PixelRNN/CNN

↳ Fully visible belief network

↳ Decompose likelihood of an image x into product of 1-d distributions using chain rule:

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

$$p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

$p(x)$: likelihood of image x

$p(x_i | x_1, \ldots, x_{i-1})$: probability of ith pixel value given all previous pixels

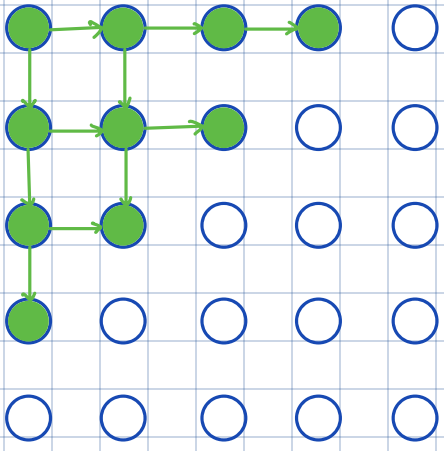
↳ Maximize likelihood of training data

PixelRNN:

↳ Generate image pixels starting from corner

↳ Dependency on previous pixels modeled using RNN (LSTM)

↳ Slow because it is sequential

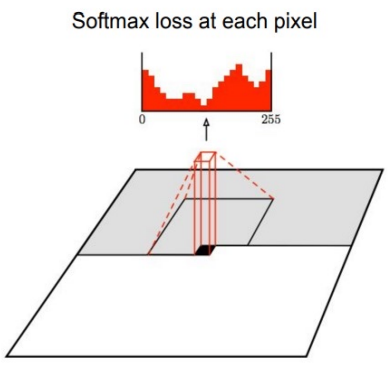


PixelCNN:

↳ Still generate image pixels starting from the corner

↳ Dependency on previous pixels now modeled using a CNN over context region

↳ Training: Maximize likelihood of training images



Variational Autoencoders (VAE)

↳ Define intractable density function with latent z:

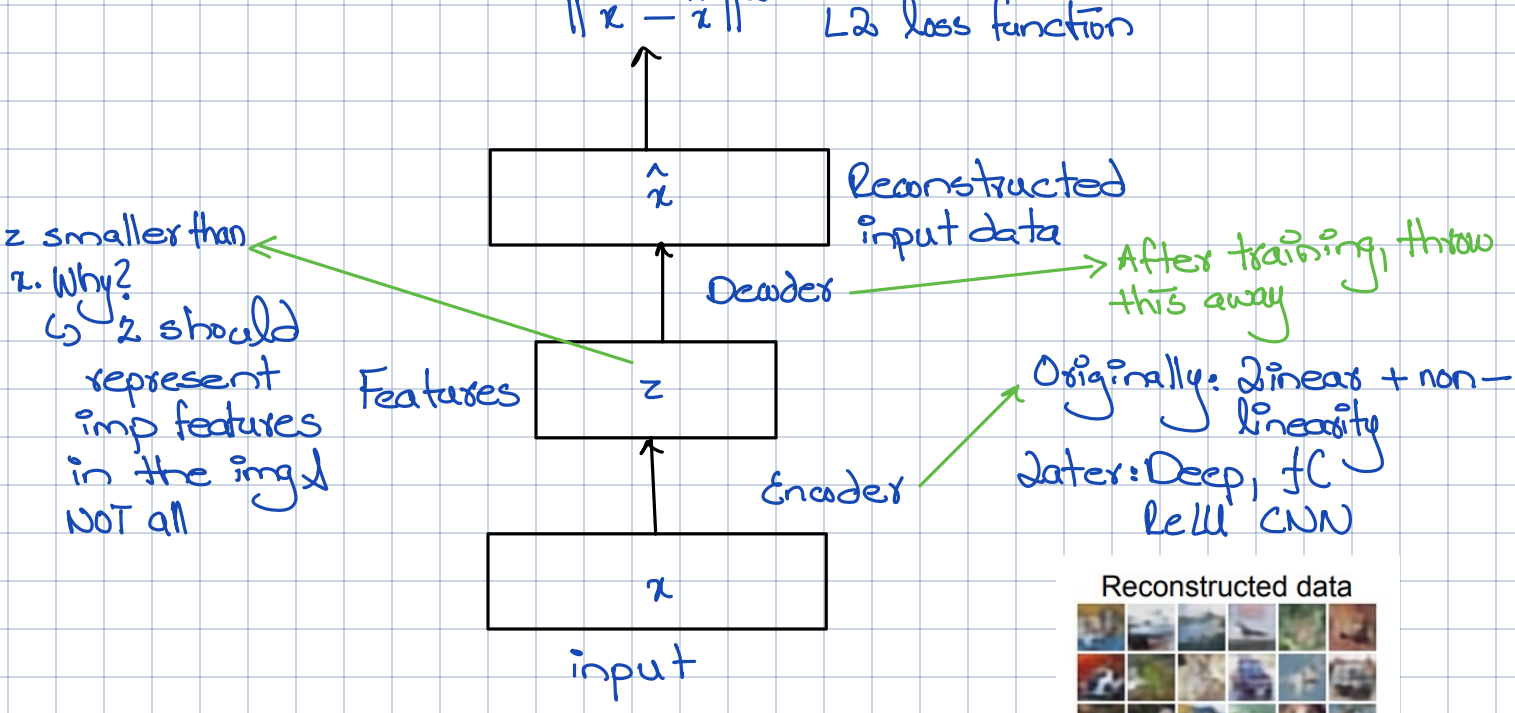
$$P_\theta(x) = \int P_\theta(z)\, P_\theta(x|z)\, dx$$

