# Computer Vision I : Project 4 (10 points)

## Experiments with Learning Deep Generative Model by Alternative Back-propagation

Due on Dec 6th (Tuesday) 11:59pm

**Objective**. The objective of this project is to experience the multi-layered generator network. It is a nonlinear generalization of the factor analysis mode including the sparse coding model. This top-down generative model has the following properties:

1. **Embedding**: The model embeds the high-dimensional non-Euclidean manifold formed by the observed examples into the low-dimensional Euclidean space of the latent factors so that linear interpolation in the low-dimensional factor space results in non-linear interpolation in the data space.

2. **Analysis**: The model disentangles the variations in the observed examples into independent variations of latent factors.

3. **Synthesis**: In contrast to the bottom-up descriptive model which samples in high-dimensional space to synthesize examples, the top-down model can synthesize new examples by sampling the factors from the known low-dimensional prior distribution and transforming the factors into the synthesized examples.

## 1 Introduction to the Deep Generative Model

Refer to textbook Section 11.2 Page 253-270 for descriptions. For more advanced materials, read Chapter 12.

Notations. Let $\mathbf{I}$ be a $D$-dimensional observed example, such as an image. Let $z$ be the $d$-dimensional vector of continuous latent factors, $z = (z_k, k = 1, ..., d)$. The traditional factor analysis model is $\mathbf{I} = Wz + \epsilon$, where $W$ is $D \times d$ matrix, $\epsilon$ is a $D$-dimensional error vector or the observational noise, usually $d < D$, $z \sim \mathrm{N}(0, I_d)$, and $\epsilon \sim \mathrm{N}(0, \sigma^2 I_D)$. The deep generative model generalizes the linear mapping $Wz$ to a non-linear mapping $g(z; \theta)$, where $g$ is a ConvNet, and $\theta$ collects all the connection weights and bias terms of the ConvNet. Then the model becomes

$$\begin{aligned} &\mathbf{I} = g(z; \theta) + \epsilon, \\ &z \sim \mathrm{N}(0, I_d),\ \epsilon \sim \mathrm{N}(0, \sigma^2 I_D),\ d < D. \end{aligned} \tag{1}$$

The reconstruction error is $\|\mathbf{I} - g(z; \theta)\|^2$.

Although $g(z;\theta)$ can be any nonlinear mapping, the ConvNet parameterization of $g(z;\theta)$ makes it particularly close to the original factor analysis. Specifically, we can write the top-down ConvNet as follows:

$$z^{(l-1)} = g_l(W_l z^{(l)} + b_l), \tag{2}$$

where $g_l$ is element-wise nonlinearity at layer $l$, $W_l$ is the weight matrix, $b_l$ is the vector of bias terms at layer $l$, and $\theta = (W_l, b_l, l = 1, ..., L)$. $z^{(0)} = g(z;\theta)$, and $z^{(L)} = z$. The top-down ConvNet (2) can be considered a recursion of the original factor analysis model, where the factors at the layer $l-1$ are obtained by the linear superposition of the basis vectors or basis functions that are column vectors of $W_l$, with the factors at the layer $l$ serving as the coefficients of the linear superposition.

We employ the Alternating Back-Propagation (ABP) algorithm for maximum likelihood learning of the generator network that iterates the following two steps:

(1) *Inferential back-propagation*: For each training example, infer the continuous latent factors by Langevin dynamics.

(2) *Learning back-propagation*: Update the parameters given the inferred latent factors by gradient descent.

Specifically, the joint density is $p(z, \mathbf{I}; \theta) = p(z)p(\mathbf{I}|z;\theta)$, and

$$\log p(z, \mathbf{I}; \theta) \quad = \quad -\frac{1}{2\sigma^2}\|\mathbf{I} - g(z;\theta)\|^2 - \frac{1}{2}\|z\|^2 + \text{constant}, \tag{3}$$

where the constant term is independent of $z$ and $\theta$.

