NLP course

Lecture 11: Knowledge Distillation and Generative Adversarial Networks for texts

Radoslav Neychev

MADE, Moscow

15.06.2022

Outline

1. Knowledge Distillation
2. Bonus: generative models overview
3. Outro

Based on: [Distilling the Knowledge in a Neural Network]

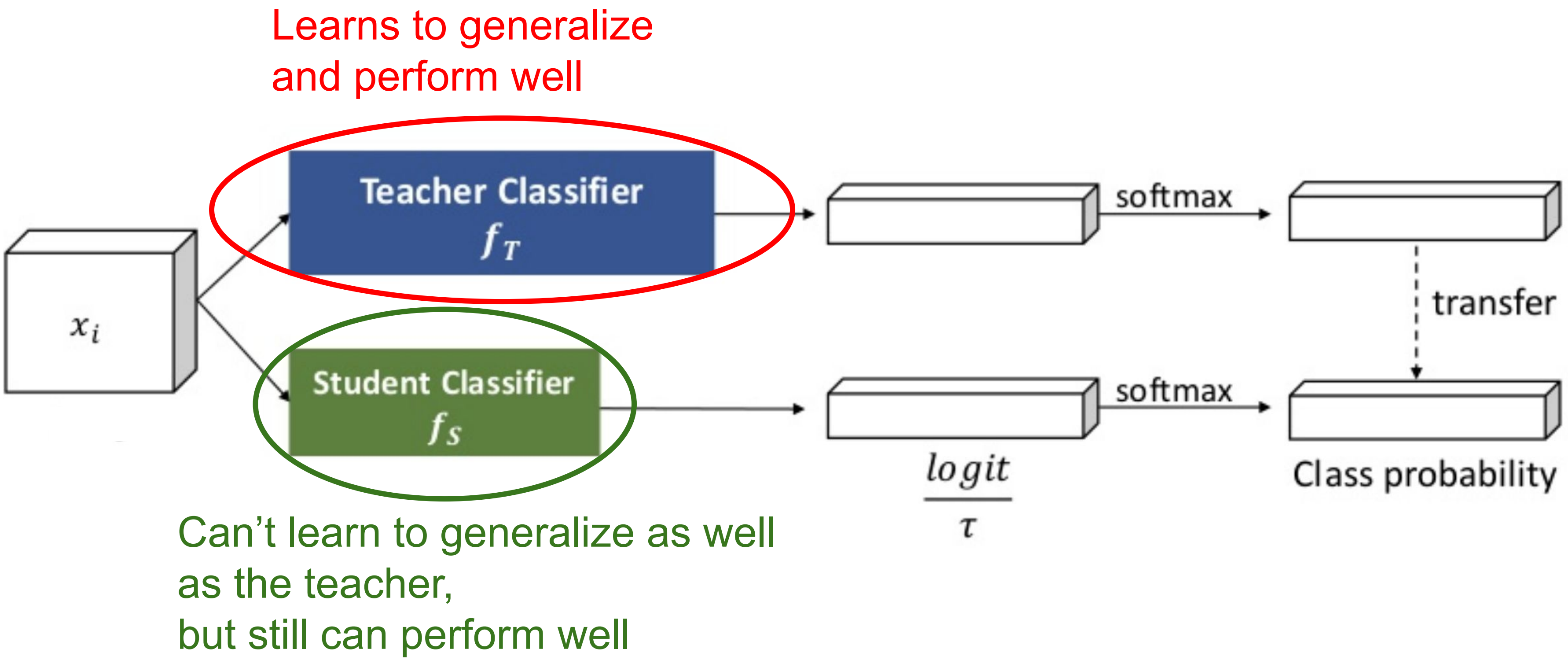
Knowledge Distillation

Cerura Vinula in caterpillar and butterfly forms

Do they have the same “life purpose”
and solve the same problems?

Image credits: commons.wikimedia.org (left), Ben Sale/Flickr.com (right), copied from
https://nat-geo.ru/nature/insects/bylo-stalo-kak-vyglyadyat-babochki-i-ikh-gusenitsy/

Knowledge distillation



Source: https://www.slideshare.net/NaverEngineering/relational-knowledge-distillation

Knowledge distillation

Denote **teacher** and **student** models.

Student model has logits z_i and corresponding probabilities q_i, derived with the softmax operation:

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

where *T* stays for the temperature.

Teacher model has logits v_i and corresponding probabilities p_i.

Source: [Distilling the Knowledge in a Neural Network]

Knowledge distillation

Let’s derive the cross-entropy gradient on **student** logits using the **teacher** predictions as targets:

$$\frac{\partial C}{\partial z_i} = \frac{1}{T}(q_i - p_i) = \frac{1}{T}\left(\frac{e^{z_i/T}}{\sum_j e^{z_j/T}} - \frac{e^{v_i/T}}{\sum_j e^{v_j/T}}\right)$$

If the temperature is high, the following equation takes place:

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{T}\left(\frac{1 + z_i/T}{N + \sum_j z_j/T} - \frac{1 + v_i/T}{N + \sum_j v_j/T}\right)$$

Source: [Distilling the Knowledge in a Neural Network]

Knowledge distillation

Logits can be centered, so

$$\sum_j z_j = \sum_j v_j = 0$$

Then the gradient takes form:

