# Assignment 4

Chaehun Shin

Electrical and Computer Engineering

Seoul National University

http://data.snu.ac.kr

# Assignment Objectives

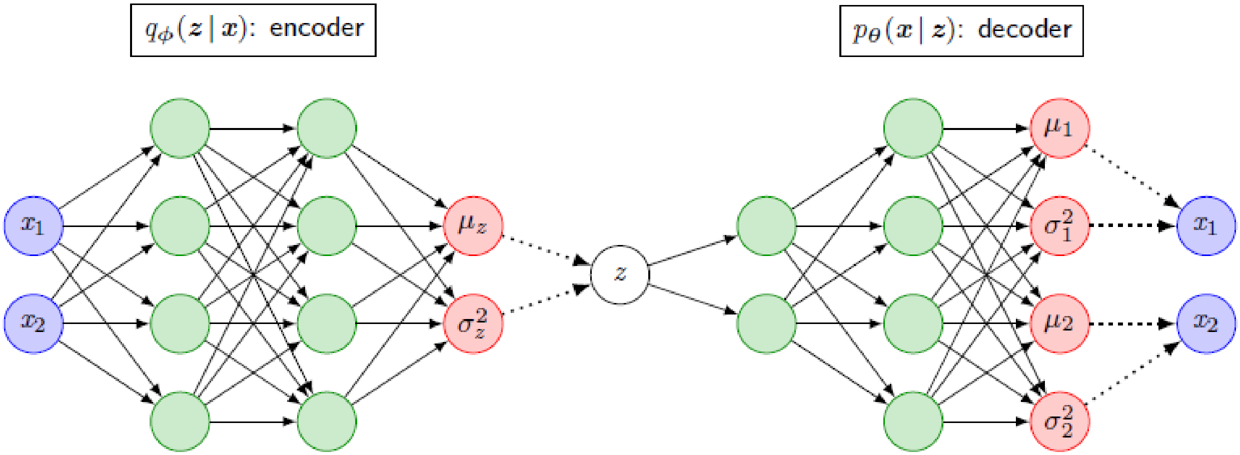
- Part 1: Implementing VAE with MNIST data

- Part 2: Implementing GANs with MNIST data

- Part 3: Implementing conditional-GANs with MNIST data

# Variational AutoEncoder(VAE)

- **Encoder**
- **Decoder**

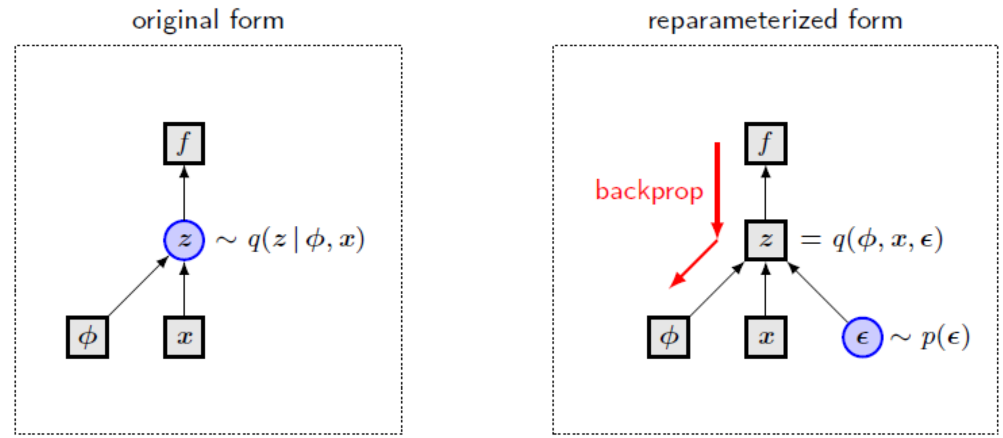



- **Elbo Loss**

$$\begin{aligned}\log p(x) &\geq \mathcal{L}(x, \theta, \phi)\\ &= \underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(z \mid x)}\left[\log p(z, x)\right]}_{①} + \underbrace{H\left[q_\phi(z \mid x)\right]}_{②\ \text{entropy}}\\ &= \underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(z \mid x)}\left[\log p_\theta(x \mid z)\right]}_{③\ \text{reconstruction}} - \underbrace{D_{\mathrm{KL}}\left[q_\phi(z \mid x) \,||\, p(z)\right]}_{④\ \text{regularizer}}\end{aligned}$$

