Project 5: Generator and descriptor

Yufei Hu (404944367)

March 22, 2018

1 Generator: real inference

The model has the following form:

$$Y = f(Z;W) + \varepsilon \tag{1}$$

where Y is the synthesized image, Z is latent factors, W is model weights, and ε is reconstruction error following Normal distribution. Assume Z also follows Normal distribution.

Then, in order to maximize the observed-data log-likelihood, EM algorithm is applied to compute the log-likelihood and Langevin dynamics is used for sampling Z to approximate the posterior distribution.

The Langevin dynamics for sampling Z follows:

$$Z_{\tau+1} = Z_\tau + \delta U_\tau + \frac{\delta^2}{2}[\frac{1}{\sigma^2}(Y - f(Z_\tau;W)) \cdot \frac{\partial}{\partial Z} f(Z_\tau;W) - Z_\tau] \tag{2}$$

where τ is the time step, δ is the step size, and U_τ denotes a random vector following Normal distribution.

Based on such sampling manner, the loss function is described as follows and can be updated using stochastic gradient descent.

$$L(W) = \sum_{i=1}^{n} \frac{1}{2\sigma^2} \left\| Y_i - f(Z_i;W) \right\|^2 \tag{3}$$

A generator model with 2-dim latent factor vector is learned for synthesizing lion-tiger pictures. The network architecture for the generator model is summarized in the table below:

Table 1: Neural network architecture for generator model

Block	Parameters
FC - BN - LReLU	hidden unit size = 512×4×4
Deconv - BN - LReLU	output size = 8×8×256, stride = 2
Deconv - BN - LReLU	output size = 16×16×128, stride = 2
Deconv - BN - LReLU	output size = 32×32×64, stride = 2
Deconv - tanh	output size = 64×64×3, stride = 2

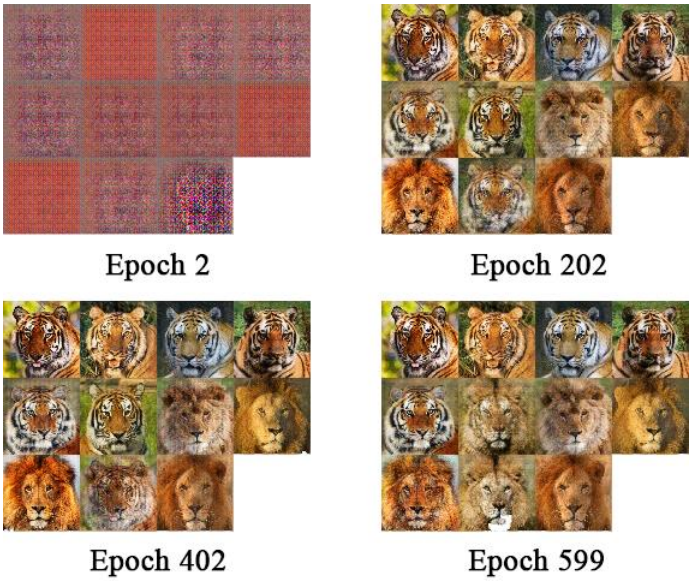

Figure 1: Reconstructed training images at different training epochs

As shown in the figure above, the reconstructed training images reach a pretty good effect at the 202nd training epoch. The images are getting more and more authentic after a sufficient number of training epochs.

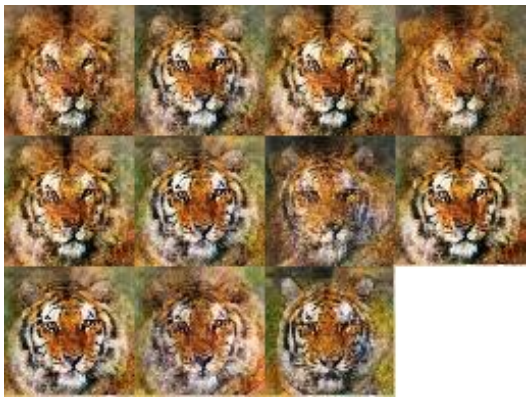

Figure 2: Randomly synthesized fake images

By randomly sampling the latent factors Z, a group of fake images are randomly synthesized as shown in the figure above. Interestingly, some pictures look like a combination of a tiger and a lion.