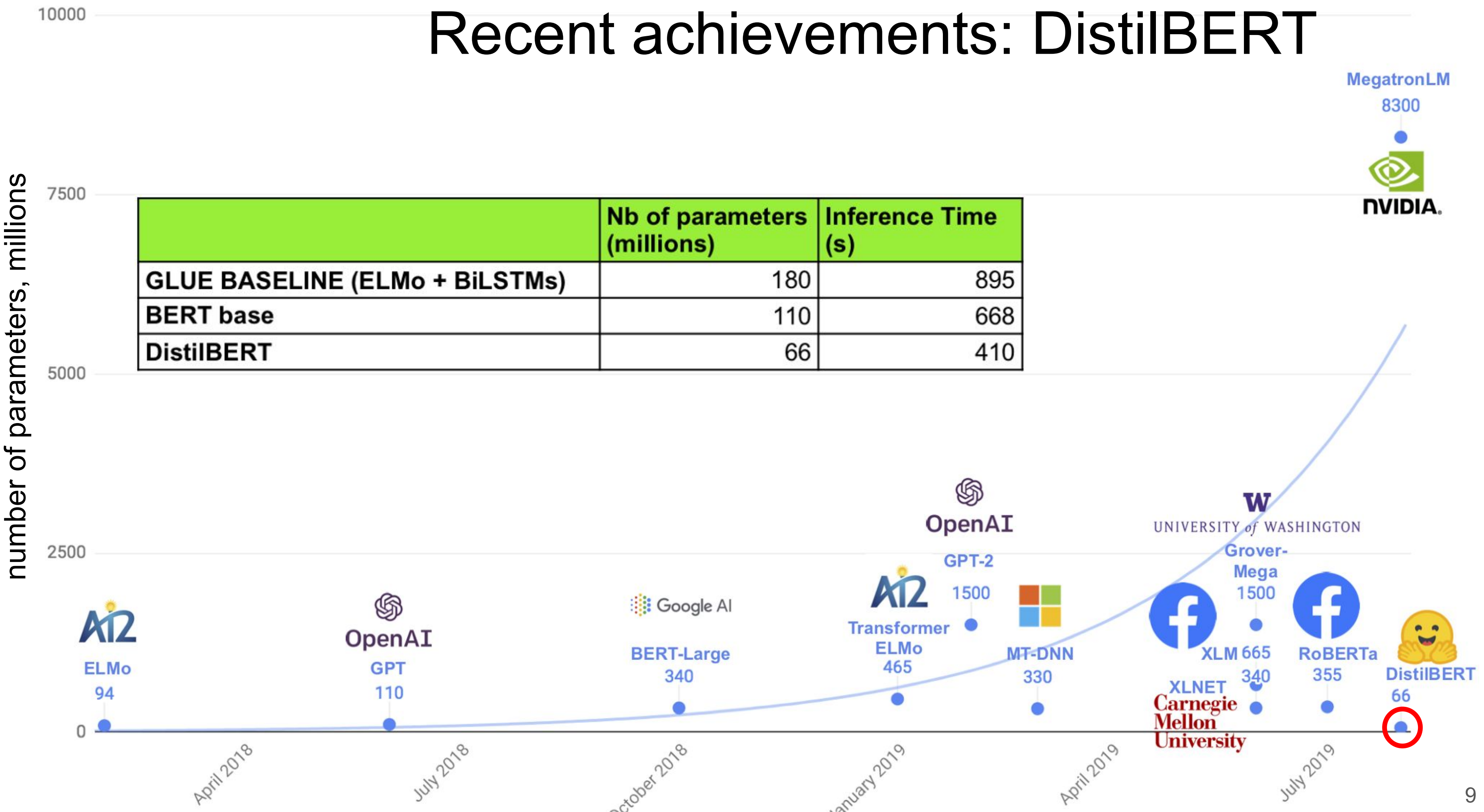
$$\frac{\partial C}{\partial z_i} \approx \frac{1}{T}\left(\frac{1 + z_i/T}{N + \sum_j z_j/T} - \frac{1 + v_i/T}{N + \sum_j v_j/T}\right) \approx \frac{1}{NT^2}\left(z_i - v_i\right)$$

Constant

$$\frac{dC}{dz_i} = \frac{1}{NT^2}(z_i - v_i) \sim (z_i - v_i) = M\frac{d(z_i - v_i)^2}{dz_i}$$

Source: [Distilling the Knowledge in a Neural Network]

Recent achievements: DistilBERT



	Nb of parameters (millions)	Inference Time (s)
GLUE BASELINE (ELMo + BiLSTMs)	180	895
BERT base	110	668
DistilBERT	66	410

Image source: [Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT]

Main ideas

- DistilBERT is initialized from its teacher, BERT, by taking one layer out of two, leveraging the common hidden size.
    - *Comment: Training a sub-network is not only about the architecture. It is also about finding the right initialization for the sub-network to converge.*

- DistilBERT is trained on very large batches leveraging gradient accumulation (up to 4000 examples per batch), with dynamic masking and removed the next sentence prediction objective.
    - *Comment: the way BERT is trained is crucial for its final performance.*

- DistilBERT was trained on eight 16GB V100 GPUs for approximately three and a half days using the concatenation of Toronto Book Corpus and English Wikipedia (same data as original BERT).

Image source: [Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT]

Generative Models

Generative models taxonomy

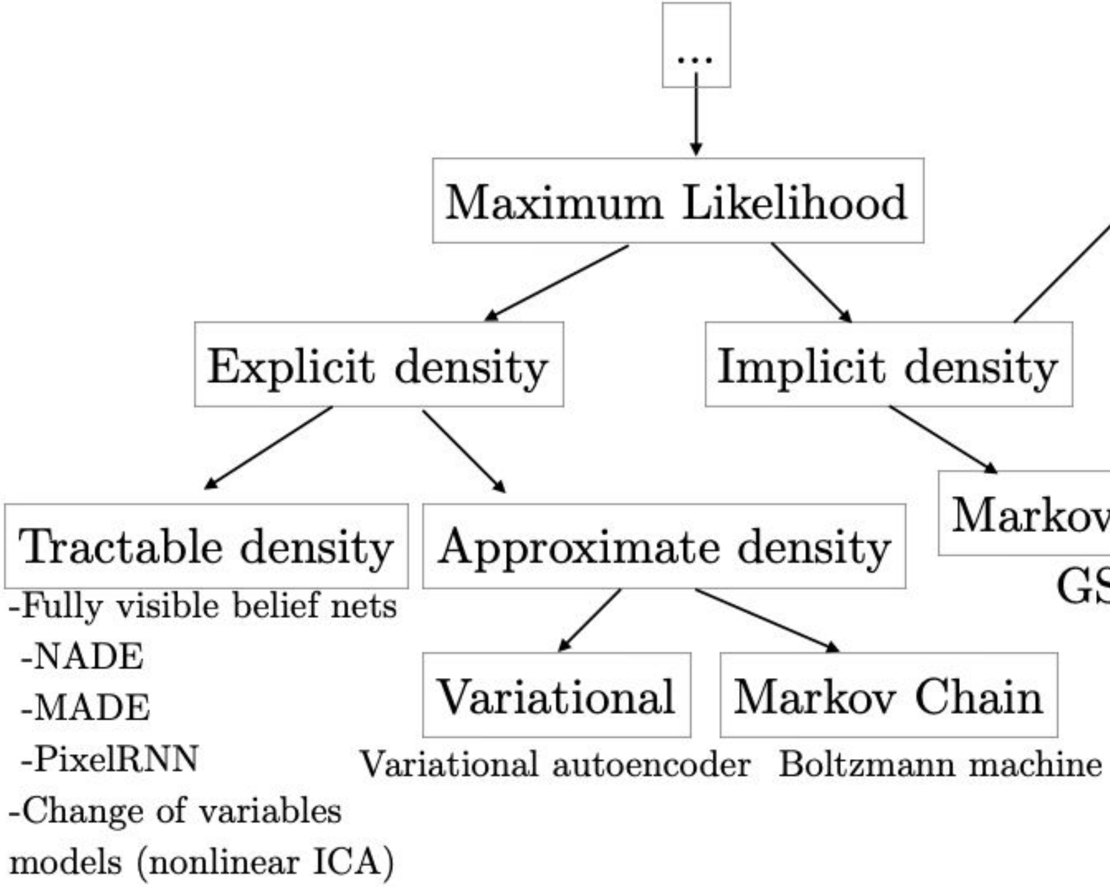


Image source: [Tutorial on Generative Adversarial Networks, Ian Goodfellow, NIPS, 2016]

