Statistics and Its Interface Volume 9 (2016) 485–496

Generative modeling of convolutional neural networks

Jifeng Dai, Yang Lu, and Ying Nian Wu*

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

Keywords and phrases: Big data, Deep learning.

1. INTRODUCTION

1.1 Big data and big model

Recent years have witnessed the triumphant return of the feedforward neural networks, especially the convolutional neural networks (ConvNets) [16]. Fueled by the availabilities of large labeled data sets such as ImageNet [1] and the increased computing power, ConvNets have proven to be a powerful tool for discriminative or predictive learning [14, 6].

The ImageNet dataset and the associated ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [23] were the products of the big data era. The ImageNet dataset, first released in 2009, is a collection of more than 15 million images organized into roughly 22,000 categories. The categories are from the visually meaningful concepts in the WordNet, a database of English words. The images were collected by querying the categories on the internet search engines such as Google, and were manually examined by crowed-sourcing workers from Amazon's Mechanical Turk. Starting from 2010, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has been held annually. It is a challenging large-scale contest in computer vision. In its image classification task, which is to assign each image to a category, there are roughly 1.2 million training images, 50,000 validation images, and 100,000 testing images from a 1,000 category subset of the ImageNet dataset.

In ILSVRC 2012, [14] won the image classification task using a big and deep ConvNet. This big model, now commonly called AlexNet, has 60 million parameters and 650,000 hidden nodes. It consists of 5 convolutional layers (some of them are followed by max-pooling layers), and 3 fully-connected layers. AlexNet significantly reduced the classification error on the classification task and surpassed the runner-up by big margin, a feat that stunned the computer vision community. Subsequent research further revealed that the deep ConvNet pre-trained on ImageNet can be adapted to other vision tasks with limited training data, e.g., object detection [6] and semantic segmentation [7]. This indicates that deep convolutional network can learn powerful generic features from the big ImageNet dataset.

1.2 Generative modeling

Despite the successes of the discriminative learning of ConvNets, the generative aspect of ConvNets 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) The generative semi-supervised learning algorithms can be employed to absorb knowledge from the unlabeled data; (3) Samples can be drawn from the generative model to reveal the knowledge learned by the ConvNet. Although many generative models and learning algorithms have been proposed [8, 9, 21, 24], most of them do not scale well and have not been applied to learning large and deep ConvNets.

In this paper, we study the generative modeling of the ConvNets. We start from defining probability distributions of images given the underlying object categories or class labels, such that the ConvNet 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,

*Corresponding author.

i.e., exponential family 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 theme, we propose a non-parametric generative gradient for pre-training the ConvNet, where the ConvNet is learned by the stochastic gradient algorithm that seeks to maximize 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 [17] and the challenging ImageNet [1] classification benchmarks show that this generative pre-training scheme consistently helps improve the performance of ConvNets in both supervised and semi-supervised settings.

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 algorithm [20], 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 ConvNet, without resorting to any extra hold-out images. Experiments on ImageNet show that meaningful and varied synthetic images can be generated for nodes of a big and deep ConvNet discriminatively trained on ImageNet.

2. PAST WORK

The generative model that we study is an energy-based model. Such models include field of experts [22], product of experts [10], Boltzmann machines [8], model based on neural networks [9], etc. However, most of these generative models and learning algorithms have not been applied to learning deep ConvNets. The proposed non-parametric generative gradient as well as the generative model based on deep ConvNets 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 [3], and more recently by [12, 18], etc. Moreover, the usefulness of generative pre-training for deep learning has been studied by [4] etc. However, this issue has not been thoroughly investigated for ConvNets. Moreover, unlike previous pre-training methods, our non-parametric generative gradient shares the same computational architecture as the discriminative gradient.

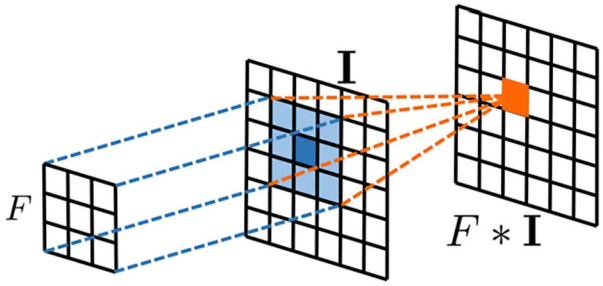


Figure 1. Filtering or convolution: applying a filter F (3×3) on an image $\mathbf{I}$ (6×6) to get a filtered image (6×6) or feature map $F * \mathbf{I}$. Each pixel of $F * \mathbf{I}$ is computed by the weighted sum of the 3×3 pixels of $\mathbf{I}$ centered at this pixel.

As to visualization, our work is related to [5, 15, 6, 26, 19]. In [6, 19], the high-scoring image patches are directly presented. In [26], 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 [5, 15], 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 [6, 26, 19].

3. BACKGROUND

3.1 Filters

To fix notation, let $\mathbf{I}(x)$ be an image defined on the square (or rectangular) domain $\mathcal{D}$, where $x = (x_1, x_2)$ (a two-dimensional vector) indexes the coordinates of pixels. We can treat $\mathbf{I}(x)$ as a two-dimensional function defined on $\mathcal{D}$. We can also treat $\mathbf{I}(x)$ as a vector if we fix an ordering for the pixels.

