Generative Models

GAN

Generative Adversarial Nets

explicit $p(x)$

Implicit Generative Model

- **Goal:** a sampler $g(\cdot)$ to generate images
- A simple generator $g(z;\theta)$:
  - $z \sim N(0,I)$
  - $x = g(z;\theta)$ deterministic transformation

$\{x_1, \ldots, x_N\}$

- Likelihood-free training:
  - Given a dataset from some distribution p_{data}
  - Goal: $g(z;\theta)$ defines a distribution, we want this distribution $\approx p_{data}$
  - Training: minimize $D(g(z;\theta), p_{data})$
    - D is some distance metric (not likelihood)
  - Key idea: ***Learn* a differentiable** D

Metrics:
(1) Kullback-Leibler divergence (KL-divergence)
(2) Total variation
(3) Wasserstein distance
- - - -

GAN (Goodfellow et al., '14)

- Parameterize the discriminator $D(\cdot\,; \phi)$ with parameter ϕ

- **Goal:** learn ϕ such that $D(x; \phi)$ measures how likely x is from p_{data}
  - $D(x, \phi) = 1$ if $x \sim p_{data}$
  - $D(x, \phi) = 0$ if $x! \sim p_{data}$
  - a.k.a., a binary classifier
- GAN: use a neural network for $D(\cdot\,; \phi)$

- **Training:** need both negative and positive samples
  - Positive samples: just the training data $\{x_1, \ldots, x_N\}$
  - Negative samples: use our sampler $g(\cdot\,; z)$ (can provide infinite samples).

$D(\cdot, \cdot) \in [0, 1]$

- **Overall objectives:**
  - Generator: $\theta^* = \max_{\theta} D(g(z; \theta); \phi)$
  - Discriminator uses MLE Training:

$$\phi^* = \max_{\phi} \mathbb{E}_{x \sim p_{data}}[\log D(x; \phi)] + \mathbb{E}_{\hat{x} \sim g(\cdot)}[\log(1 - D(\hat{x}; \phi))]$$

GAN (Goodfellow et al., '14)

- Generator $G(z;\theta)$ where $z \sim N(0,I)$
  - Generate realistic data

- Discriminator $D(x;\phi)$
  - Classify whether the data is real (from p_{data}) or fake (from G)

- Objective function:

$$L(\theta,\phi) = \min_{\theta}\max_{\phi} \mathbb{E}_{x\sim p_{data}}\left[\log D(x;\phi)\right] + \mathbb{E}_{\hat{x}\sim G}\left[\log(1 - D(\hat{x};\phi))\right]$$

- Training procedure:
  - Collect dataset $\{(x,1)\,|\,x \sim p_{data}\} \cup \{(\hat{x},0) \sim g(z;\theta)\}$
  - Train discriminator
    $D : L(\phi) = \mathbb{E}_{x\sim p_{data}}\left[\log D(x;\phi)\right] + \mathbb{E}_{\hat{x}\sim G}\left[\log(1 - D(\hat{x};\phi))\right]$
  - Train generator $G : L(\theta) = \mathbb{E}_{z\sim N(0,I)}\left[\log D(G(z;\theta),\phi)\right]$
  - Repeat

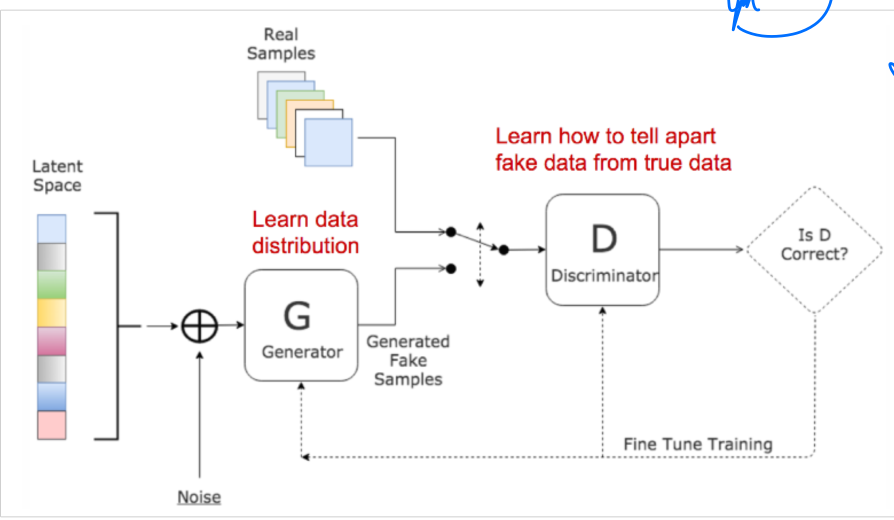
GAN (Goodfellow et al., '14)