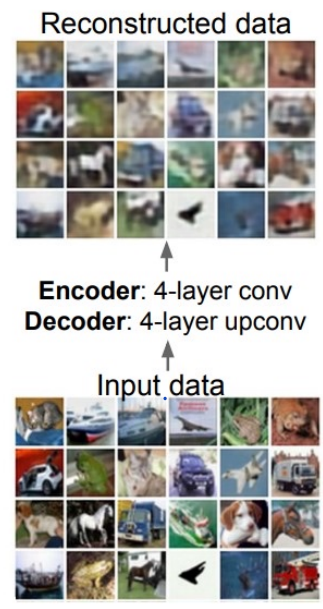
↳ Cannot optimize directly, derive & optimize lower bound on likelihood instead

Background:

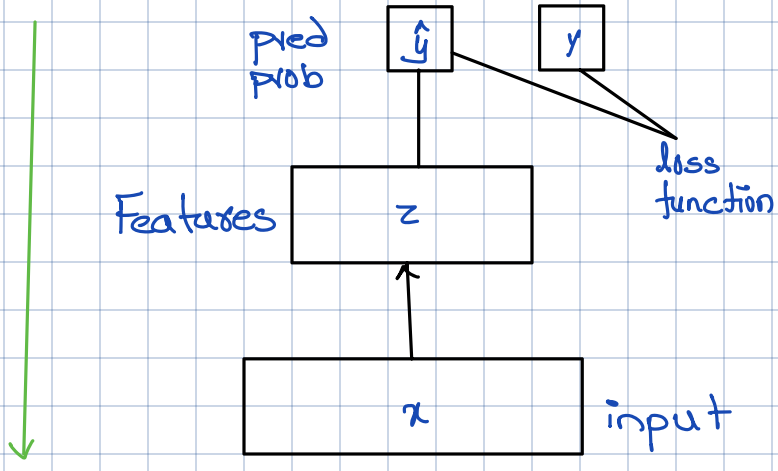
↳ VAE derived from AE

↳ AE is unsupervised approach for learning a lower dimensional feature representation from unlabeled data.





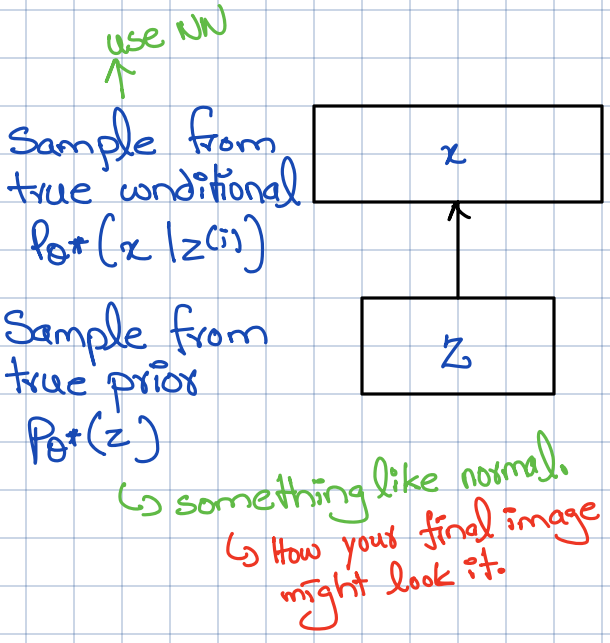
↳ We can use this initialize a supervised model



VAE:

↳ Adds probalistic spin on autoencoders — will let us sample from model to generate data!