# Variational AutoEncoder(VAE)

- **Reparameterization trick**




- **Elbo Loss**

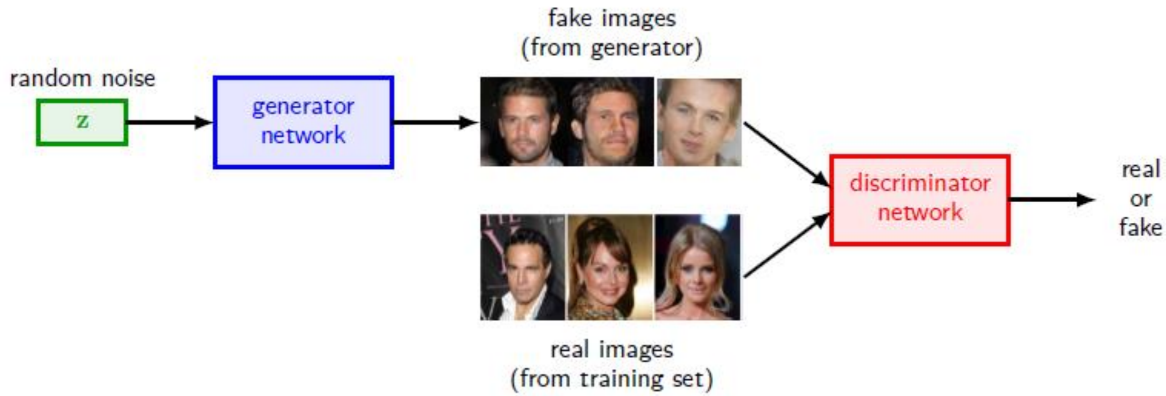
$$\begin{aligned}
\log p(x) &\geq \mathcal{L}(x, \theta, \phi) \\
&= \underbrace{\mathbb{E}_{\mathbf{z} \sim q_\phi(z \mid x)} \left[\log p(z, x)\right]}_{①} + \underbrace{H\left[q_\phi(z \mid x)\right]}_{② \text{ entropy}} \\
&= \underbrace{\mathbb{E}_{\mathbf{z} \sim q_\phi(z \mid x)} \left[\log p_\theta(x \mid z)\right]}_{③ \text{ reconstruction}} - \underbrace{D_{\mathrm{KL}}\left[q_\phi(z \mid x) \,||\, p(z)\right]}_{④ \text{ regularizer}}
\end{aligned}$$

$$\mathcal{L}(x^{(i)}, \theta, \phi) = \underbrace{-D_{\mathrm{KL}}\left[q_\phi(z \mid x) \,||\, p(z)\right]}_{\text{encoding objective}} + \underbrace{\mathbb{E}_{\mathbf{z} \sim q_\phi(z \mid x)} \left[\log p_\theta(x \mid z)\right]}_{\text{decoding objective}} \tag{3}$$

$$\simeq \frac{1}{2} \sum_{k=1}^{K} \left(1 + \ln \sigma_k^2(x^{(i)}; \phi) - \mu_k^2(x^{(i)}; \phi) - \sigma_k^2(x^{(i)}; \phi)\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_\theta \left(x^{(i)} | z^{(i,l)}\right)$$

# Generative Adversarial Networks(GANs)

- **Generator**
- **Discriminator**




- **Adversarial loss**

$$\min_{\theta_g}\max_{\theta_d}\left[\mathbb{E}_{x\sim p_{\text{data}}}\log \underbrace{D_{\theta_d}(x)}_{\text{discriminator output for real data } x} + \mathbb{E}_{z\sim p(z)}\log(1-\underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{discriminator output for generated fake data } G(z)})\right]$$

# Generative Adversarial Networks(GANs)

$$\min_{\theta_g}\max_{\theta_d}\left[\mathbb{E}_{x\sim p_{\text{data}}}\log \underbrace{D_{\theta_d}(x)}_{\text{discriminator output for real data } x}+\mathbb{E}_{z\sim p(z)}\log(1-\underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{discriminator output for generated fake data } G(z)})\right]$$