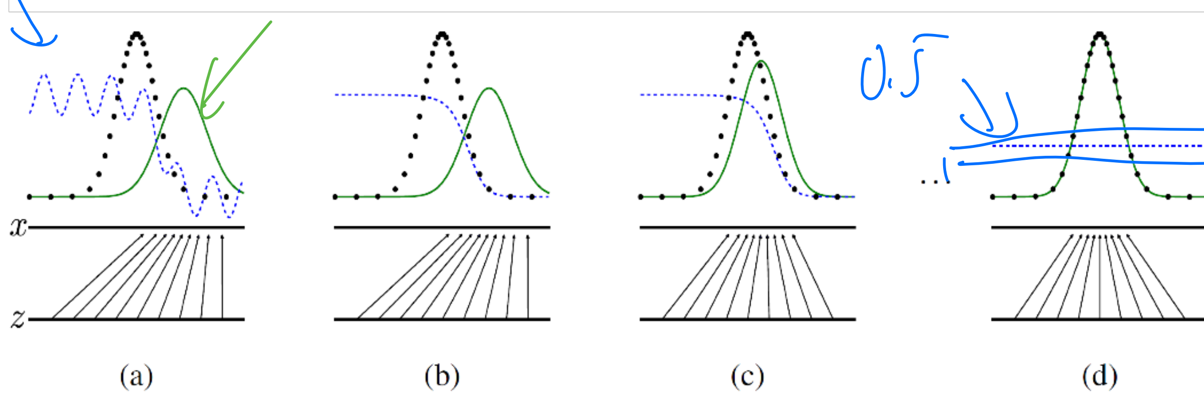
- Objective function:

$$L(\theta, \phi) = \min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p_{data}} \left[\log D(x; \phi)\right] + \mathbb{E}_{\hat{x} \sim G} \left[\log(1 - D(\hat{x}; \phi))\right]$$

D

Generator





Math Behind GAN

$$L(\theta, \phi) = \min_{\theta} \max_{\phi} \mathbb{E}_{x \sim P_{data}} [\log D(x; \phi)] + \mathbb{E}_{\hat{x} \sim g(\cdot)} [\log(1 - D(\hat{x}; \phi))]$$

Let D^*, g^* be the solution to J

Optimal D^*, for a given x

$$L_x(D) = P_{data}(x) \cdot \log D(x) + P_g(x) \cdot \log(1 - D(x))$$

$$\Rightarrow \frac{\partial L}{\partial D} = 0 \text{, first-order condition}$$

$$\Rightarrow \frac{P_{data}(x)}{D^*(x)} - \frac{P_g(x)}{1 - D^*(x)} = 0$$

$$\Rightarrow D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)}$$

(if $P_{data} = P_g$, $D^*(x) = 0.5$)

Math Behind GAN

Consider optimal generator g^*, given optimal D^*

$$L(\theta,\phi) = \mathbb{E}_{x\sim P_{data}}\left[\log \frac{P_{data}(x)}{P_{data}(x)+P_g(x)}\right] + \mathbb{E}_{\hat{x}\sim g}\left[\log \frac{P_g(\hat{x})}{P_{data}(\hat{x})+P_g(\hat{x})}\right]$$

(will show $g^* = P_{data}$)

$$= \mathbb{E}_{x\sim P_{data}}\left[\log\left(\frac{P_{data}(x)}{\left(\frac{P_{data}(x)+P_g(x)}{2}\right)}\right)\right] - \log 2$$

$$+ \mathbb{E}_{\hat{x}\sim g}\left[\log\left(\frac{P_g(\hat{x})}{\left(\frac{P_{data}(\hat{x})+P_g(\hat{x})}{2}\right)}\right)\right] - \log 2$$

$KL(p\|q) = \mathbb{E}_p\left[\log \frac{p}{q}\right]$

$$= KL\left(P_{data} \,\middle\|\, \tfrac{1}{2}(P_{data}+P_g)\right) + KL\left(P_g \,\middle\|\, \tfrac{1}{2}(P_{data}+P_g)\right) - \log 4$$

2 · Jensen-Shannon Divergence (JSD)

