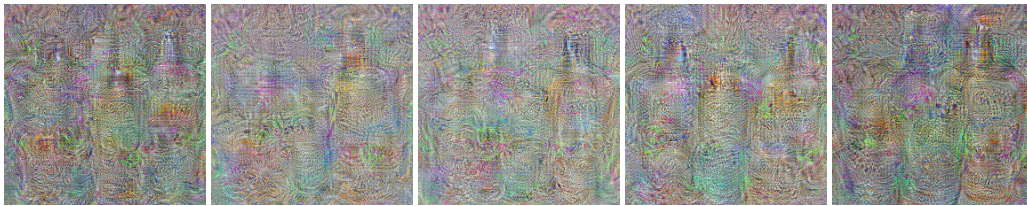




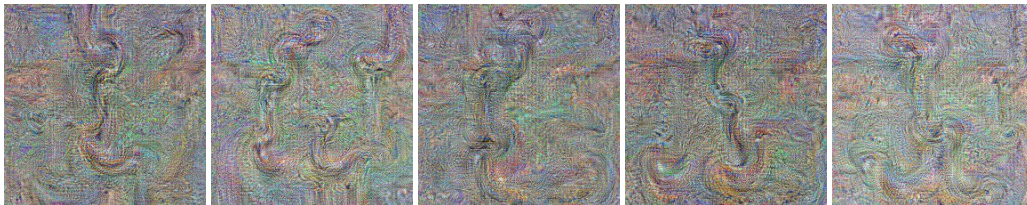
(a) Giant panda, panda, panda bear, coon bear, *Ailuropoda melanoleuca*



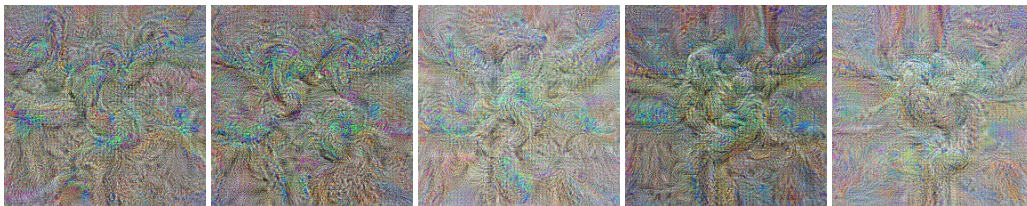
(b) Orange



(c) Lotion



(d) Hook, claw



(e) Knot



(f) Nail

Figure 5: Samples from the nodes at the final fully-connected layer (fc8) in the fully trained AlexNet model, which correspond to different object categories (part 2).

- Hinton, Geoffrey, Osindero, Simon, and Teh, Yee-Whye. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006a.
- Hinton, Geoffrey, Osindero, Simon, Welling, Max, and Teh, Yee-Whye. Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive Science*, 2006b.
- Hinton, Geoffrey E. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8), 2002.
- Jia, Yangqing, Shelhamer, Evan, Donahue, Jeff, Karayev, Sergey, Long, Jonathan, Girshick, Ross, Guadarrama, Sergio, and Darrell, Trevor. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.
- Jordan, A. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *NIPS*, 14:841, 2002.
- Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.
- Le, Quoc V, Ranzato, M A, Monga, R, Devin, M, Chen, K, Corrado, G S, Dean, J, and Ng, A Y. Building high-level features using large scale unsupervised learning. In *ICML*, 2012.
- LeCun, Yann, Boser, Bernhard, Denker, John S, Henderson, Donnie, Howard, Richard E, Hubbard, Wayne, and Jackel, Lawrence D. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.
- LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.
- Liang, Percy and Jordan, Michael I. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *ICML*, 2008.
- Long, Jonathan L, Zhang, Ning, and Darrell, Trevor. Do convnets learn correspondence? In *NIPS*, 2014.
- Neal, Radford. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2011.
- Rifai, Salah, Vincent, Pascal, Muller, Xavier, Glorot, Xavier, and Bengio, Yoshua. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*, pp. 833–840, 2011.
- Roth, Stefan and Black, Michael J. Fields of experts. *IJCV*, 2009.
- Salakhutdinov, Ruslan and Hinton, Geoffrey E. Deep boltzmann machines. In *AISTATS*, 2009.
- Zeiler, Matthew D and Fergus, Rob. Visualizing and understanding convolutional neural networks. *arXiv preprint arXiv:1311.2901*, 2013.