ECE544NA: Variational Autoencoders & Generative Adversarial Network

Raymond Yeh

University of Illinois at Urbana Champaign

yeh17@illinois.edu

November 16, 2016

ILLINOIS

1. Generative models deal with modeling the distribution $P(X)$ defined over some datapoints $X \in \mathcal{X}$.
2. We get samples X drawn from some unknown distribution $P_{gt}(X)$, the goal is to learn a model P, which we can sample from, such that $P \approx P_{gt}$.
3. "What I cannot create, I do not understand." — Richard Feynman
4. Applications: Unsupervised feature learning, image and speech synthesis, and various image editing applications.

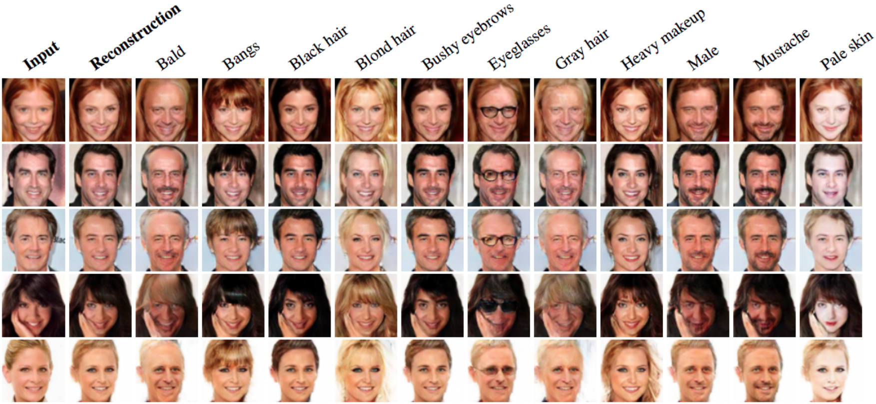
Generative Models & Motivation

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

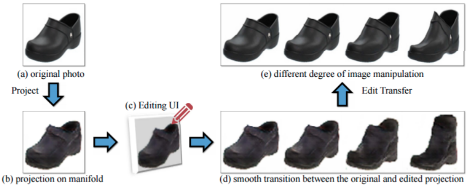
(a) (b)

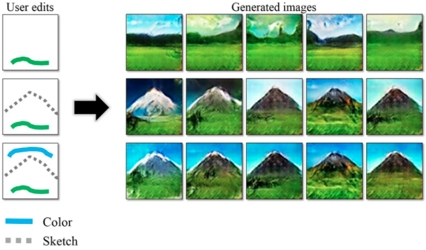
Figure : (a) Generated Images, (b) Image editing using vector arithmetic

Larsen, Anders Boesen Lindbo, Sren Kaae Snderby, and Ole Winther. "Autoencoding beyond pixels using a learned similarity metric." *arXiv preprint arXiv:1512.09300* (2015).



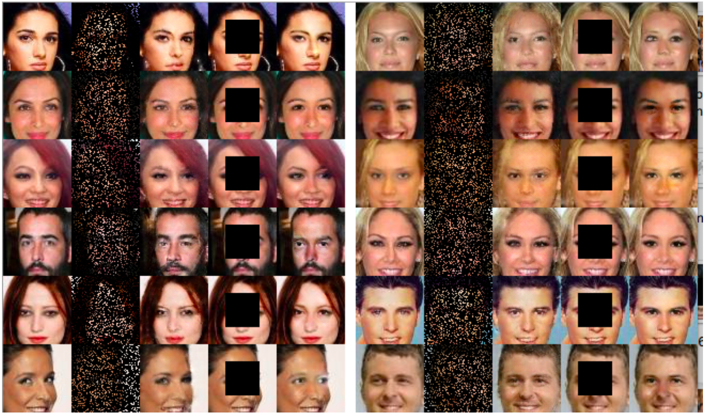
Zhu, Jun-Yan, et al. "Generative visual manipulation on the natural image manifold." *ECCV*, 2016.



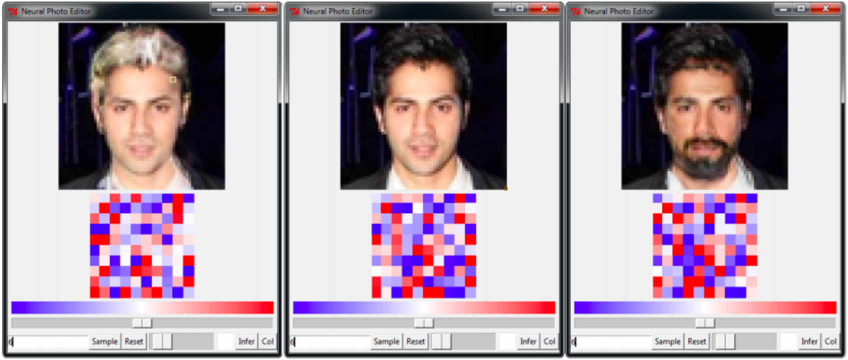


Generative Models & Motivation

Yeh, Raymond, et al. "Semantic Image Inpainting with Perceptual and Contextual Losses." *arXiv preprint arXiv:1607.07539* (2016).

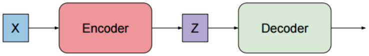


Brock, Andrew, et al. "Neural Photo Editing with Introspective Adversarial Networks." *arXiv preprint arXiv:1609.07093* (2016).

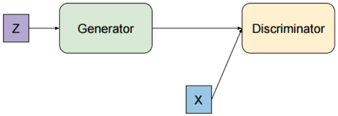


Youtube Demo: [link]

1. **Variational Autoencoders (VAEs)**
   Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114* (2013).
   Using notation in Doersch, Carl. "Tutorial on variational autoencoders." *arXiv preprint arXiv:1606.05908* (2016).