Generative models taxonomy

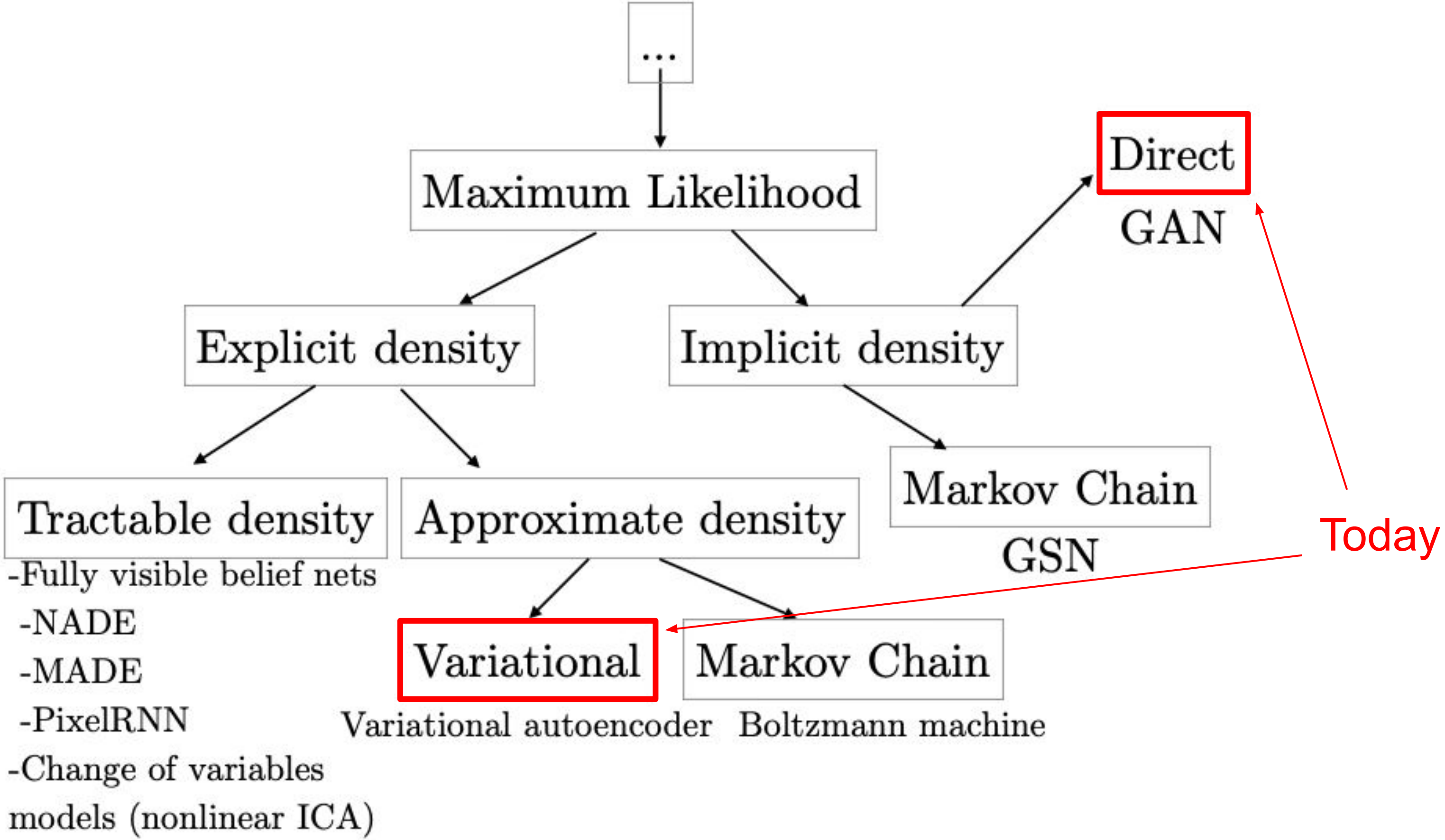


Image source: [Tutorial on Generative Adversarial Networks, Ian Goodfellow, NIPS, 2016]

Autoencoders

Denote **z** as encoded with encoder E input **x**

$$\mathbf{z} = E(\mathbf{x}, \boldsymbol{\theta}_E)$$

Decoder D recovers **x** from latent representation

$$\hat{\mathbf{x}} = D(\mathbf{z}, \boldsymbol{\theta}_D)$$

Optimal parameters learned w.r.t. loss function L

$$[\boldsymbol{\theta}_E, \boldsymbol{\theta}_D] = \arg\min L(\hat{\mathbf{x}}, \mathbf{x})$$