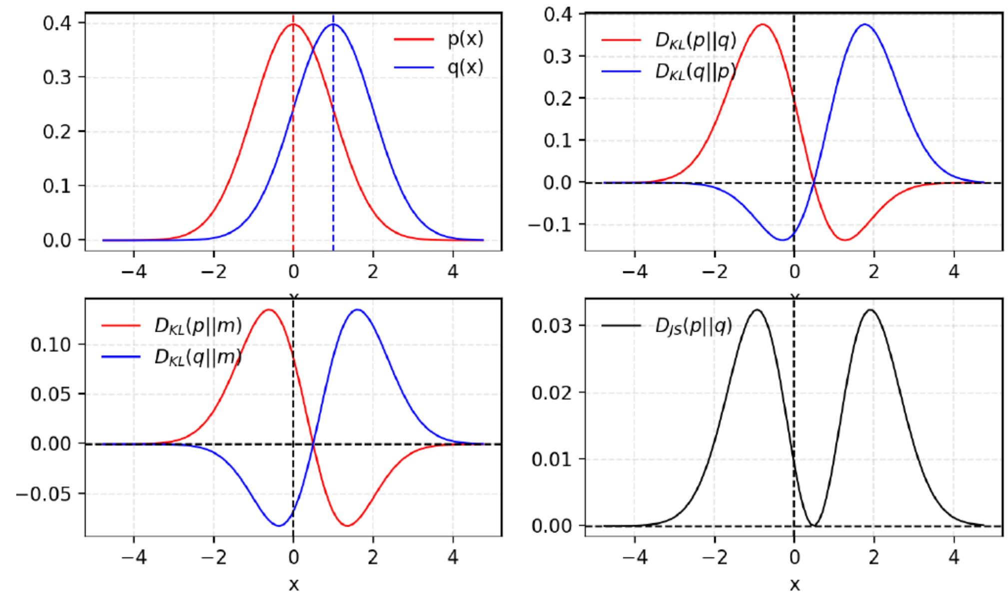
KL-Divergence and JS-Divergence

$\rightarrow KL(p||q) = \mathbb{E}_{x \sim p}\left[\log \frac{p(x)}{q(x)}\right]$

$\Rightarrow$ not symmetric

$\rightarrow JSD(p||q)$
$= \frac{1}{2}\left(KL\left(p \middle|\middle| \frac{p+q}{2}\right) + KL\left(q \middle|\middle| \frac{p+q}{2}\right)\right)$

$\rightarrow$ symmetric



$JSD \geq 0, \quad JSD(p||q) = 0 \Leftrightarrow p = q$

Math Behind GAN

$\Rightarrow$ Given D^*

$$\min_g L(g) = 2\,JSD(P_g \| P_{data}) - \log 4$$

$\Rightarrow$ global minimizer g^*:

$$g^* = P_{data}$$

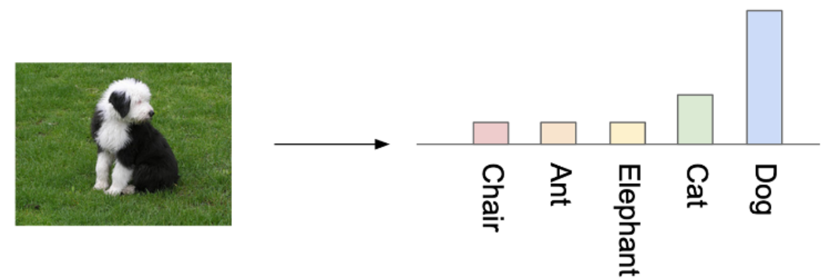
D

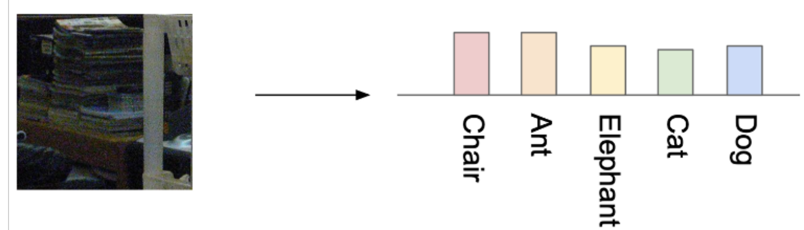
Evaluation of GAN

$f(y|x) \in \mathbb{R}^K$

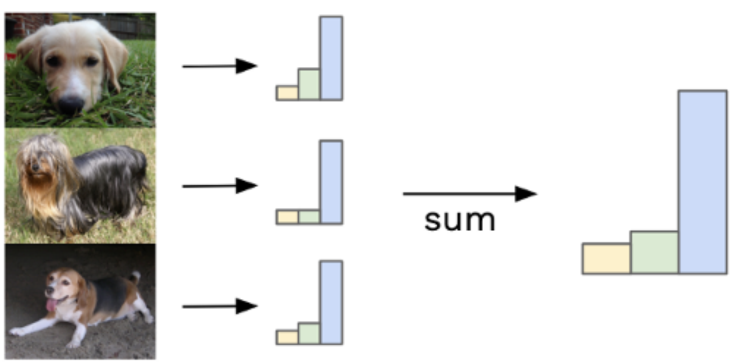
- No $p(x)$ in GAN.
- Idea: use a trained classifier $f(y \mid x)$:
- If $x \sim p_{data}$, $f(y|x)$ should have low entropy
  - Otherwise, $f(y \mid x)$ close to uniform.
- Samples from G should be diverse:
  - $p_f(y) = \mathbb{E}_{x \sim G}[f(y|x)]$ close to uniform.