A linear filter is just a local weighted sum of image intensities around each pixel. Suppose we have a set of linear filters $\{F_k, k = 1, ..., K\}$. We can apply each F_k to image $\mathbf{I}$ to obtain a filtered image or feature map, denoted by $F_k * \mathbf{I}$, which is of the same size as $\mathbf{I}$ and is also defined on $\mathcal{D}$ (with proper handling of boundaries). Let $[F_k * \mathbf{I}](y)$ be the filter response or feature at position y. Then

$$[F_k * \mathbf{I}](y) = \sum_{x \in \mathcal{S}} w_{k,x} \mathbf{I}(y + x), \tag{1}$$

where the weights or coefficients $w_{k,x}$ define the filter F_k, and $\mathcal{S}$ is the localized support of the filter centered at the origin. See Fig. 1 for an illustration, where $\mathcal{S}$ is 3×3, and

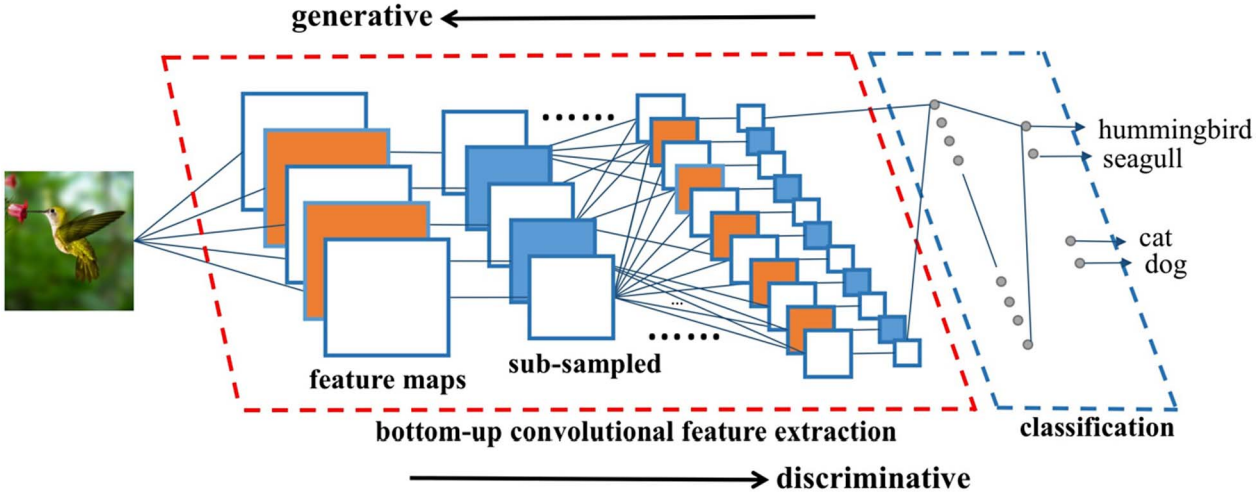


Figure 2. Convolutional neural networks consist of multiple layers of filtering and sub-sampling operations for bottom-up feature extraction, resulting in multiple layers of feature maps and their sub-sampled versions. The top layer features are used for classification via multinomial logistic regression. The discriminative direction is from image to category, whereas the generative direction is from category to image. This illustration is adapted from [17].

$\mathcal{D}$ is 6×6. In practice, both $\mathcal{S}$ and $\mathcal{D}$ can be much larger. $\mathcal{S}$ can be different for different F_k. The filtering operation is also a convolution operation.

3.2 ConvNet: filters on top of filters

For statisticians, a neural network can be viewed as a generalization of the generalized linear model (GLM) such as logistic regression. Specifically, it can be viewed as a multi-layer recursive composition of GLMs. The convolutional neural network, or ConvNet [17], is a specialized neural network devised for analyzing signals such as images, where the linear combinations take place around each pixel, i.e., they are filters or convolutions. See Fig. 2 for an illustration.

A ConvNet consists of multiple layers of linear filtering and non-linear transformation, as expressed by the following recursive formula of filters on top of filters

(2)
$$[F_j^{(l)} * \mathbf{I}](y) = h\left(\sum_{k=1}^{N_{l-1}} \sum_{x \in \mathcal{S}_l} w_{k,x}^{(l,j)} [F_k^{(l-1)} * \mathbf{I}](y+x) + b_{l,j}\right),$$

where l indexes the layer. $\{F_j^{(l)}, j = 1, ..., N_l\}$ are the filters at layer l, $\{F_k^{(l-1)}, k = 1, ..., N_{l-1}\}$ are the filters at layer $l-1$. j and k are used to index filters at layers l and $l-1$ respectively, and N_l and N_{l-1} are the numbers of filters at layers l and $l-1$ respectively. The filters are locally supported, so the range of x in $\sum_x$ is within a local support $\mathcal{S}_l$ (such as a 7×7 image patch). We define the image $\mathbf{I}$ to be the feature map at the 0-th layer. The filter responses at layer l are computed from the filter responses at layer $l-1$, by linear filtering defined by the weights $w_{k,x}^{(l,j)}$ as well as the bias term $b_{l,j}$ ($w_{k,x}^{(l,j)}$ and $b_{l,j}$ correspond to the coefficients and the intercept of a GLM), followed by the non-linear transformation $h()$ (corresponding to the link function of a GLM). $\{F_j^{(l)}\}$ are non-linear filters because we incorporate $h()$ in the computation of the filter responses. We call $F_j^{(l)} * \mathbf{I}$ the filtered image or the feature map of filter j at layer l. There are a total N_l feature maps in layer l, and $j = 1, ..., N_l$. In Fig. 2, the feature maps are illustrated by the square shapes. Each $[F_j^{(l)} * \mathbf{I}](x)$ is called a feature extracted by a node or a unit at layer l.