For the training data $\{\mathbf{I}_i, i = 1, ..., n\}$, the generator model can be trained by maximizing the log-likelihood

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p(\mathbf{I}_i;\theta). \tag{4}$$

The gradient of $L(\theta)$ is obtained according to the following identity

$$\frac{\partial}{\partial\theta}\log p(\mathbf{I};\theta) = \mathrm{E}_{p(z|\mathbf{I};\theta)}\left[\frac{\partial}{\partial\theta}\log p(z, \mathbf{I};\theta)\right]. \tag{5}$$

In general, the expectation in (5) is analytically intractable and has to be approximated by MCMC that samples from the posterior $p(z|\mathbf{I};\theta)$, such as the Langevin inference dynamics, which iterates

$$z_{\tau+1} = z_\tau + \frac{s^2}{2}\frac{\partial}{\partial z}\log p(z_\tau, \mathbf{I};\theta) + se_\tau, \tag{6}$$

where $\tau$ indexes the time step, $s$ is the step size, and $e_\tau$ denotes the noise term, $e_\tau \sim \mathrm{N}(0, I_d)$. for each $z_i$, only a single copy of $z_i$ is sampled from $p(z_i|\mathbf{I}_i, \theta)$ by running a finite

number of steps of Langevin dynamics starting from the current value of $z_i$, i.e., the warm start. With $z_i$ sampled from $p(z_i \mid \mathbf{I}_i, \theta)$ for each observation $\mathbf{I}_i$ by the Langevin inference process, the Monte Carlo approximation to $L'(\theta)$ is

$$\begin{aligned} L'(\theta) &\approx \frac{1}{n}\sum_{i=1}^{n} \frac{\partial}{\partial \theta} \log p(z_i, \mathbf{I}_i; \theta) \\ &= \frac{1}{n}\sum_{i=1}^{n} \frac{1}{\sigma^2}\left(\mathbf{I}_i - g(z_i;\theta)\right) \frac{\partial}{\partial \theta} g(z_i;\theta). \end{aligned} \tag{7}$$

The updating of $\theta$ solves a nonlinear regression problem so that the learned $\theta$ enables better reconstruction of $\mathbf{I}_i$ by the inferred $z_i$.

## 2 Experiment

For the lion-tiger images, you will learn a generative model with 2-dimensional vector for the latent factors . To reduce the computational complexity, all images can be down-sized into the size of $128 \times 128$ pixels.

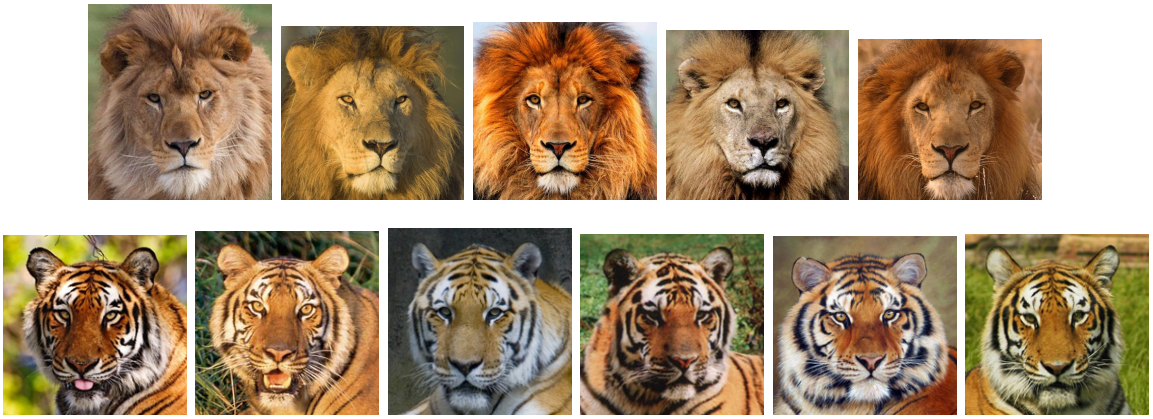

Figure 1: lion-tiger images.

Fill the blank part of ./code/generator.py. Design your own structure of the generator. To generate vivid results, you should adjust the learning rate, descretization step size of the Langevin dynamics, the number of langevin steps, and the annealing or tempering parameter $\sigma^2$ in Eq.3. Show:

1. Reconstructed images of the training images, using the inferred z from training images. Try both the warm-start scheme (running a finite number of steps of Langevin

dynamics starting from the current value of $z_i$) and cold-start scheme (running a finite number of steps of Langevin dynamics starting from the fixed prior distribution of the latent factors)

2. Randomly generated images using randomly sampled z.

3. Generated images with linearly interpolated latent factors from (-d, d). For example, you inperlolate 10 points from (-d, d) for each dimension of z. Then you will get a $10 \times 10$ panel of images. Detemine the reasonable range (-d, d), and you should be able to see that tigers slight change to lion. Try to explain whether the two Z variables can be interpretable (i.e. where they correspond to geometric or appearance transformations.

4. Plot of loss over iteration.

Write a report to show your results. And zip the report with your code.