↳ Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from underlying observed (latent) representation z



Intuition:
z contains different types of attributes. for example, how much of a smile on the face?

How to train this network?

↳ Learn model params to maximize likelihood of training data

$$p_\theta(x) = \int p_\theta(z)\, p_\theta(x|z)\, dz$$

computing $P(x|z)$ for every z is intractable

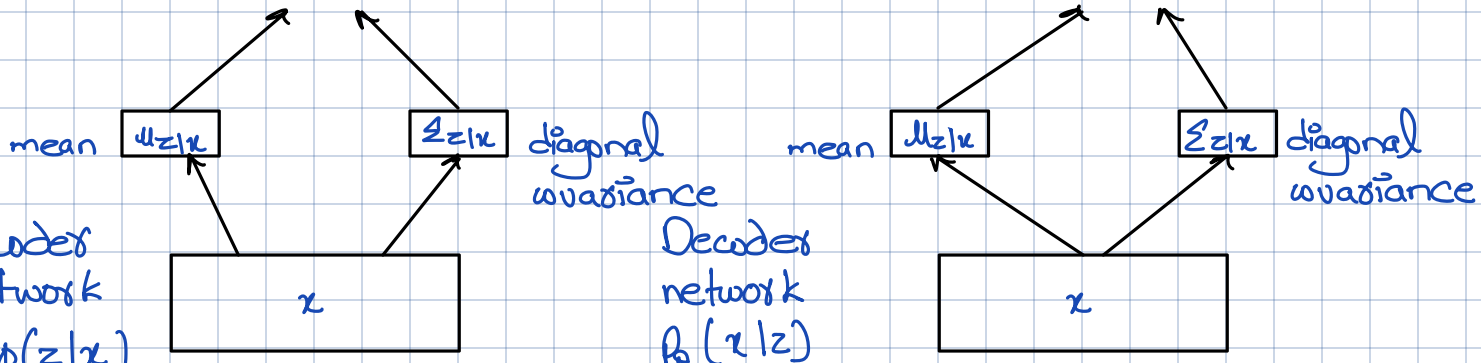
Q. What is the problem?
↳ This is NOT tractable
↳ Complex ∧ computationally expensive

↳ Posterior density also intractable: $p(z|x) = \dfrac{p_\theta(x|z)\, p_\theta(z)}{p_\theta(x)}$

Solution:
↳ Use a NN (of courses!) called encoder network $q_\phi(z|x)$ that approx $p(z|x)$

↳ Will allow to derive a lower bound on the data likelihood that is tractable, which we can optimize

sample z from $z|x \sim N(\mu_{z|x}, \Sigma_{z|x})$ sample $x|z$ from $x|z \sim N(\mu_{x|z}, \Sigma_{x|z})$



Encoder network $q_\phi(z|x)$

params ϕ

Decoder network $p_\theta(x|z)$

params θ

↳ Encoder & decoder network also called recognition / inference & generation networks