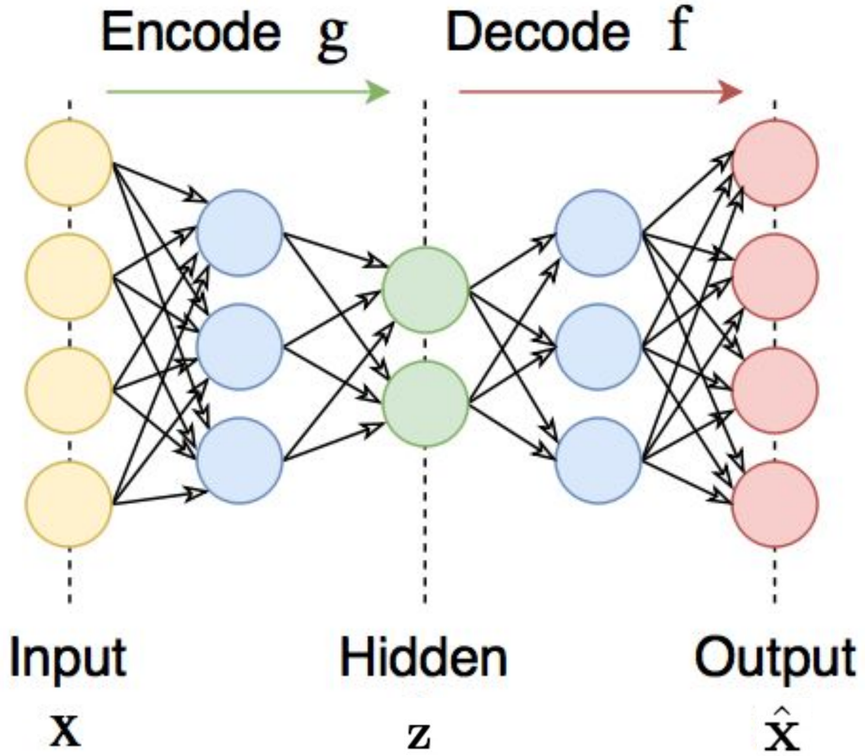


Image source: Habr post on autoencoders and GANs

Autoencoders

Denote **z** as encoded with encoder E input **x**

$$\mathbf{z} = E(\mathbf{x}, \boldsymbol{\theta}_E)$$

Decoder D recovers **x** from latent representation

$$\hat{\mathbf{x}} = D(\mathbf{z}, \boldsymbol{\theta}_D)$$

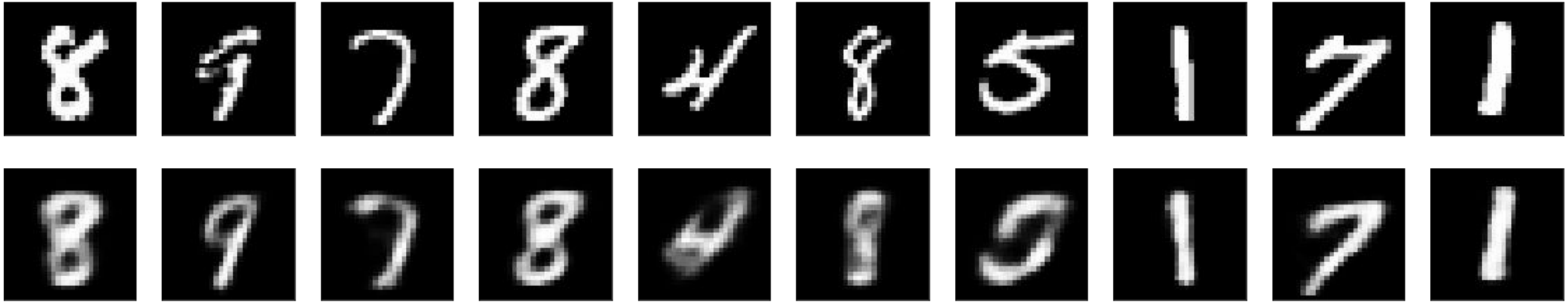
Simple example: PCA

Optimal parameters learned w.r.t. loss function L

$$[\boldsymbol{\theta}_E, \boldsymbol{\theta}_D] = \arg\min L(\hat{\mathbf{x}}, \mathbf{x})$$

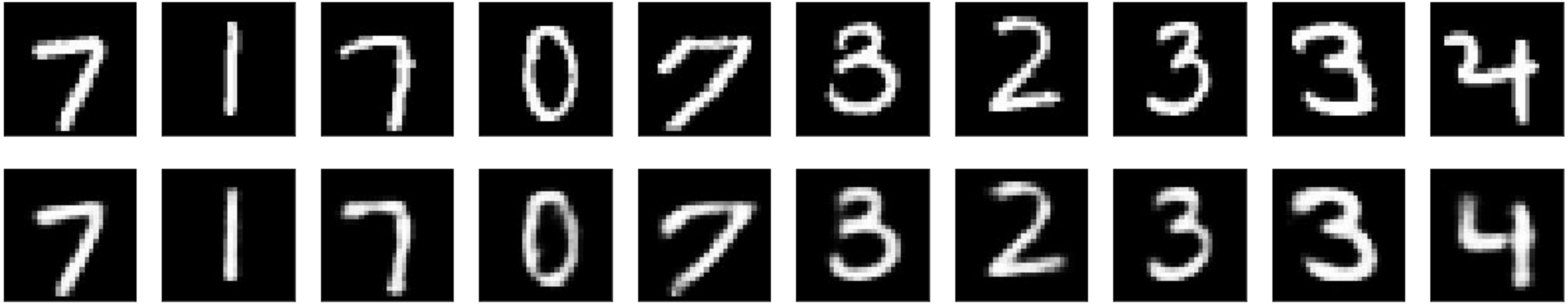
Image source: [Habr post on autoencoders and GANs]

PCA performance on MNIST



16 components

Convolutional performance on MNIST

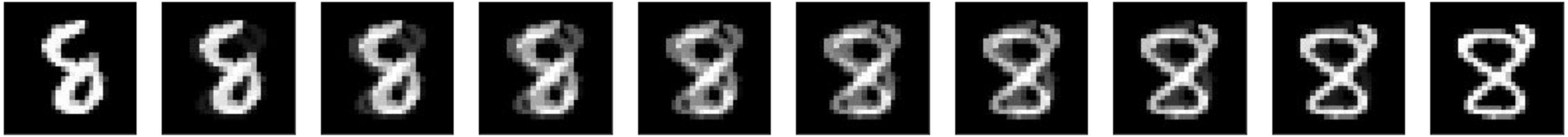


7 x 7 latent space

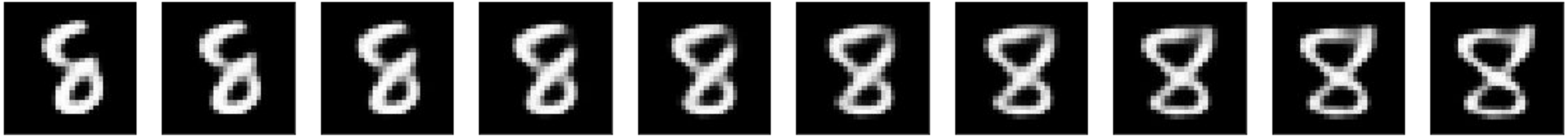
Homotopy between samples

10 steps between samples

- In original feature space (28 x 28):



- In latent space (7 x 7):