K-classes $\in \mathbb{R}^K$

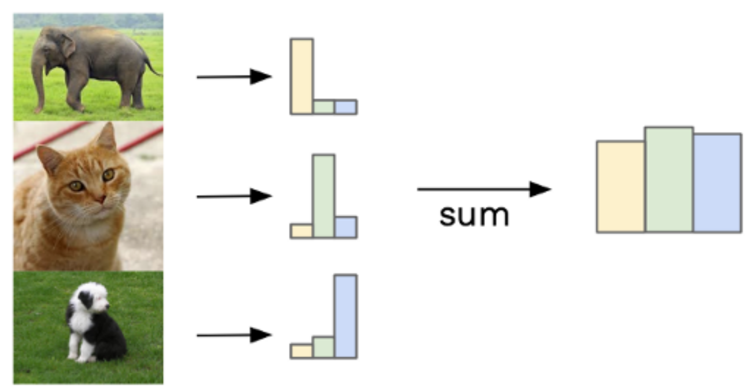




Similar labels sum to give focussed distribution



Different labels sum to give uniform distribution

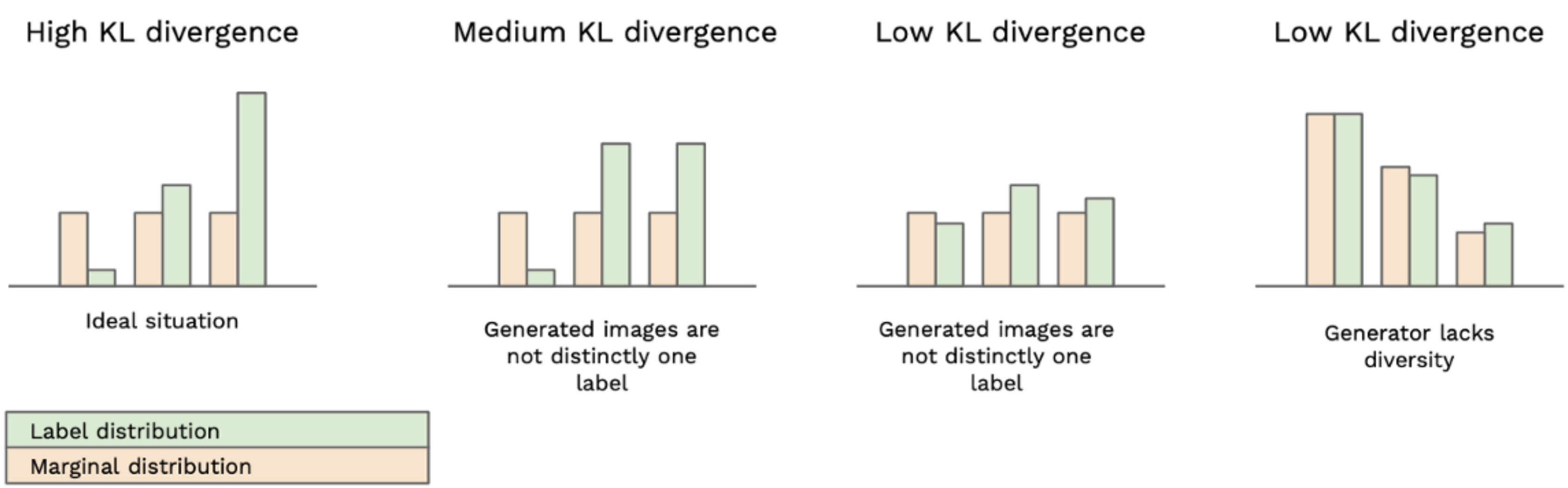


Evaluation of GAN

- **Inception Score** (IS, Salimans et al. '16)
  - Use Inception V3 trained on ImageNet as $f(y|x)$
  - $IS = \exp\left(\mathbb{E}_{x\sim G}\left[KL(f(y|x)\,||\,p_f(y))\right]\right)$
  - Higher the better

one-sample

marginal



Comments on GAN

- Other evaluation metrics:
  - Fréchet Inception Distance (FID): Wasserstein distance between Gaussians

- Mode collapse:
  - The generator only generate a few type of samples.
  - Or keep oscillating over a few modes.