Now,

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

Reconstruct the input data

Make approximate posterior distribution close to prior

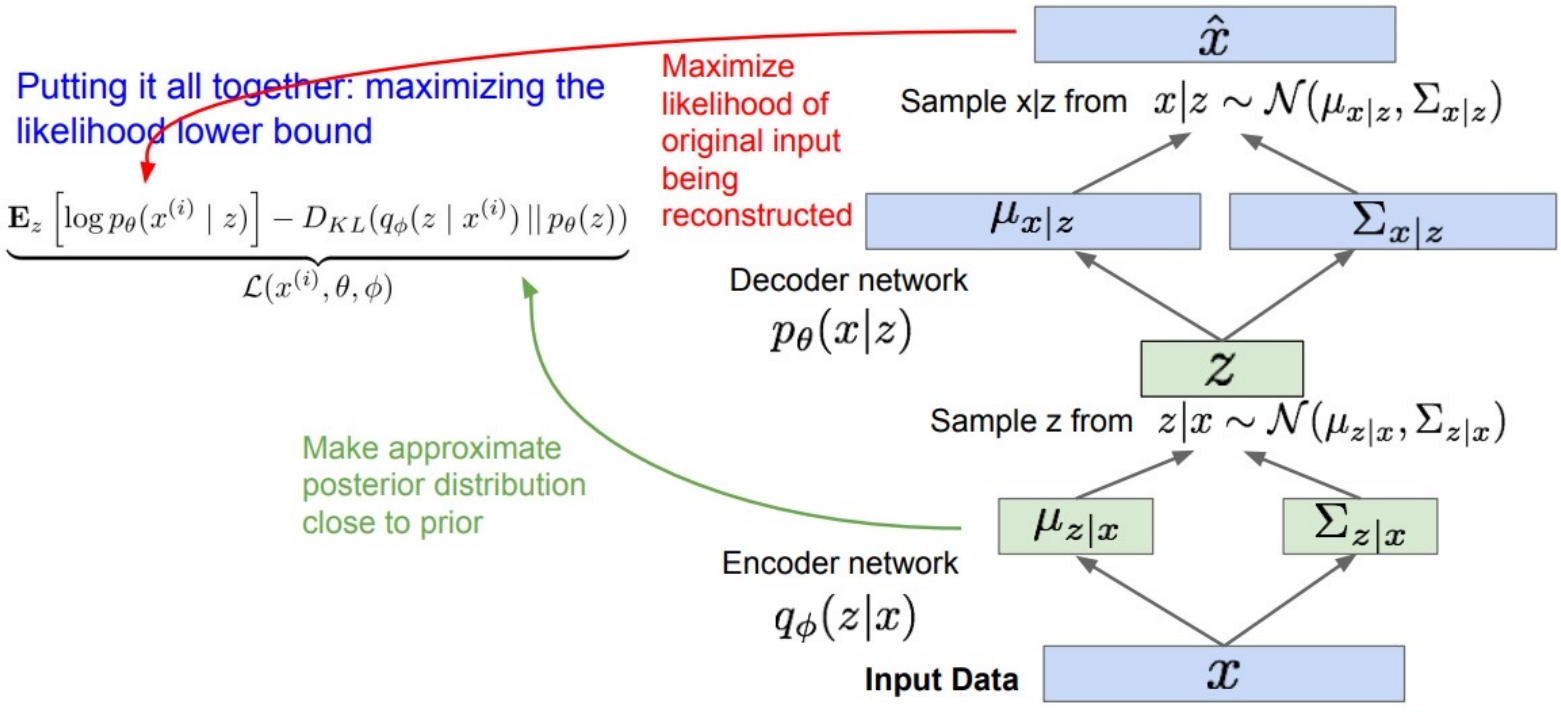
$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound ("ELBO")

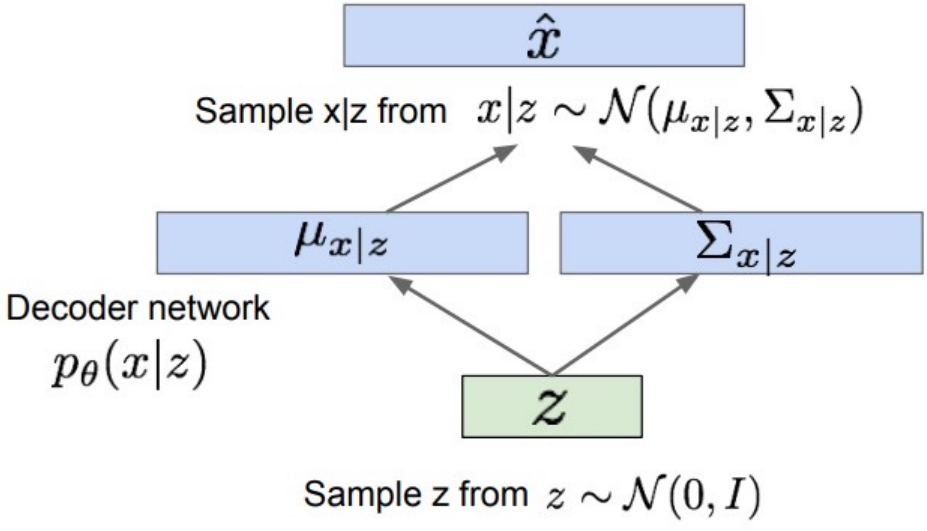
$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

Training data:

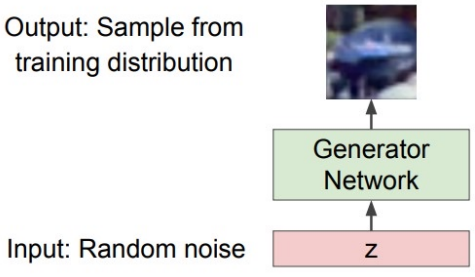


Generating data:



Generative Adversial networks:

↳ Pass input of random noise through RAN
↳ Get an output corresponding to sample from training distribution