Latent space structure

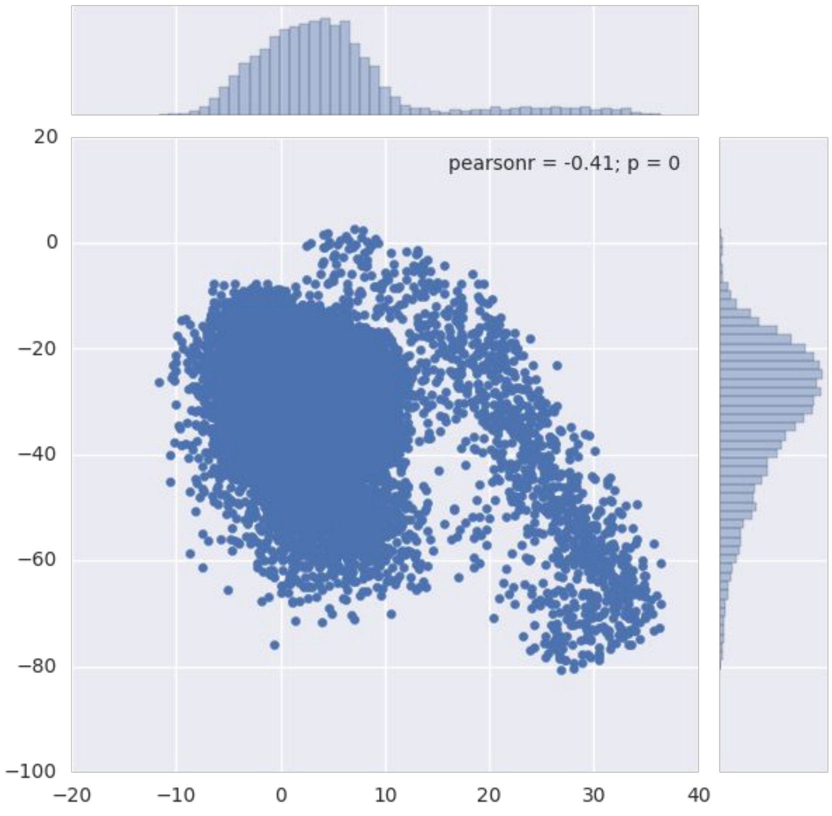


Image source: [Habr post on autoencoders and GANs]

VAE intuition

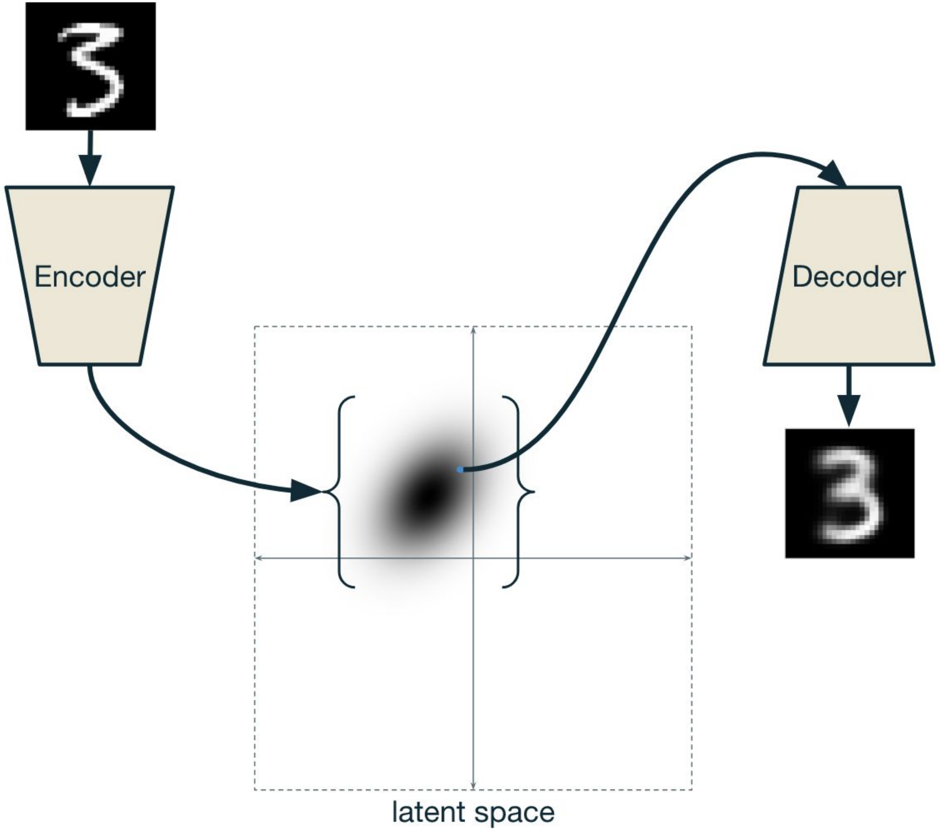


Image source: [Conditional Variational Autoencoders by Isaac Dykeman, 2016]

KL divergence

Denote distributions $Q(z)$ and $P(z|X)$.

Kullback–Leibler divergence is defined as

$$\mathcal{D}\left[Q(z)\|P(z|X)\right] = E_{z\sim Q}\left[\log Q(z) - \log P(z|X)\right]$$

KL divergence

$$\mathcal{D}\left[Q(z)\|P(z|X)\right] = E_{z\sim Q}\left[\log Q(z) - \log P(z|X)\right]$$

Applying the Bayes rule:

$$\mathcal{D}\left[Q(z)\|P(z|X)\right] = E_{z\sim Q}\left[\log Q(z) - \log P(X|z) - \log P(z)\right] + \log P(X)$$

KL divergence

$$\mathcal{D}\left[Q(z)\|P(z|X)\right] = E_{z\sim Q}\left[\log Q(z) - \log P(z|X)\right]$$

Applying the Bayes rule:

$$\mathcal{D}\left[Q(z)\|P(z|X)\right] = E_{z\sim Q}\left[\log Q(z) - \log P(X|z) - \log P(z)\right] + \log P(X)$$

$$\log P(X) - \mathcal{D}\left[Q(z)\|P(z|X)\right] = E_{z\sim Q}\left[\log P(X|z)\right] - \mathcal{D}\left[Q(z)\|P(z)\right]$$

KL divergence

$$\mathcal{D}\left[Q(z)\|P(z|X)\right] = E_{z\sim Q}\left[\log Q(z) - \log P(z|X)\right]$$

Applying the Bayes rule:

$$\mathcal{D}\left[Q(z)\|P(z|X)\right] = E_{z\sim Q}\left[\log Q(z) - \log P(X|z) - \log P(z)\right] + \log P(X)$$

$$\log P(X) - \mathcal{D}\left[Q(z)\|P(z|X)\right] = E_{z\sim Q}\left[\log P(X|z)\right] - \mathcal{D}\left[Q(z)\|P(z)\right]$$

$$\boxed{\log P(X) - \mathcal{D}\left[Q(z|X)\|P(z|X)\right] = E_{z\sim Q}\left[\log P(X|z)\right] - \mathcal{D}\left[Q(z|X)\|P(z)\right]}$$