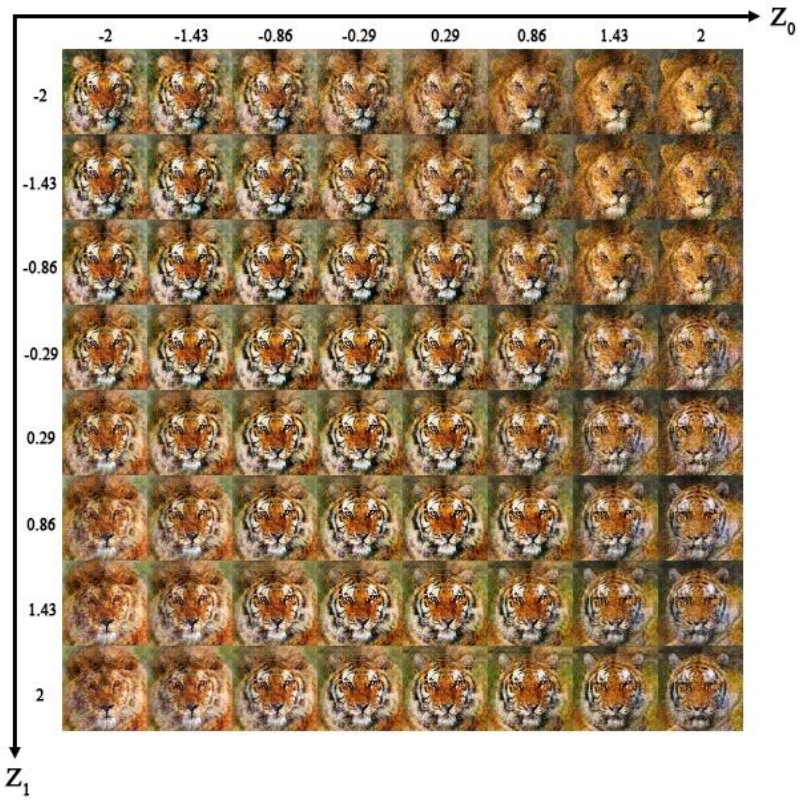

Figure 3: Synthesized images with linearly interpolated latent factors

Generated images with linearly interpolated latent factors between -2 and 2 for both dimensions are shown in the figure above. It can be seen that tiger face slightly change to lion face as Z_0 gradually changes from -2 to 2 when Z_1 is close to -2. Opposite phenomenon occurs when Z_0 gradually changes from -2 to 2 when Z_1 is close to 2. This observation indicates that the latent factors have successfully captured some key features lying in the training images.

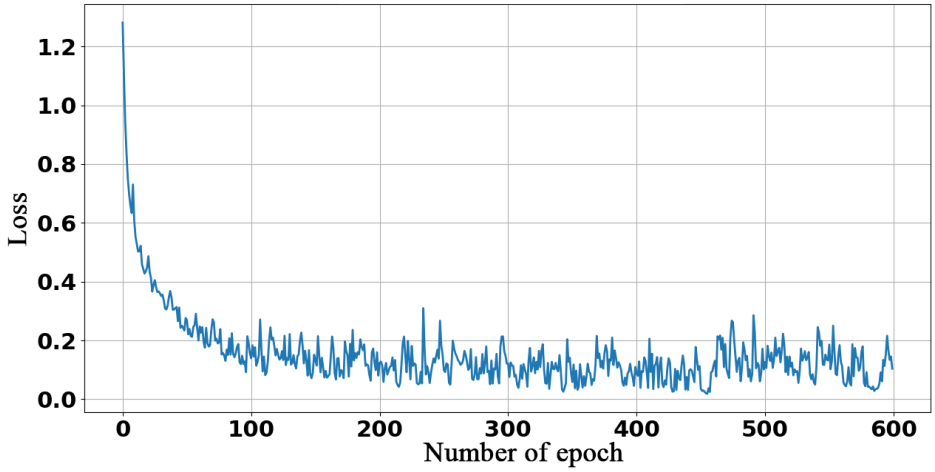

Figure 4: Training loss of GenNet

Finally, the training loss is shown in the figure above. The loss function is depicted in Equation 2. The training loss quickly decreases to around 0.2 after merely 100 training epochs, which perfectly explains the reason why training effect is already quite acceptable at the 202nd training epoch as shown in Figure 1 above.

2 Descriptor: real sampling

The descriptor model is depicted in the equation below:

$$P_\theta(Y) = \frac{1}{Z(\theta)} \cdot \mathrm{e}^{f_\theta(Y)} \cdot p_0(Y) \tag{4}$$

where $p_0(Y)$ is a reference distribution such as Gaussian white noise.