The filtering operations are often followed by sub-sampling and local-max pooling (e.g., $\mathbf{I}(x_1, x_2) \leftarrow \max_{(\delta_1,\delta_2)\in\{0,1\}^2} \mathbf{I}(2x_1 + \delta_1, 2x_2 + \delta_2)$). See Fig. 2 for an illustration of sub-sampling. After a number of layers with sub-sampling, the filtered images or feature maps are reduced to 1×1 at the top layer. These features are then used for classification (e.g., does the image contain a hummingbird or a seagull or a dog). Specifically, let the top layer filter responses or features be $\{F_k^{(L)} * \mathbf{I}, k = 1, ..., N_L\}$. These features are used for classification via multinomial logistic regression. Specifically, let $c \in \{1, 2, ..., C\}$ be the category of image $\mathbf{I}$, then the score is

(3)
$$f_c(\mathbf{I}; w) = \sum_k w_{c,k} [F_k^{(L)} * \mathbf{I}] + b_{c,k},$$

where $w_{c,k}$ and $b_{c,k}$ are the weights (coefficients) and bias (intercept) for computing the score of category c, and the parameter w includes the category-specific $w_{c,k}$ and $b_{c,k}$, as well as the weight and bias parameters at all the layers below. The conditional probability of the category c given the image $\mathbf{I}$ is

$$p(c|\mathbf{I}, w) = \frac{\exp(f_c(\mathbf{I}; w))}{\sum_c \exp(f_c(\mathbf{I}; w))}. \tag{4}$$

For identifiability, we may choose a base category, e.g., background, with $c = 0$, and define $f_0(\mathbf{I}) = 0$.

The estimation of the weight and bias parameters can be accomplished by gradient ascent on the log-likelihood. For big data, we can divide the data into mini-batches, so that at each step, we run gradient ascent based on the log-likelihood of a randomly sampled mini-batch. The gradient can be calculated by back-propagation, which is an application of the chain rule.

4. GENERATIVE MODELING OF CONVNET

4.1 Probability distributions on images

For the rest of the paper, we shall switch to the conventional notation in statistical literature where we use x to denote the input (which was denoted by $\mathbf{I}$ for input image in the above subsection), and we use y to denote the output (which was denoted by c for output category in the above subsection).

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,

$$p_y(x; w) = \frac{1}{Z_y(w)} \exp\left(f_y(x; w)\right) q(x), \tag{5}$$

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 $Z_y(w) = \mathrm{E}_q[\exp(f_y(x; w))] = \int \exp(f_y(x; w))q(x)dx$ is the normalizing constant. 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 (5), 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

$$p(y|x, w) = \frac{\exp(f_y(x; w) + \alpha_y)}{\sum_y \exp(f_y(x; w) + \alpha_y)}, \tag{6}$$

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 α_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 α_y is already absorbed into w for the rest of the paper. Note that $f_y(x; w)$ is not unique in (6). 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 (5) mentioned above.

Given a set of labeled data $\{(x_i, y_i)\}$, equations (5) and (6) 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 ρ_y can be estimated by class frequency of category y. The other is to maximize the discriminative log-likelihood $l_D(w) = \sum_i \log p(y_i|x_i, w)$.

4.2 Generative gradient

The gradient of the discriminative log-likelihood defined by (6) is calculated according to

$$\begin{aligned} &\frac{\partial}{\partial w} \log p(y_i|x_i, w) \\ =&\frac{\partial}{\partial w} f_{y_i}(x_i; w) - \mathrm{E}_D\left[\frac{\partial}{\partial w} f_y(x_i; w)\right], \end{aligned} \tag{7}$$

where α_y is absorbed into w as mentioned above, and

$$\begin{aligned} &\mathrm{E}_D\left[\frac{\partial}{\partial w} f_y(x_i; w)\right] \\ =&\sum_y \frac{\partial}{\partial w} f_y(x_i; w) \frac{\exp(f_y(x_i; w))}{\sum_y \exp(f_y(x_i; w))}. \end{aligned} \tag{8}$$

The gradient of the generative log-likelihood defined by (5) is calculated according to

$$\begin{aligned} &\frac{\partial}{\partial w} \log p_{y_i}(x_i; w) \\ =&\frac{\partial}{\partial w} f_{y_i}(x_i; w) - \mathrm{E}_G\left[\frac{\partial}{\partial w} f_{y_i}(x; w)\right], \end{aligned} \tag{9}$$

where

$$\begin{aligned} &\mathrm{E}_G\left[\frac{\partial}{\partial w} f_{y_i}(x; w)\right] \\ =&\int \frac{\partial}{\partial w} f_{y_i}(x; w) \frac{1}{Z_{y_i}(w)} \exp(f_{y_i}(x; w)) q(x), \end{aligned} \tag{10}$$

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,

$$\mathrm{E}_G\left[\frac{\partial}{\partial w} f_{y_i}(x; w)\right] \approx \sum_j \frac{\partial}{\partial w} f_{y_i}(\tilde{x}_j; w) W_j, \tag{11}$$

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

$$\frac{\partial}{\partial w} \log p_{y_i}(x_i; w) \approx \frac{\partial}{\partial w} f_{y_i}(x_i; w) - \sum_j \frac{\partial}{\partial w} f_{y_i}(\tilde{x}_j; w) \frac{\exp(f_{y_i}(\tilde{x}_j; w))}{\sum_k \exp(f_{y_i}(\tilde{x}_k; w))}. \tag{12}$$

The discriminative gradient and the generative gradient differ subtly and yet fundamentally in calculating $\mathrm{E}[\partial f_y(x; w)/\partial w]$, 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.

4.3 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 $\{\tilde{x}_j\}_{j=1}^m$ in (12) by the batch sample $\{x_j\}_{j=1}^n$, we can rewrite (12) in the following form,

$$\frac{\partial}{\partial w} \log p_{y_i}(x_i; w) \approx \sum_{y,j} \frac{\partial \log p_{y_i}(x_i; w)}{\partial f_y(x_j; w)} \frac{\partial f_y(x_j; w)}{\partial w}, \tag{13}$$

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 (12), $\partial \log p_{y_i}(x_i; w)/\partial f_y(x_j; w)$ is defined by

$$\frac{\partial \log p_{y_i}(x_i; w)}{\partial f_y(x_j; w)} = \begin{cases} 0 & y \neq y_i; \\ 1 - \frac{\exp(f_{y_i}(x_j; w))}{\sum_k \exp(f_{y_i}(x_k; w))} & y = y_i, j = i; \\ -\frac{\exp(f_{y_i}(x_j; w))}{\sum_k \exp(f_{y_i}(x_k; w))} & y = y_i, j \neq i. \end{cases} \tag{14}$$