This equation is the core of Variational Autoencoders

Structure of the latent space

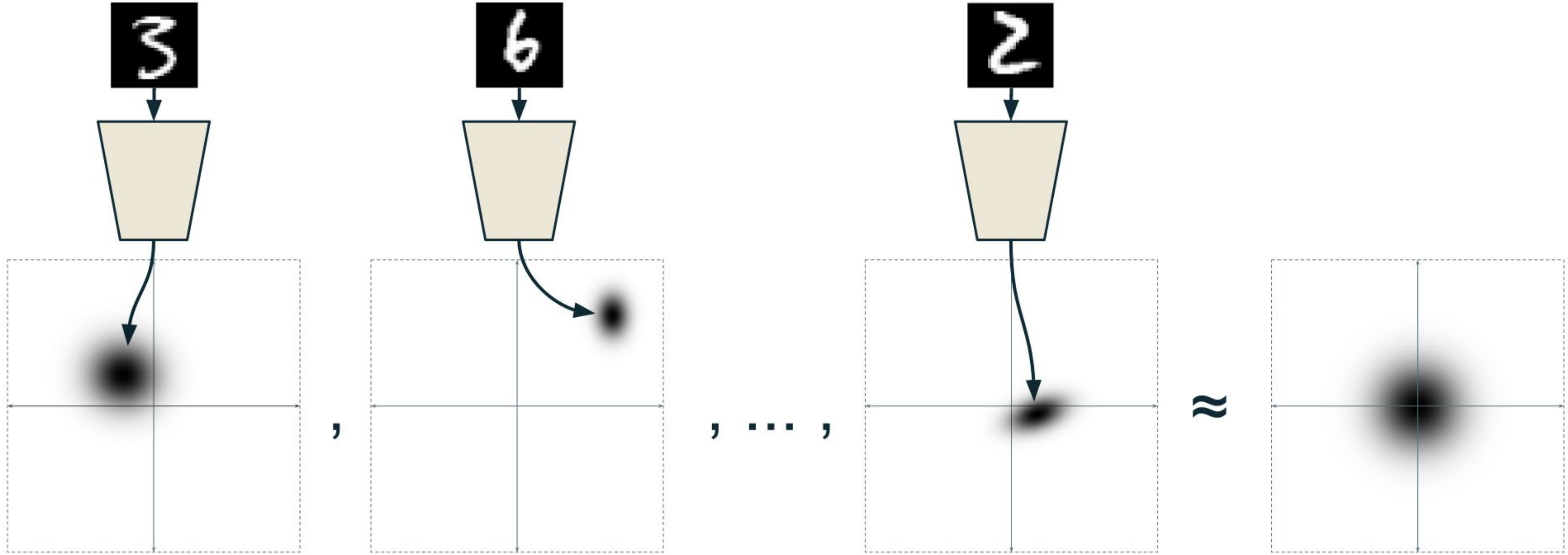


Image source: [Conditional Variational Autoencoders by Isaac Dykeman, 2016]

VAE latent space distribution

VAE so far

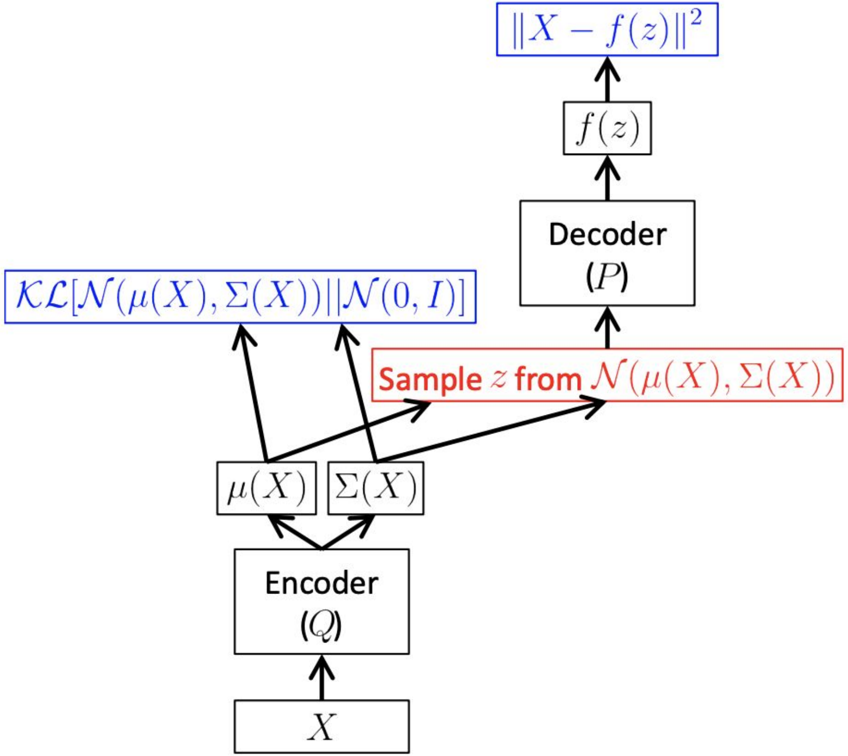


Image source: [Tutorial on Variational Autoencoders by Carl Doersch, 2016]

VAE so far

$$\mathcal{D}[\mathcal{N}(\mu(X),\Sigma(X))\|\mathcal{N}(0,I)] = \frac{1}{2}\left(\mathrm{tr}\left(\Sigma(X)\right) + \left(\mu(X)\right)^\top\left(\mu(X)\right) - k - \log\det\left(\Sigma(X)\right)\right)$$

Try to derive it by yourself

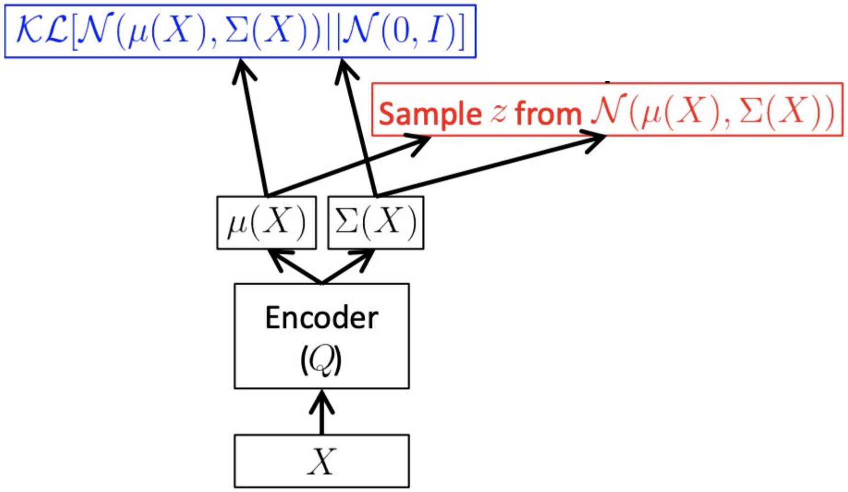
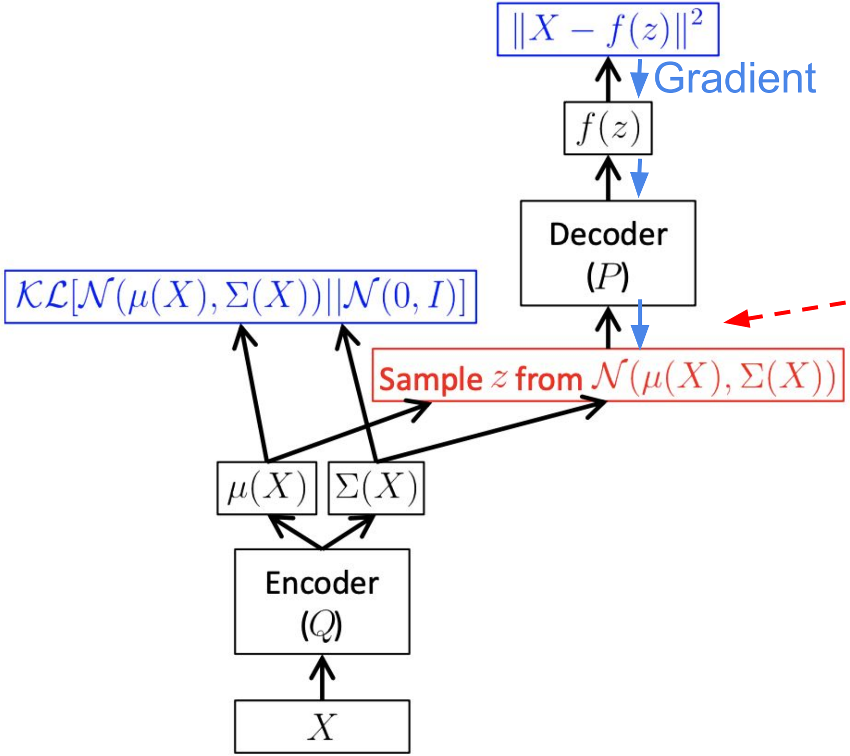