- **Discriminator**

$$\max_{\theta_d}\left[\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\log D_{\theta_d}(x)+\mathbb{E}_{\mathbf{z}\sim p(z)}\log(1-D_{\theta_d}(G_{\theta_g}(z)))\right]$$

$$J_D=\mathbb{E}_{x\sim p_{\text{data}}}\log \overbrace{D_{\theta_d}(x)}^{\text{how likely } x \text{ is real}}+\mathbb{E}_{z\sim p(z)}\overbrace{\log(1-D_{\theta_d}(G_{\theta_g}(z)))}^{\text{how likely } G(z) \text{ is fake}}$$

$$\approx\underbrace{\frac{1}{m}\sum_{i=1}^{m}\log D_{\theta_d}(x^{(i)})}_{m \text{ real samples}}+\underbrace{\frac{1}{m}\sum_{i=1}^{m}\log(1-D_{\theta_d}(G_{\theta_g}(z^{(i)})))}_{m \text{ synthetic samples}}$$

- **Generator**

$$\min_{\theta_g}\mathbb{E}_{\mathbf{z}\sim p(z)}\log(1-D_{\theta_d}(G_{\theta_g}(z)))$$

$$J_G=\mathbb{E}_{z\sim p(z)}\log(1-D_{\theta_d}(G_{\theta_g}(z)))$$

$$\approx\underbrace{\frac{1}{m}\sum_{i=1}^{m}\log(1-D_{\theta_d}(G_{\theta_g}(z^{(i)})))}_{m \text{ synthetic samples}}$$

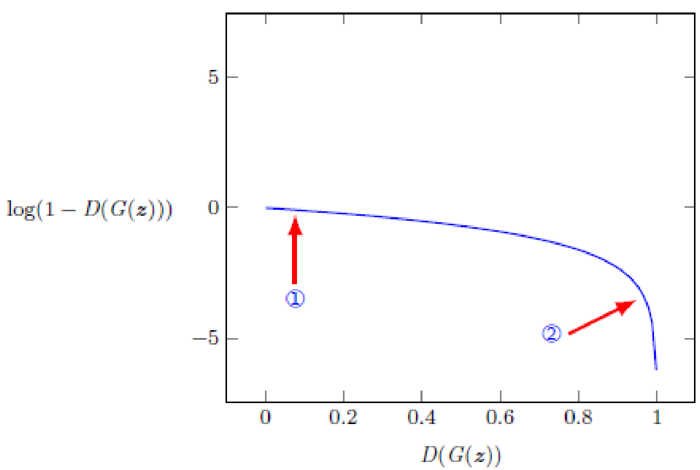
# Generative Adversarial Networks(GANs)

$$\min_{\theta_g} \max_{\theta_d} [\mathbb{E}_{x \sim p_{\text{data}}} \log \underbrace{D_{\theta_d}(x)}_{\text{discriminator output for real data } x} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{discriminator output for generated fake data } G(z)})]$$

- **Generator**

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

$$J_G = \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$
$$\approx \underbrace{\frac{1}{m}\sum_{i=1}^{m} \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)})))}_{m \text{ synthetic samples}}$$




$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

$$J_G = \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$
$$\approx \underbrace{\frac{1}{m}\sum^{m} \log(D_{\theta_d}(G_{\theta_g}(z)))}_{m \text{ synthetic samples}}$$

# Conditional Generative Adversarial Networks(cGANs)

- **Above objectives does not exploit any information.**
- **Add the label information(condition)**

- **Generator**
  - $Minimize\ -E_{c,z}[log(D(G(z,c),c)]$
- **Discriminator**
  - $Minimize\ -E_{x,c}[log(D(x,c)]\ -\ E_{c,z}\ [log(1\ -\ D(G(z,c),c)]$

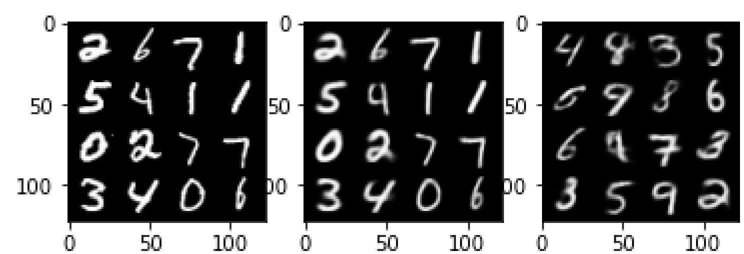
# Assignment 4 check-list

- Assignment files
  - Assignment4_1.ipynb
  - Assignment4_2.ipynb
  - Assignment4_3.ipynb
  - CollectSubmission.ipynb

# Part1 : VAE with MNIST

- **Todo parts**
  - Encoder
  - Decoder
  - Reparameterization trick
  - ELBO Loss
  - Hyperparameter (batch size should be larger than 16 for visualizing)