- Training instability:
  - Discriminator and generator may keep oscillating
  - Example: $-xy$, generator x, discriminatory y. NE: $x = y = 0$ but GD oscillates.
  - No stopping criteria.
  - Use Wsserstein GAN (Arjovsky et al. '17):
    $$\min_{G} \max_{f:\text{Lip}(f)\leq 1} \mathbb{E}_{x\sim p_{data}}\left[f(x)\right] - \mathbb{E}_{\hat{x}\sim p_G}[f(\hat{x})]$$
  - And need many other tricks…

Variational Autoencoder

auto-encoder $D(E(x)) \approx x$

Architecture

- Auto-encoder: $x \rightarrow z \rightarrow x$
- Encoder: $q(z \mid x; \phi) : x \rightarrow z$
- Decoder: $p(x \mid z; \theta) : z \rightarrow x$

NN

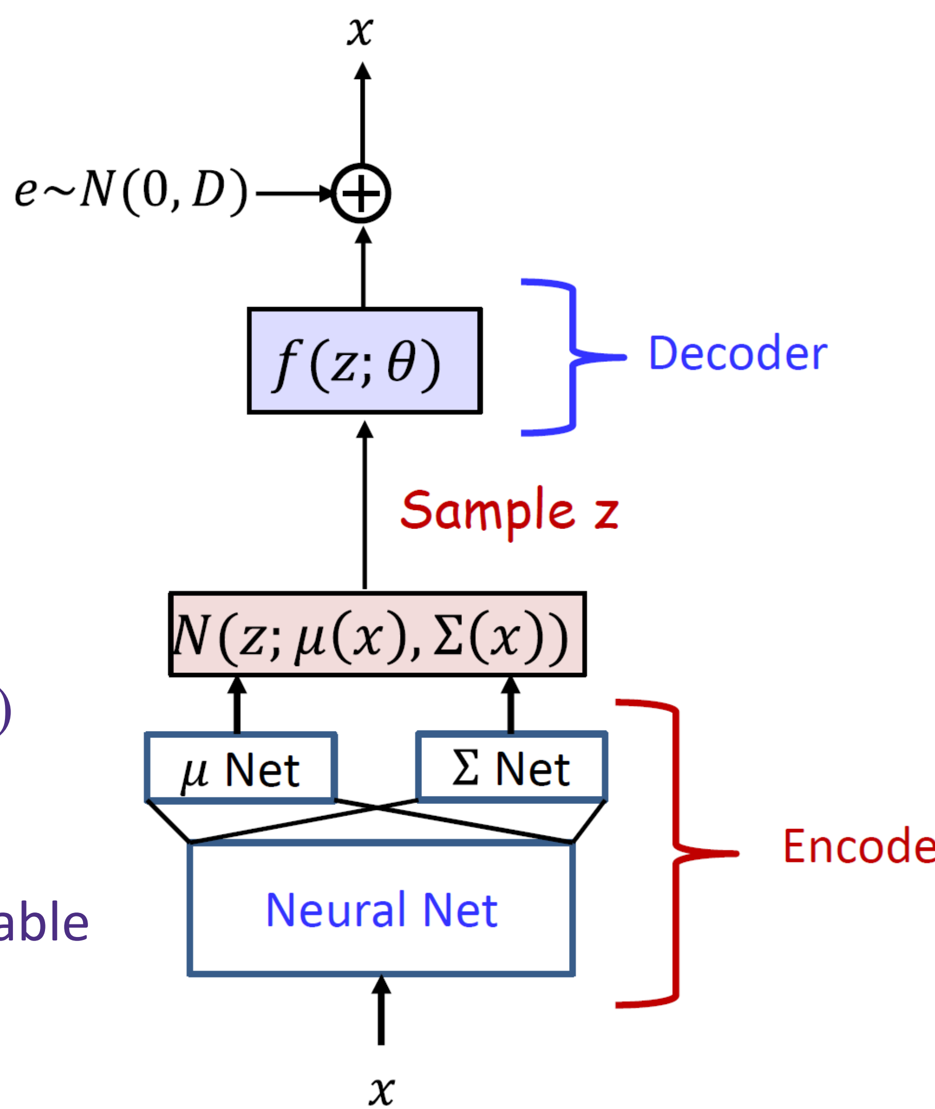
- Isomorphic Gaussian:

$q(z \mid x; \phi) = N(\mu(x; \phi), \mathrm{diag}(\exp(\sigma(x; \phi))))$

- Gaussian prior: $p(z) = N(0, I)$
- Gaussian likelihood: $p(x \mid z; \theta) \sim N(f(z; \theta), I)$

NN

- Probabilistic model interpretation: latent variable model.