Image source: [Tutorial on Variational Autoencoders by Carl Doersch, 2016]

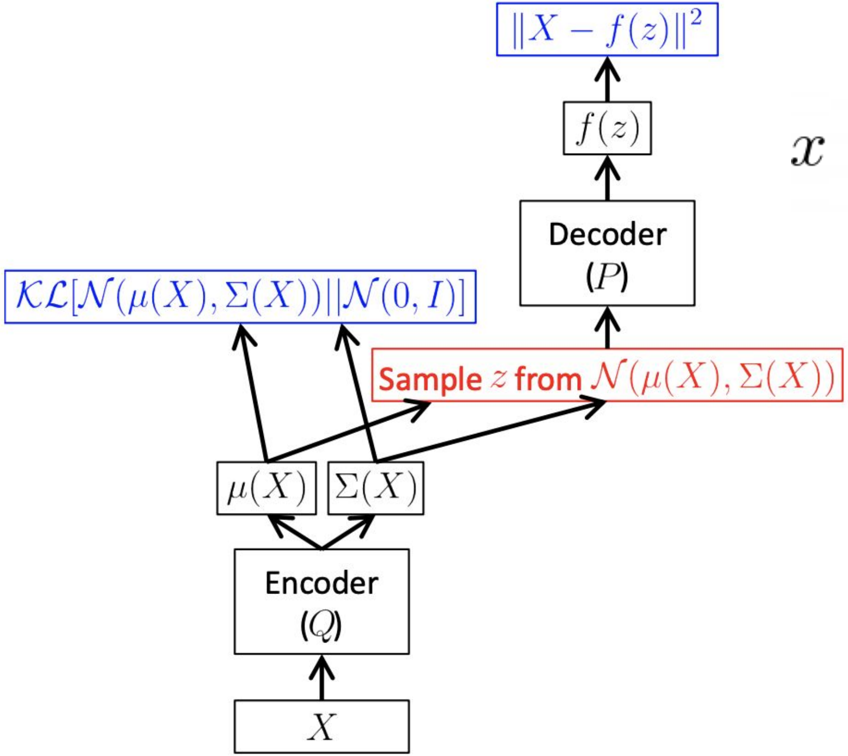
VAE so far



Gradient flow is broken!

Image source: [Tutorial on Variational Autoencoders by Carl Doersch, 2016]

Reparametrization trick



$$x \sim \mathcal{N}(\mu, \sigma^2), \quad z = \frac{x - \mu}{\sigma} \sim \mathcal{N}(0, 1)$$