4.4 Generative vs discriminative for batch training

We can derive the generative gradient for batch training more directly as follows. During training, on a batch of training examples, $\{(x_i, y_i), i = 1, ..., n\}$, the generative log-likelihood is,

$$l_G(w) = \sum_i \log p(x_i | y_i, w) = \sum_i \log \frac{\exp(f_y(x; w))}{Z_y(w)} \approx \sum_i \log \frac{\exp(f_y(x; w))}{\sum_i \exp(f_y(x_i; w))/n}. \tag{15}$$

The gradient with respect to w is

$$l'_G(w) = \sum_i \left[\frac{\partial}{\partial w} f_{y_i}(x_i; w) - \sum_j \frac{\partial}{\partial w} f_{y_i}(x_j; w) \frac{\exp(f_{y_i}(x_j; w))}{\sum_k \exp(f_{y_i}(x_k; w))} \right]. \tag{16}$$

As a comparison, the discriminative log-likelihood is

$$l_D(w) = \sum_i \log p(y_i | x_i, w) = \sum_i \log \frac{\exp(f_{y_i}(x_i; w))}{\sum_y \exp(f_y(x_i; w))}. \tag{17}$$

The gradient with respect to w is

$$l'_D(w) = \sum_i \left[\frac{\partial}{\partial w} f_{y_i}(x_i; w) - \sum_y \frac{\partial}{\partial w} f_y(x_i; w) \frac{\exp(f_y(x_i; w))}{\sum_y \exp(f_y(x_i; w))} \right]. \tag{18}$$

l'_D and l'_G are similar in form but they are 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 (as well as the intercepts for different y) to satisfy the discriminative constraint.

4.5 Semi-supervised learning

In semi-supervised learning, given a set of labeled training examples $\{(x_i, y_i)\}_{i=1}^{l}$ and a set of unlabeled training examples $\{x_i\}_{i=l+1}^{l+u}$, we seek to maximize the semi-supervised log-likelihood

$$l_S(w) = \sum_{i=1}^{l} \log p_{y_i}(x_i; w) + \lambda \sum_{i=l+1}^{l+u} \log p(x_i; w), \tag{19}$$

where λ is a tuning parameter, and $p(x_i; w) = \sum_y p_y(x_i; w)\rho_y$.

As a common practice, we approximate $p(x_i; w)$ by $\max_y p_y(x_i; w)\rho_y$. Since $p(x_i; w) \geq \max_y p_y(x_i; w)\rho(y)$, we maximize a lower bound of equation (19). The gradient can be calculated according to Section 4.3.

4.6 Generative visualization

Recently, researchers have become interested in understanding what the ConvNet has learned. 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) [20].

Specifically, we can write $p_y(x; w)$ as $p_y(x; w) \propto \exp(-U(x))$, where $U(x) = (-f_y(x; w) + \frac{1}{2\sigma^2}|x|^2)/T$, with T being the temperature. 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 ϕ 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, ϕ) 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 ϕ follows a Gaussian distribution and is independent of x. Each time HMC draws a random sample from the marginal Gaussian distribution of ϕ, and then evolve according to the Hamiltonian dynamics that conserves the total energy.

In practical implementation, the leapfrog algorithm [20] is used to discretize the continuous Hamiltonian dynamics as follows, with ϵ being the step-size:

$$\phi^{(t+\epsilon/2)} = \phi^{(t)} - (\epsilon/2)\frac{\partial U}{\partial x}(x^{(t)}), \tag{20}$$

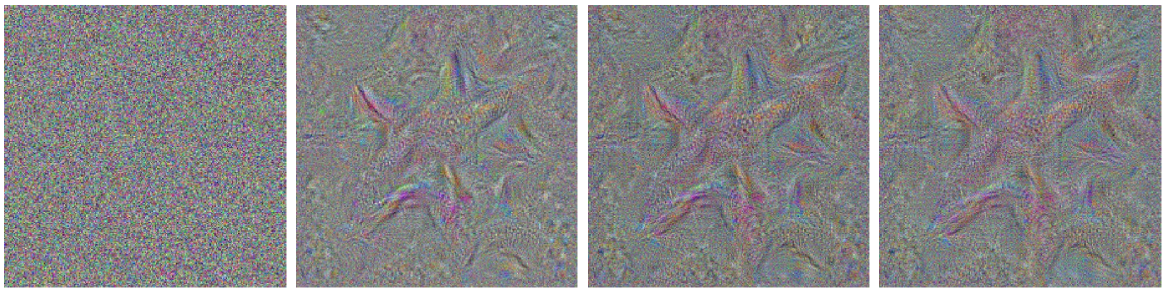
$$x^{(t+\epsilon)} = x^{(t)} + \epsilon \frac{\phi^{(t+\epsilon/2)}}{m}, \tag{21}$$

$$\phi^{(t+\epsilon)} = \phi^{(t+\epsilon/2)} - (\epsilon/2)\frac{\partial U}{\partial x}(x^{(t+\epsilon)}), \tag{22}$$

that is, a half-step update of ϕ 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 step is added 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 [26]. The visualization sequence of a sample is shown in Fig. 3.

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.



Iteration 0 Iteration 10 Iteration 50 Iteration 100

Figure 3. The sequence of images produced by HMC that samples from the "Starfish, sea star" category of the "AlexNet" network [14] discriminatively trained on ImageNet ILSVRC-2012.

5. EXPERIMENTS

5.1 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 [11] and the experiment settings are identical to [11]. Experiments are performed on two widely used image classification benchmarks: MNIST [17] handwritten digit recognition and ImageNet ILSVRC-2012 [1] natural image classification.

MNIST handwritten digit recognition. We first study generative pre-training on the MNIST dataset. The "LeNet" network [17] 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.

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" [14] and "ZeilerFergusNet" (fast) [26]. 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.