2. **Generative Advserial Networks (GANs)**
   Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in Neural Information Processing Systems.* 2014.



1. Latent Variable Models. (Assume the observed data, $X \in \mathcal{X}$ has dependencies on some hidden factors $z \in \mathcal{Z}$.)
2. Latent variables z, which we can sample from some $P(z)$, (*e.g.* Gaussian).
3. A family of parametric function $f(z;\theta)$. $f : \mathcal{Z} \times \Theta \mapsto \mathcal{X}$. (*e.g.* a neural network)
4. Optimize θ such that $f(z;\theta)$ is "like" the X in our dataset.
5. We assume the following likelihood model:

$$P(X) = \int P(X|z;\theta)P(z)dz \tag{1}$$

,where $P(X|z;\theta) = \mathcal{N}(X|f(z;\theta), \sigma^2 \times I)$.

1. We want to maximize $P(X)$, for all examples,X, in the dataset.

$$P(X) = \int P(X|z;\theta)P(z)dz \tag{2}$$

,where $P(X|z;\theta) = \mathcal{N}(X|f(z;\theta), \sigma^2 \times I)$.

2. Need to approximate the integral, as no analytical form, through sampling large number of z, $\{z_1, ..., z_n\}$.

$$P(X) \approx \frac{1}{n}\sum_{i=0}^{n} P(X|z_i) \tag{3}$$

3. This approximation is very inefficient, as a lot of the $P(X|z) \approx 0$, and does not contribute to $P(X)$. **Idea:** Use a distribution $Q(z|X)$, to produce z, where $P(X|z)$ is large.

How to relate $\mathbb{E}_{z \sim Q(z|X)} P(X|z)$ and $P(X)$?

$$D[Q(z|X)||P(z|X)] = \mathbb{E}_{z \sim Q(z|X)}[\log Q(z|X) - \log P(z|X)] \tag{4}$$

Expand $P(z|X)$ using Bayes rule

$$D[Q(z|X)||P(z|X)] = \mathbb{E}_{z \sim Q}[\log Q(z|X) - \log P(X|z) - \log P(z)] + \log P(X) \tag{5}$$

Rearrange, and writing using KL divergence

$$\log P(x) - D[Q(z|X)||P(z|X)] = \mathbb{E}_{z \sim Q}[\log P(X|z)] - D[Q(z|X)||P(z)] \tag{6}$$

1. Recall we want to maximize $\log P(X)$, instead we maximize the lower bound.

$$\log P(x) \geq \mathbb{E}_{z \sim Q}[\log P(X|z)] - D[Q(z|X)||P(z)] \tag{7}$$

   As usual, we will use gradient descent.

2. $Q(z|X)$ is assumed to have the form of $\mathcal{N}(z|\mu(X;\theta), \Sigma(X;\theta))$, then $D[Q(z|X)||P(z)]$ can be computed in closed form, and thus we can back-prop the loss.
3. How do we back-prop through $\mathbb{E}_{z \sim Q}[\log P(X|z)]$?

1. Approximate $\mathbb{E}_{z\sim Q}[\log P(X|z)]$, with a batch of z samples.
2. **The reparametrization trick:**
   Let $z \sim P(z|X) = \mathcal{N}(\mu, \sigma^2)$. Then a valid reparametrization, $z = \mu + \sigma \cdot \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$. Now one can back propagate through the z samples.
3. **Summary + common terms:**
   1. $Q(z|X)$ is the encoder, from the data space to latent space.
   2. $P(X|z)$ is the decoder, from the latent space to the data space.
   3. One can view VAE as an Auto-encoder, with a regularizer (i.e. behaves like Gaussians) on the latent variables.

1. Strong assumptions on the likelihood function. Either Gaussians or Bernoulli.
2. The Gaussian assumption tend to lead to overly smooth images, or "convergence to mean".

Figure : **Top Row:** Generated from VAE, **Bottom Row:** Generated from GAN.

3. Next, we will introduce GAN, which does not assume a likelihood function assumption on the data space, but uses a "learned" one.

1. Let $\mathcal{X}$ be the data space, and $\mathcal{Z}$ be the latent/noise space.
2. X is drawn from some unknown distribution $p_{data}(X)$.
3. $p_g(z) =$ a prior distribution on the input noise variables.
4. $G(z;\theta_g) =$ parametric differentiable function, $\mathcal{Z} \mapsto \mathcal{X}$, which we call the generator.
5. $D(X;\theta_d) =$ parametric differentiable function, $\mathcal{X} \mapsto [0,1]$, modeling the probability that X is from p_{data}, which we call the discriminator.
6. Goal of the generator, G, fool the discriminator, D, to label $G(z;\theta_g)$ to be from p_{data}. On the other hand, D tries not be be fooled.

1. Goal of the generator, G, fool the discriminator, D, to label $G(z;\theta_g)$ to be from p_{data}. On the other hand, D tries not be be fooled.
2. The above intuition can be formulated as the following minimax optimization problem with function V(D,G):

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}}[log(D(x)] + \mathbb{E}_{z\sim p_g}[1 - log(1 - D(G(z))] \tag{8}$$

3. D is trained for a binary classification problem, where samples from the dataset are labeled as 1, and generated samples are labeled as 0.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Take Away!

1. VAEs and GANs are interesting and currently very active research area.
2. VAEs and GANs are effective generative models, fast, simple to train, and scales well to large datasets.
3. Questions?