The scoring function is defined by a bottom-up convolutional neural network. The normalizing constant is intractable, thus Langevin dynamics is applied once again for approximation. The synthesized images follow the sampling steps below:

$$Y_{\tau+1} = Y_\tau - \frac{\delta^2}{2}[\frac{Y_\tau}{\sigma^2} - \frac{\partial}{\partial Y} f_\theta(Y_\tau)] + \delta \cdot U_\tau \tag{5}$$

where τ is the time step, δ is the step size, and U_τ denotes a random vector following Normal distribution. The Langevin dynamics relaxes Y_τ to a low energy region, while the noise term provides randomness and variability.

Finally, the loss function is described as follows:

$$L(\theta) = \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} f_\theta(\tilde{Y}_i) - \frac{1}{n} \sum_{i=1}^{n} f_\theta(Y_i) \tag{6}$$

To make Langevin sampling easier, mean images of the training images are used as the sampling starting point. The network architecture for the descriptor model is summarized in the table below:

Table 2: Neural network architecture for the descriptor model

Block	Parameters
Conv - BN - LReLU	filter size = 64×3×4×4, stride = 2
Conv - BN - LReLU	filter size = 128×64×2×2, stride = 1
FC	hidden unit size = 1

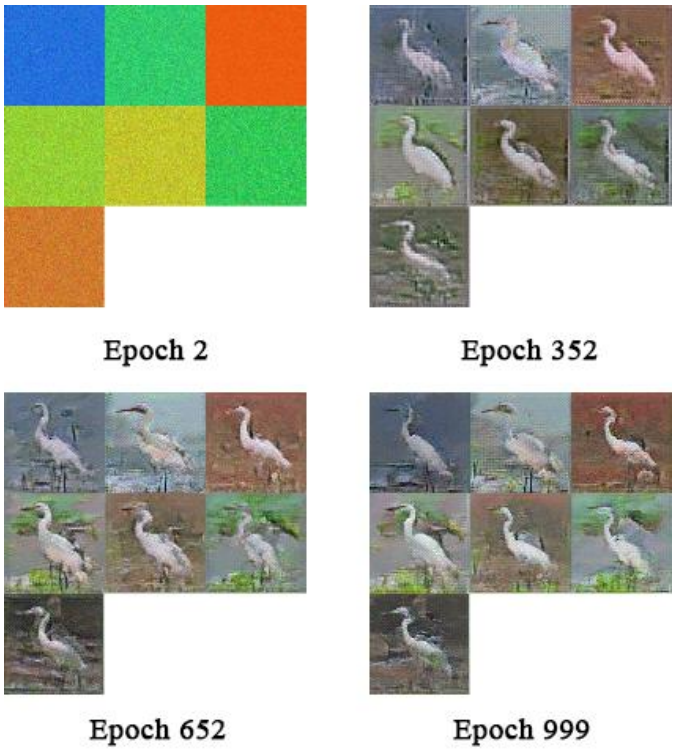

Figure 5: Reconstructed training images at different training epochs

As shown in the figure above, the reconstructed training images match the real images pretty well. The images are getting more and more authentic after a sufficient number of training epochs.

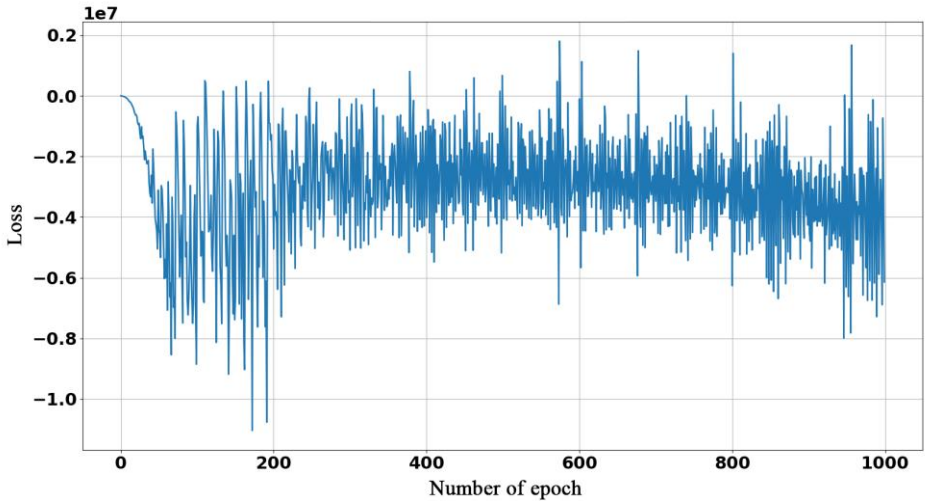

Figure 6: Training loss of DesNet

The training loss for the descriptor model is quite intimidating. It can be seen that the loss oscillates severely, reaching a scale of negative tens of millions. Despite this weird training loss observation, the training effect is quite acceptable based on the results shown in Figure 5.