Table 1. Error rates on the MNIST test *set of different training approaches utilizing the "LeNet" network [17]*

Training approaches	DG	Ours (GG)	Ours (GG+DG)
Error rates	1.03	0.85	**0.78**

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

Training approaches	DG	Ours (GG)	Ours (GG+DG)
AlexNet	40.7	45.8	**39.6**
ZeilerFergusNet	38.4	44.3	**37.4**

As shown in Table 2, the error rates of discriminative gradient training applied on AlexNet and ZeilerFergusNet are 40.7% and 38.4% respectively. 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

Figure 4. Samples from the nodes at the final fully-connected layer in the fully trained LeNet model, which correspond to different handwritten digits.

Table 3. Top-1 classification error rates on the ImageNet ILSVRC-2012 val *set of different semi-supervised learning approaches (utilizing the "AlexNet")*

Labeled number	50,000	100,000	300,000
DG	92.2	78.7	61.3
TDG+DG [13]	91.0	77.2	60.1
LPAL [2]	94.3	-	-
Ours (*GG+DG*)	**89.2**	**75.0**	**57.7**

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.

5.2 Semi-supervised learning

In semi-supervised learning experiments, we apply the proposed semi-supervised learning algorithm on the large-scale ImageNet ILSVRC-2012 natural image classification benchmark, and compare with several semi-supervised learning approaches [13, 2]. Although there are other semi-supervised learning algorithms [25] available for ConvNets, most of them do not scale well and are not applicable on such a large-scale dataset.

In the experiments, a random subset of examples in the *train* set are used as labeled examples, while all the other examples in the *train* set are treated as unlabeled examples. We apply the proposed semi-supervised learning algorithm on the *train* dataset that consists of both labeled and unlabeled examples, and then finetune the network by discriminative gradient on the labeled subset (*GG+DG*). The tuning parameter λ is set to 0.1 in our experiments. As a comparison, we also report the error rates of discriminative gradient on the labeled subset (*DG*), a transductive discriminative ConvNet [13] trained on the full dataset and finetuned by discriminative gradient on the labeled subset (*TDG+DG*). As shown in Table 3, when the numbers of labeled examples are 50,000, 100,000 and 300,000 respectively, the error rates of the proposed approach on the *val* set are 89.2%, 75.0%, and 57.7% respectively. The error rates are 1.8%, 2.2% and 2.4% lower than those of [13], and 3.0%, 3.7% and 3.6% lower than those of discriminative gradient (all the approaches utilize the "AlexNet"). [2] performs semi-supervised learning by linear propagation with active learning (*LPAL*), and reports an error rate of 94.3% on 50,000 labeled examples. Our approach's error rate is 5.1% lower than that.

The computational cost of the proposed semi-supervised learning algorithm is the same as that in fully-supervised training, because there is no additional computational overhead.

5.3 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 the experiments, we delete the dropout layer to avoid unnecessary noise for visualization. At the beginning of visualization, x is initialized from the reference distribution $q(x)$. The HMC iteration number, the leapfrog step size, the leapfrog step number, the standard deviation of reference distribution σ, and the particle mass are set to be 300, 10^{-4}, 100, 10, and 0.01 respectively. The visualization results are shown in Fig. 4. 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