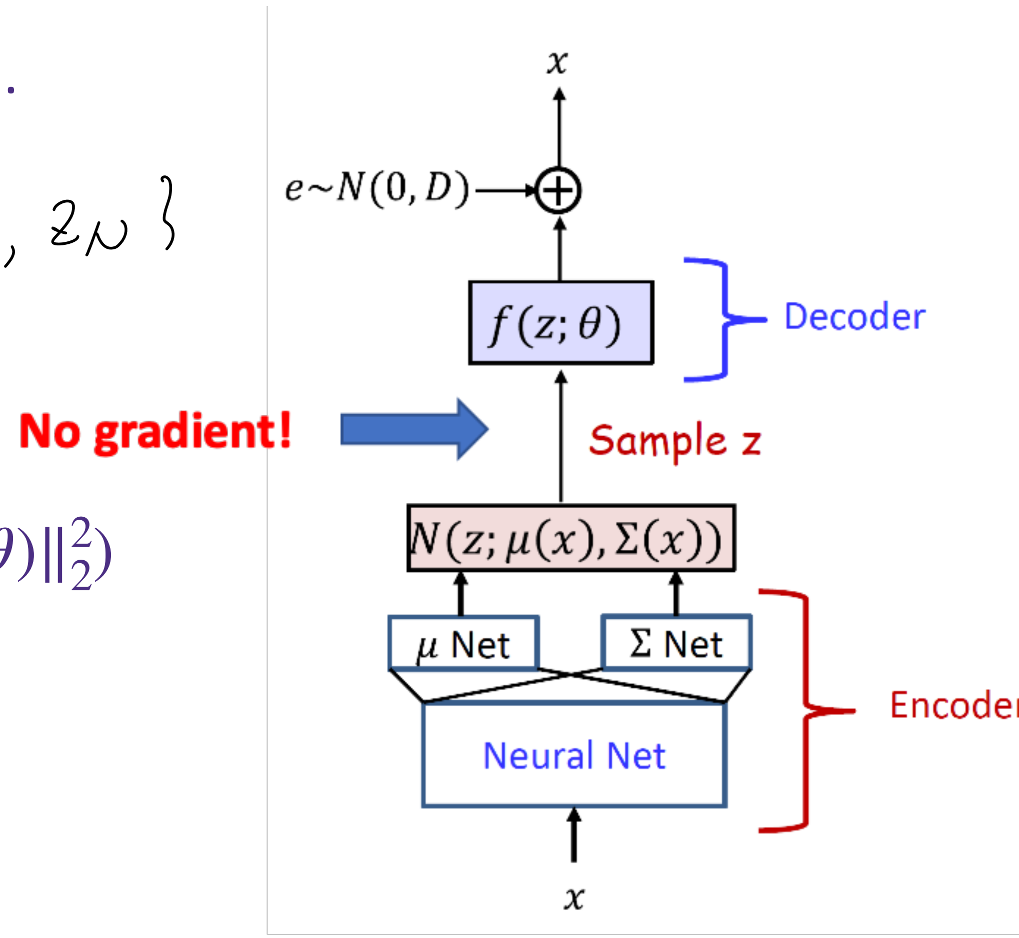


VAE Training

induced from variational inference

- Training via optimizing ELBO
  - $L(\phi, \theta; x) = \mathbb{E}_{z \sim q(z|x;\phi)}[\log p(z|x;\theta)] - KL\left(q(z|x;\phi)\,||\,p(z)\right)$
  - Likelihood term + KL penalty

- KL penalty for Gaussians has closed form.
- Likelihood term (reconstruction loss):
  - Monte-Carlo estimation
  - Draw samples from $q(z|x;\phi)$ $\{z_1, \ldots, z_N\}$
  - Compute gradient of θ:
    - $x \sim N(f(z;\theta); I)$
    - $p(x) = \dfrac{1}{\sqrt{2\pi}} \exp(-\dfrac{1}{2}\|x - f(z;\theta)\|_2^2)$