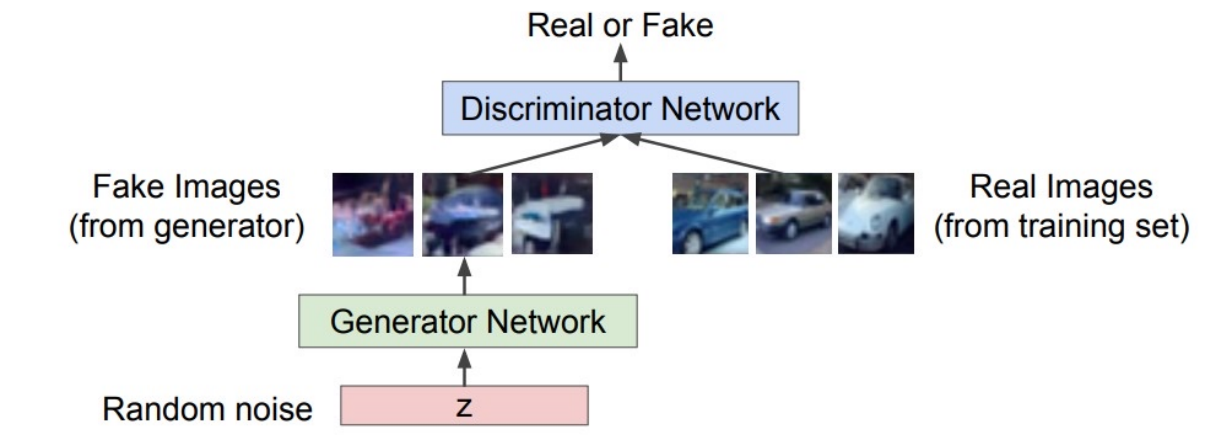


Training GANs:

↳ A two player game

Generator network: Try to fool the discriminator by generator real looking images → Player 1

Discriminator network: Try to distinguish b/w real & fake image → Player 2



↳ Train jointly in a minimax game.

↳ Minimax objective function:

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

$D_{\theta_d}(x)$: Discriminator output for real data x

$D_{\theta_d}(G_{\theta_g}(z))$: Discriminator output for fake data $G(z)$

↳ Discriminator outputs likelihood $(0,1)$ of real image

↳ Discriminator wants to maximize objective such that $D(x) \approx 1$ & $D(G(z)) \approx 0$

↳ Generator (θ_g) wants to minimize objective such that $D(G(z)) \approx 1$

Alternate b/w:

1) Gradient ascent on discriminator:

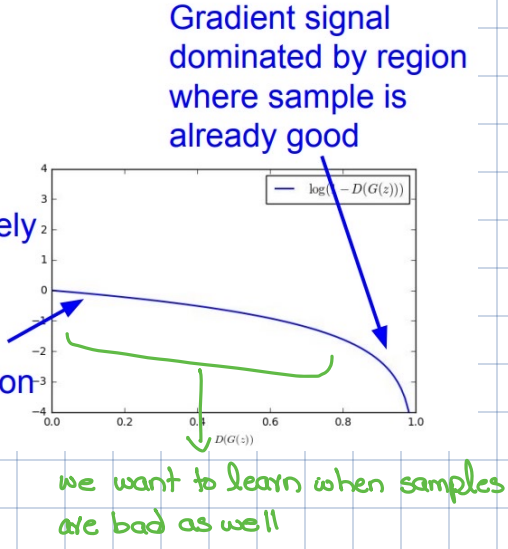
$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log \left(1 - D_{\theta_d}(G_{\theta_g}(z))\right) \right]$$

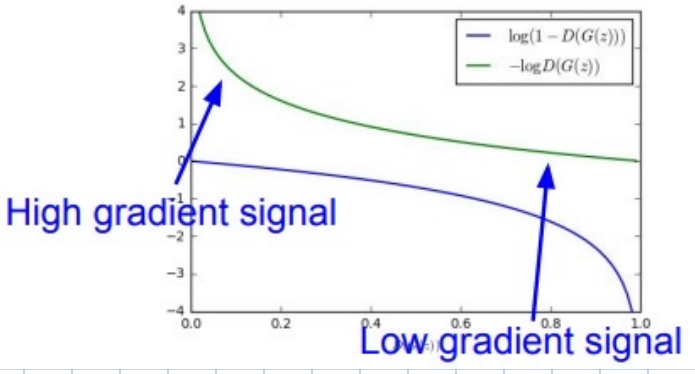
2) Gradient descent on generator:

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

So, we do gradient ascent.

Instead of minimizing the probability of the discriminator being correct, we maximize the probability of the discriminator being incorrect.





2. Gradient ascent on generator

$$\max_{\theta_g} \mathbb{E}_{z\sim p(z)} -\log\left(D_{\theta_g}(z)\right)$$

Algorithm for training GANs:

for number of training iterations **do**

  for k steps **do**

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

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

  end for

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

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

  end for