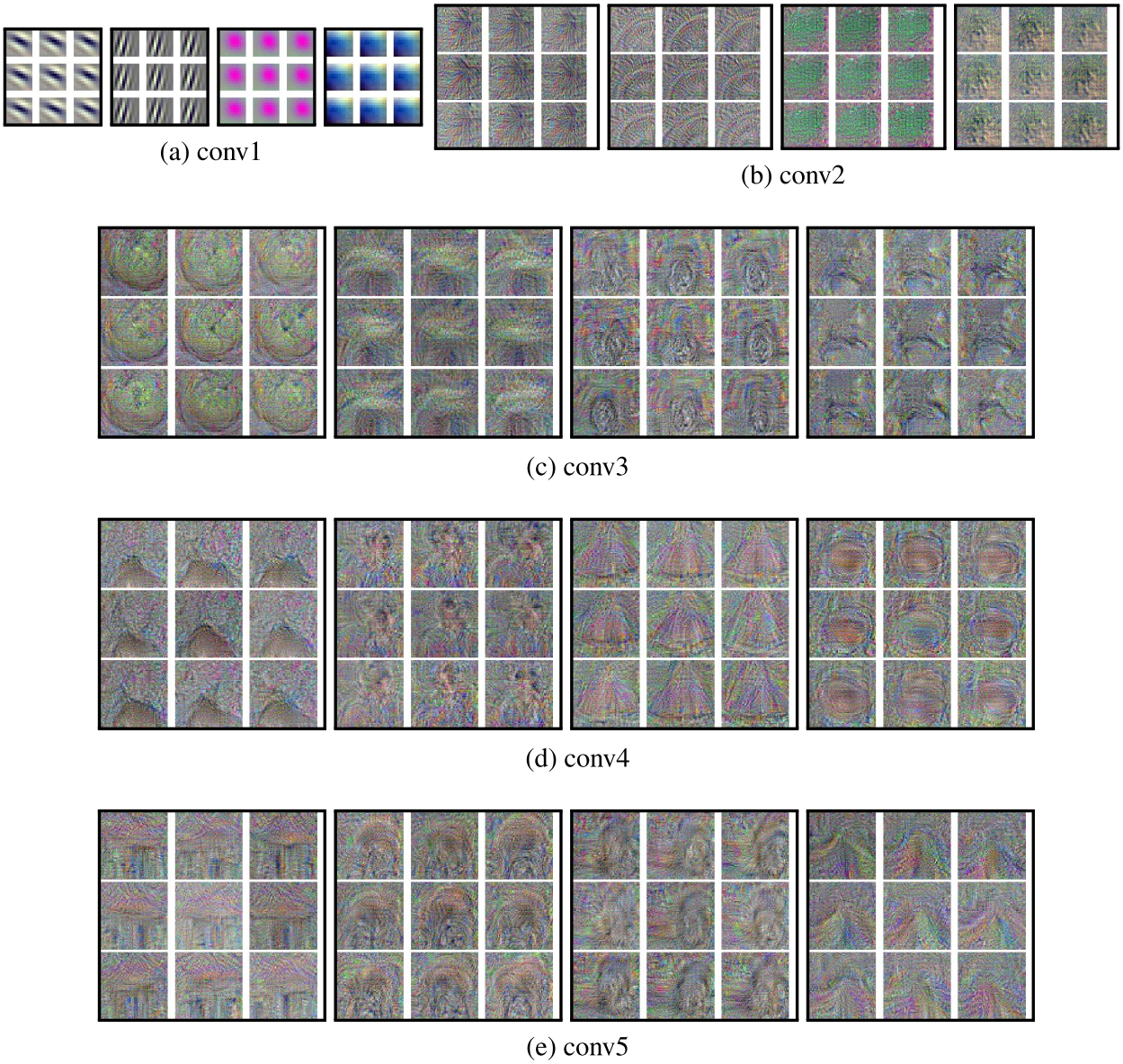


Figure 5. Samples from the nodes at the intermediate convolutional layers (conv1 to conv5) in the fully trained AlexNet model.

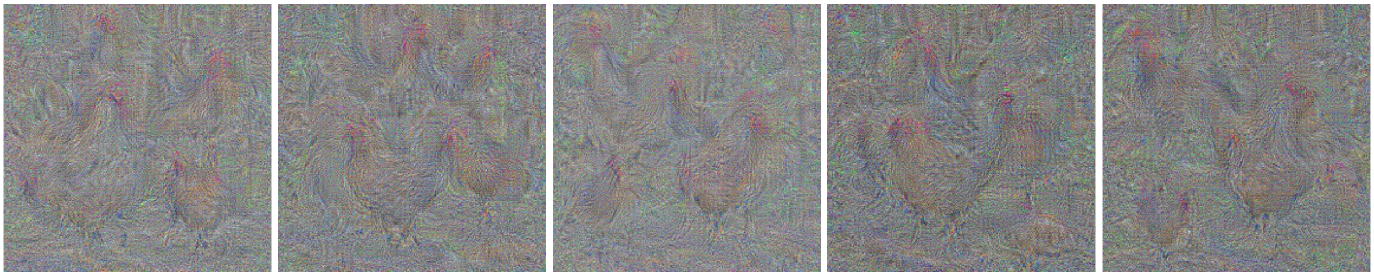
leapfrog step size, the leapfrog step number, the standard deviation of reference distribution σ, and the particle mass are set to be 3×10^{-6}, 50, 10, and 10^{-5} respectively. The HMC iteration numbers are 100 and 500 for nodes from the intermediate convolutional and the final fully-connected layers respectively. We set the temperature $T = .001$ in visualizing AlexNet.

The samples from the intermediate convolutional layers of AlexNet are shown in Fig. 5, while those from the final fully-connected layer are shown in Fig. 6 to Fig. 7. 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 [6, 26, 19].

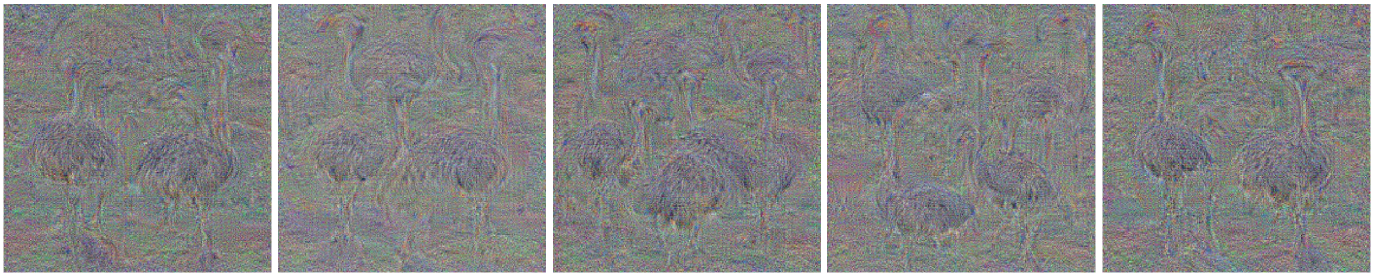
As to the computational cost, it varies for nodes at different layers within different networks. On a desktop with GTX Titian, it takes about 0.4 minute to draw a sample for nodes at the final fully-connected layer of LeNet. In AlexNet, for nodes at the first convolutional layer and at the final fully-connected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

6. CONCLUSION

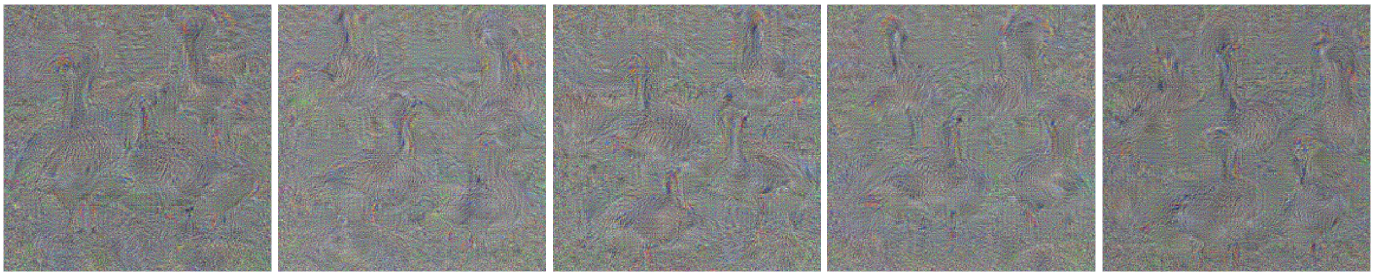
Given the recent successes of ConvNets, it is worthwhile to explore their generative aspects. In this work, we show