$\{X_1, \ldots, X_n\}$

VAE Training

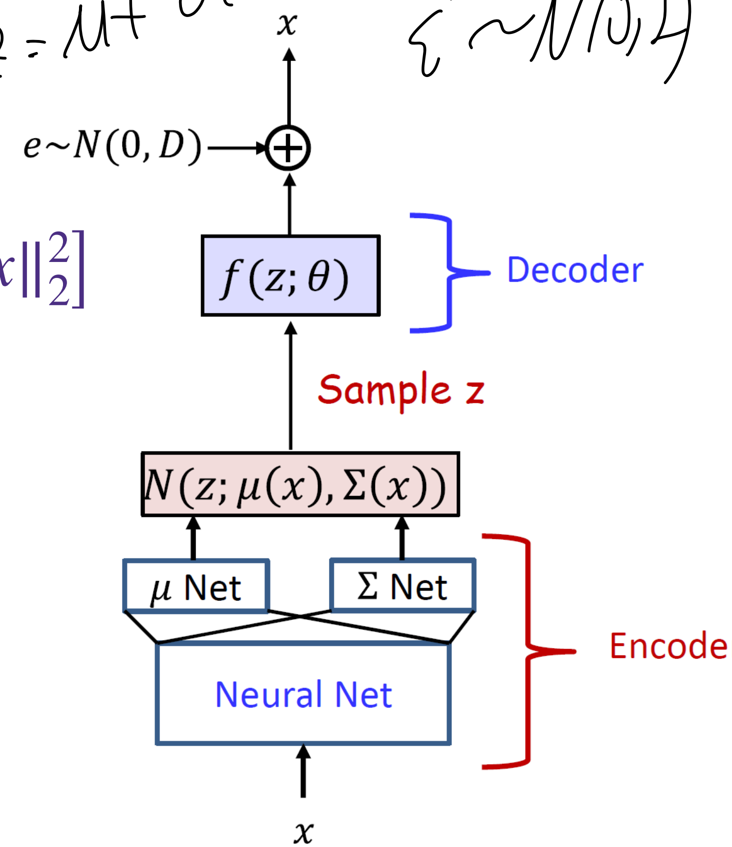
- Likelihood term (reconstruction loss):
  - Gradient for ϕ . Loss: $L(\phi) = \mathbb{E}_{z\sim q(z;\phi)}\left[\log p(x\,|\,z)\right]$
  - Reparameterization trick:
    - $z \sim N(\mu, \Sigma) \Leftrightarrow z = \mu + \epsilon, \epsilon \sim N(0,\Sigma)$
  - $L(\phi) \propto$-$\mathbb{E}_{z\sim q(z|\phi)}\left[\|f(z;\theta) - x\|_2^2\right]$

    $\propto$-$\mathbb{E}_{\epsilon\sim N(0,I)}\left[\|f(\mu(x;\phi) + \sigma(x;\phi)\cdot\epsilon;\theta) - x\|_2^2\right]$
  - Monte-Carlo estimate for $\nabla L(\phi)$

Given x, Sample $\varepsilon_1, \ldots \varepsilon_N$

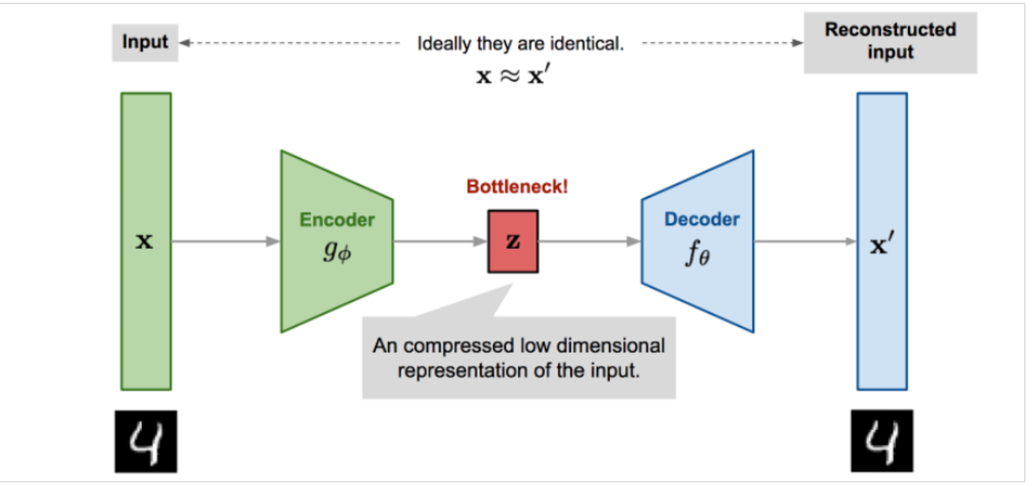
$z = \mu + \sigma(x,\phi)\cdot\varepsilon'$, $\varepsilon' \sim N(0,I)$