- **Checklist**
  - **Mnist images**
    - ✓ Reconstruction이 잘 되었는지
    - ✓ Generation이 숫자형태를 잘 나타내는지

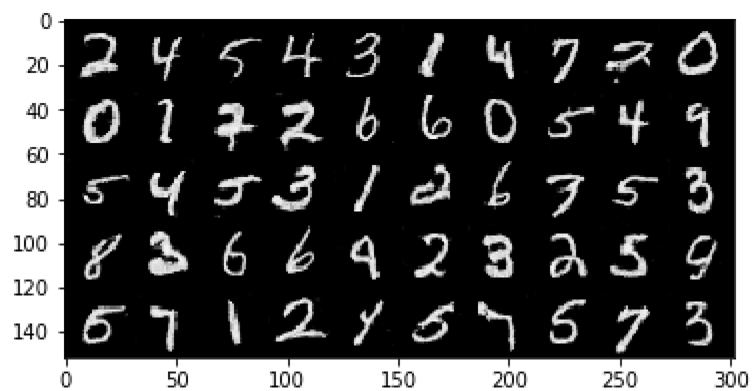



input reconstruction generation

# Part2 : GANs with MNIST

- **Todo parts**
  - Generator
  - Discriminator
  - Adversarial loss
  - Hyperparameter

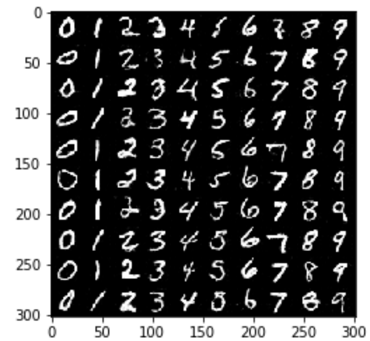
- **Checklist**
  - **Mnist images**
    - ✓ 전반적으로 숫자 형태를 띄고 있고, 어떤 숫자인지 알아볼 수 있을 정도면 충분



# Part3 : cGANs with MNIST

- **Todo parts**
  - Conditional Generator
  - Conditional Discriminator
  - Adversarial Loss
  - Hyperparameter

- **Checklist**
  - **Mnist images :** 10 images per each number -> 100 images
    - ✓ 각 class로 만든 숫자가 어떤 숫자인지 전반적으로 확인할 수 있을 정도



# Assignment4 주의사항

- **Implementation**
  – 모든 과제는 ipython file 안에서 수행할 것.
  – 학습이 잘되었으나 Visualization이 잘 안되는 것일 수도 있으니 visualize하는 image와 generate/reconstruct 된 image의 domain을 잘 맞춰야 결과가 잘 나옴. (Normalize시킬시 [0, 1] 범위로)
  – VAE과제의 경우 decoder로 Bernoulli distribution을 학습하기 위해 binary cross entropy loss를 사용해도 되고 Gaussian distribution을 학습하기 위해 L2 loss를 사용하는 것도 가능.
- **Results**
  – Ipython file내에 있는 결과를 보고 채점 할 예정.
    반드시 generate된 결과가 저장되어있는지 확인할 것.
  – Part3의 경우 최소 3개 result가 나오도록 setting 할 것.
    (학습 초기, 학습 중간, 학습 종료 후)

# FAQ

- **Skeleton code를 수정해도 되나요?**
  - 가급적이면 ToDo part 안에서 구현해주시기 바랍니다.
  - Parameter value 변경은 허용됩니다.
- **Network architecture / loss term에 대한 제한이 있나요?**
  - 없습니다. Generation을 잘하는 모델을 만들어주시기만 하면 됩니다.
  - 기본적인 GAN이 아닌 WGAN 등의 변형된 adversarial loss도 사용 가능합니다.
- **Part2, 3 에서 label을 꼭 utils.py에 있는 방식으로 주어야 하나요?**
  - 어떠한 방식으로 label을 주든 좋은 결과가 나오기만 하면 됩니다.
- **수행 시간 등이 채점에 영향을 미치나요?**
  - Ipython 내의 generation image만이 채점 대상입니다.
- **생성된 image의 quality가 어느정도여야 하나요?**
  - 전반적으로 숫자 형태임을 확인할 수 있고, conditional generation의 경우 각 class를 확인 할 수 있는 정도면 충분합니다. (예시 사진정도만 되어도 충분)


Thank You!