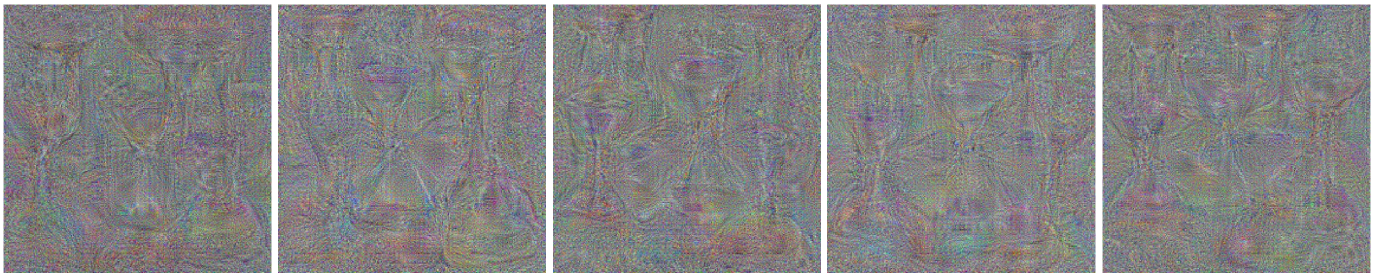
(a) Hen



(b) Ostrich



(c) Goose



(d) Hourglass

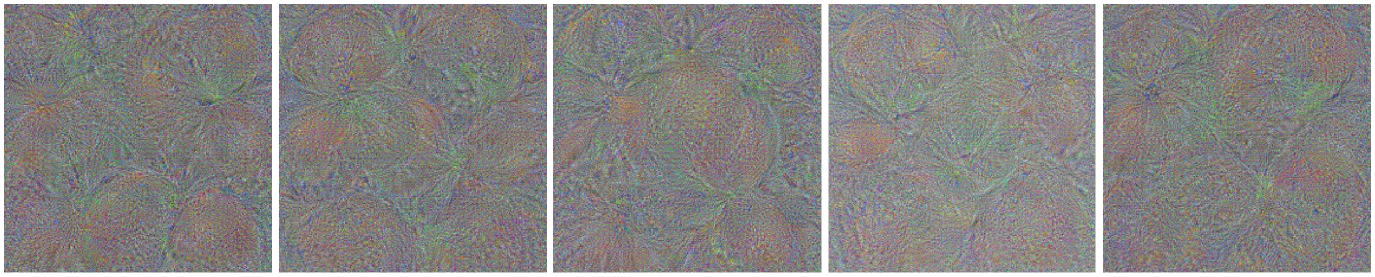
Figure 6. Samples from the nodes at the final fully-connected layer (fc8) of the AlexNet model, which correspond to different object categories.

that a simple generative model can be constructed based on the ConvNet. The generative model helps to pre-train the ConvNet. It also helps to visualize the knowledge of the learned ConvNet. Although we work on the image data in this paper, the proposed method can be applied to other types of data where ConvNets apply.

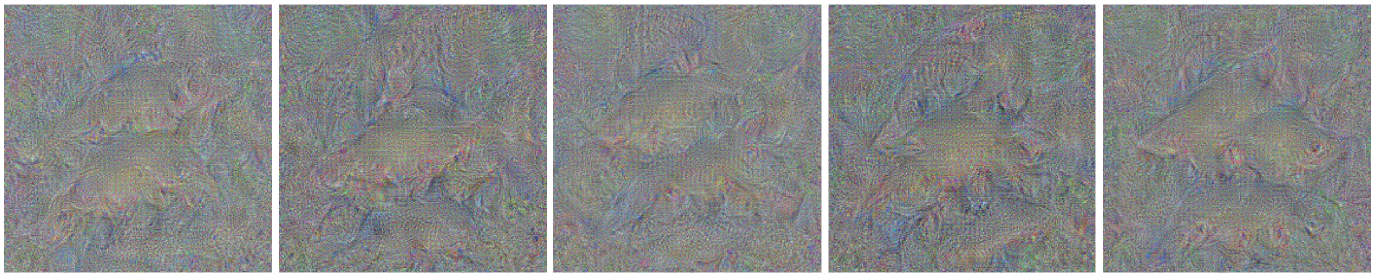
The proposed visualization scheme can sample from the generative model, and it may be turned into a parametric generative learning algorithm, where the generative gradient can be approximated by samples generated by the current model.

ACKNOWLEDGEMENT

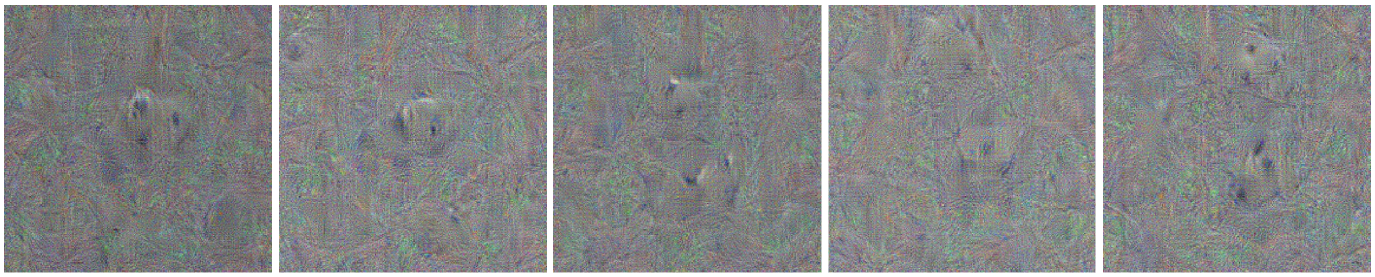
We thank Adrian Barbu for helpful discussions. The work is supported by NSF DMS 1310391, ONR MURI N00014-10-1-0933, DARPA SIMPLEX N66001-15-C-4035, and DARPA MSEE FA 8650-11-1-7149.