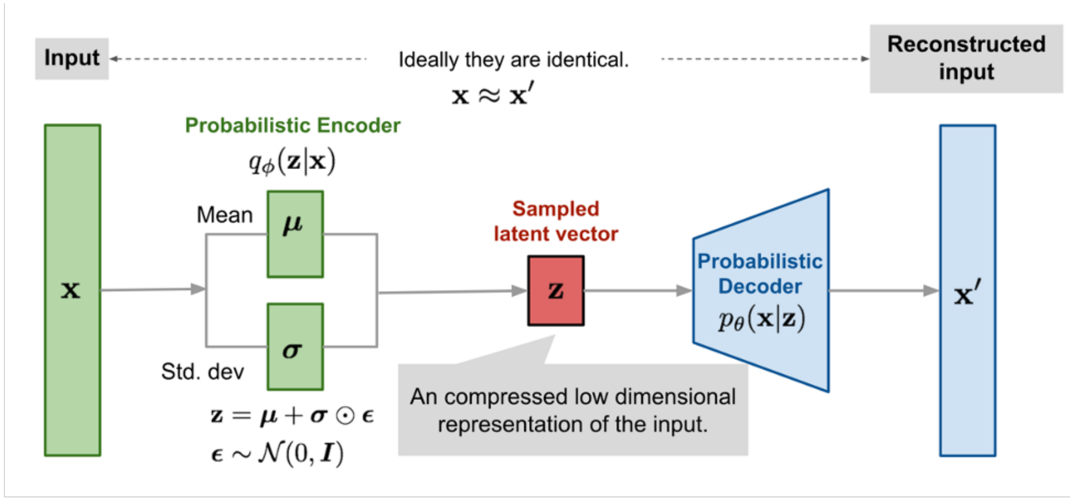
- End-to-end training



VAE vs. AE

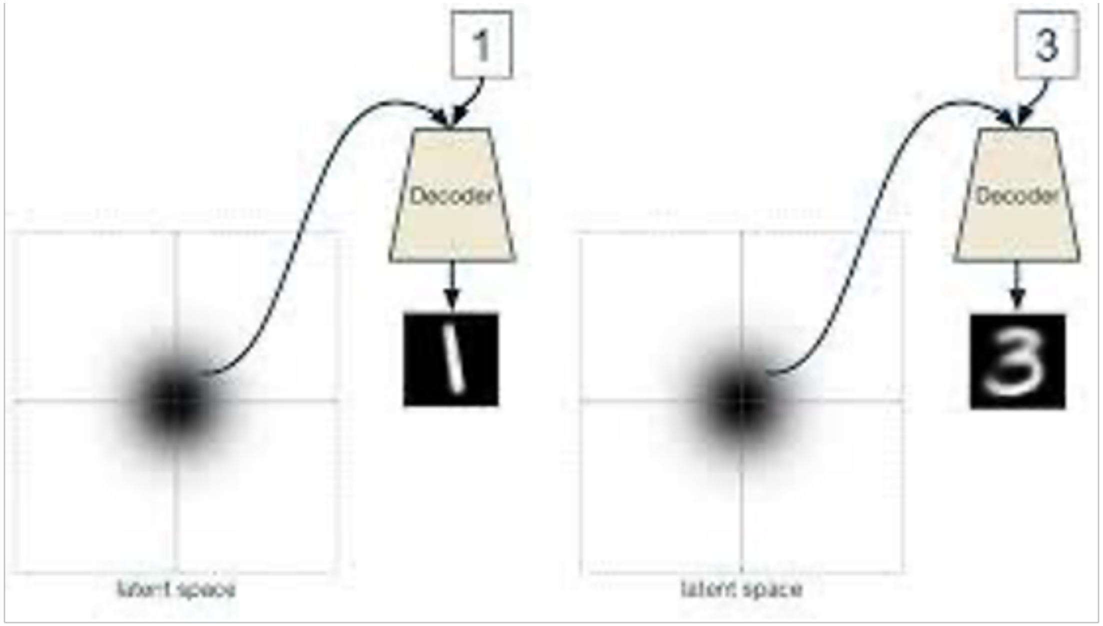
- AE: classical unsupervised representation learning method.
- VAR: a probabilistic model of AE
  - AE + Gaussian noise on z
  - KL penalty: L_2 constraint on the latent vector z





Conditioned VAE

- Semi-supervised learning: some labels are also available



conditioned generation

Comments on VAE

- Pros:
  - Flexible architecture
  - Stable training

- Cons:
  - Inaccurate probability evaluation (approximate inference)

$p(x)$