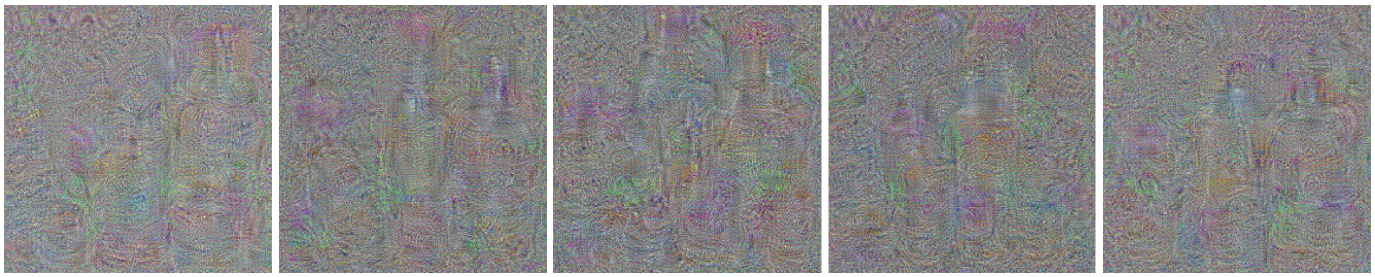
(a) Orange



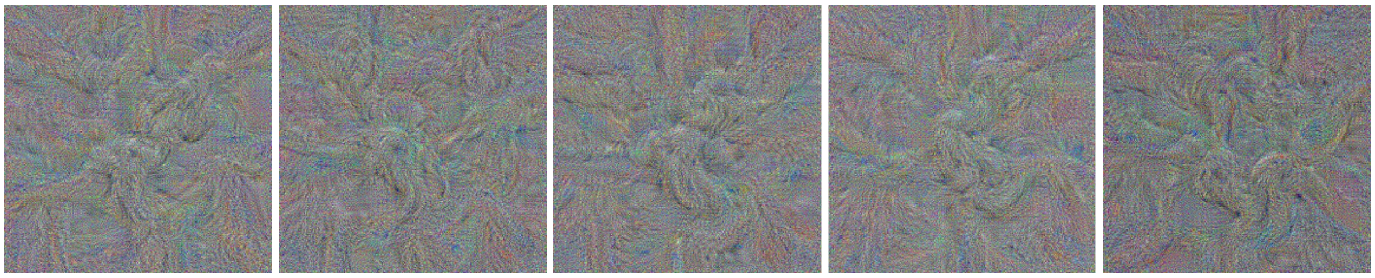
(b) Fish



(c) Panda



(d) Bottle



(e) Knot

Figure 7. Samples from the nodes at the final fully-connected layer (fc8) of the AlexNet model, which correspond to different object categories.

Received 2 April 2015

REFERENCES

[1] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *CVPR*.

[2] Ebert, S., Fritz, M. and Schiele, B. (2012). Semi-supervised learning on a budget: Scaling up to large datasets. In *ACCV*.

[3] Efron, B. (1975). The efficiency of logistic regression compared to normal discriminant analysis. *Journal of the American Statistical Association*. MR0391403

[4] Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P. and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *JMLR* **11**. MR2600623

[5] Erhan, D., Bengio, Y., Courville, A. and Vincent, P. (2009). Visualizing higher-layer features of a deep network. *Dept. IRO, Université de Montréal, Tech. Rep.*

[6] Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv preprint arXiv:1311.2524.

[7] Hariharan, B., Arbeláez, P., Girshick, R. and Malik, J. (2014). Simultaneous detection and segmentation. In *ECCV*.

[8] Hinton, G., Osindero, S. and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation* **18** 1527–1554. MR2224485

[9] Hinton, G., Osindero, S., Welling, M. and Teh, Y.-W. (2006). Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive Science*.

[10] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation* **14**.

[11] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093.

[12] Jordan, A. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *NIPS* **14** 841.

[13] Karlen, M., Weston, J., Erkan, A. and Collobert, R. (2008). Large scale manifold transduction. In *ICML*.

[14] Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.

[15] Le, Q. V., Ranzato, M. A., Monga, R., Devin, M., Chen, K., Corrado, G. S., Dean, J. and Ng, A. Y. (2012). Building high-level features using large scale unsupervised learning. In *ICML*.

[16] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*.

[17] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.

[18] Liang, P. and Jordan, M. I. (2008). An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *ICML*.

[19] Long, J. L., Zhang, N. and Darrell, T. (2014). Do convnets learn correspondence? In *NIPS*.

[20] Neal, R. (2011). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*. MR2858447

[21] Rifai, S., Vincent, P., Muller, X., Glorot, X. and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML* 833–840.

[22] Roth, S. and Black, M. J. (2009). Fields of experts. *IJCV*.

[23] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. and Fei-Fei, L. (2014). ImageNet Large Scale Visual Recognition Challenge. arXiv preprint arXiv:1409.0575.

[24] Salakhutdinov, R. and Hinton, G. E. (2009). Deep Boltzmann machines. In *AISTATS*.

[25] Weston, J., Ratle, F., Mobahi, H. and Collobert, R. (2012). Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*.

[26] Zeiler, M. D. and Fergus, R. (2013). Visualizing and understanding convolutional neural networks. arXiv preprint arXiv:1311.2901.

Jifeng Dai
Microsoft Research Asia
Building 2, No. 5 Dan Ling Street
Haidian District
Beijing, 100080
P.R. China
E-mail address: jifdai@microsoft.com

Yang Lu
UCLA Department of Statistics
Los Angeles, CA 90095-1554
USA
E-mail address: yanglv@ucla.edu

Ying Nian Wu
UCLA Department of Statistics
Math Sciences Bldg.
Los Angeles, CA 90095-1554
USA
E-mail address: ywu@stat.ucla.edu