$$\Rightarrow x = \sigma(z + \mu) \sim \mathcal{N}(\mu, \sigma^2)$$

Image source: Tutorial on Variational Autoencoders by Carl Doersch, 2016

Reparametrization trick

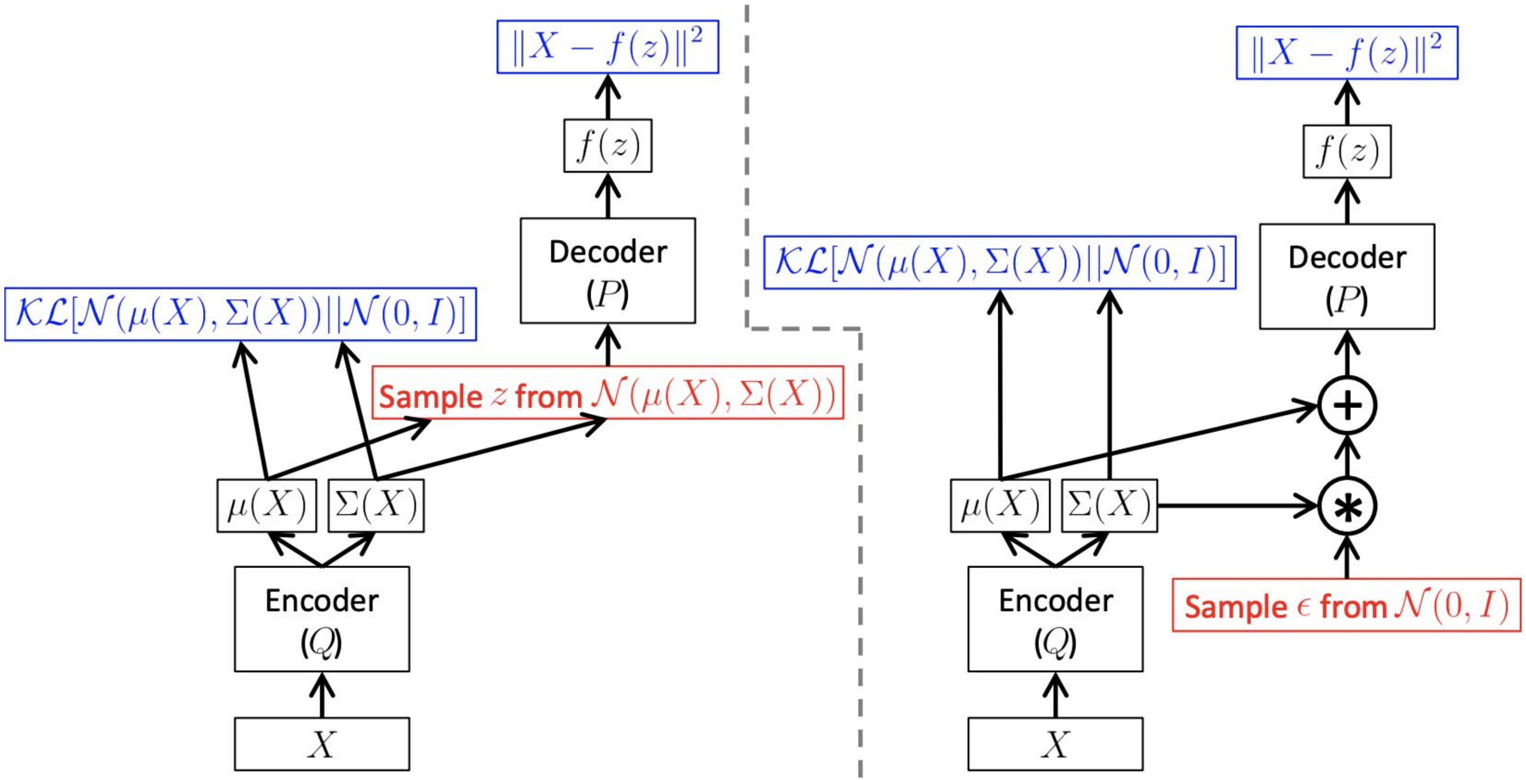
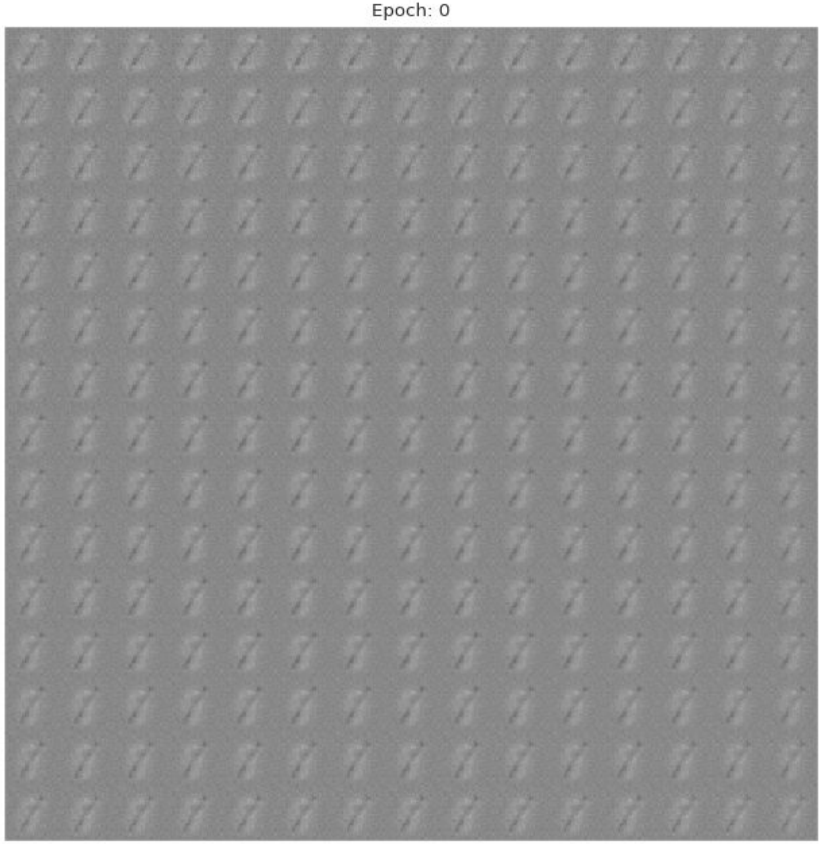


Image source: [Tutorial on Variational Autoencoders by Carl Doersch, 2016]

VAE manifold



Structure of the latent space

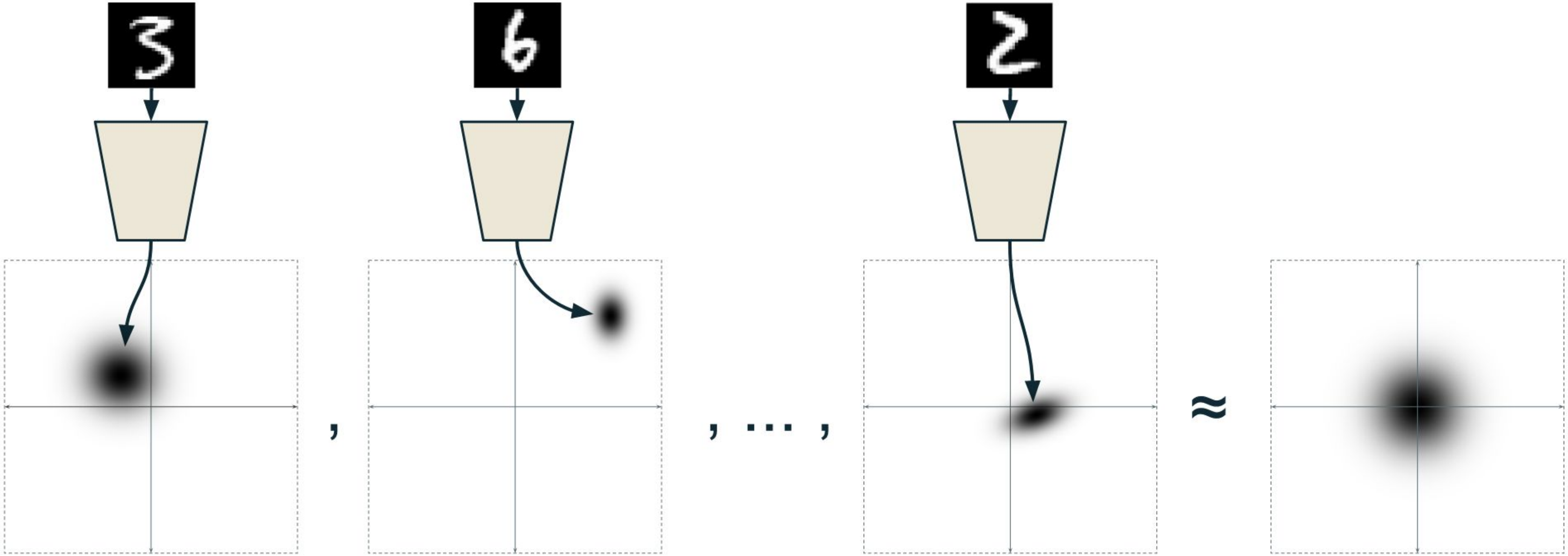


Image source: [Conditional Variational Autoencoders by Isaac Dykeman, 2016]

Conditional VAE intuition

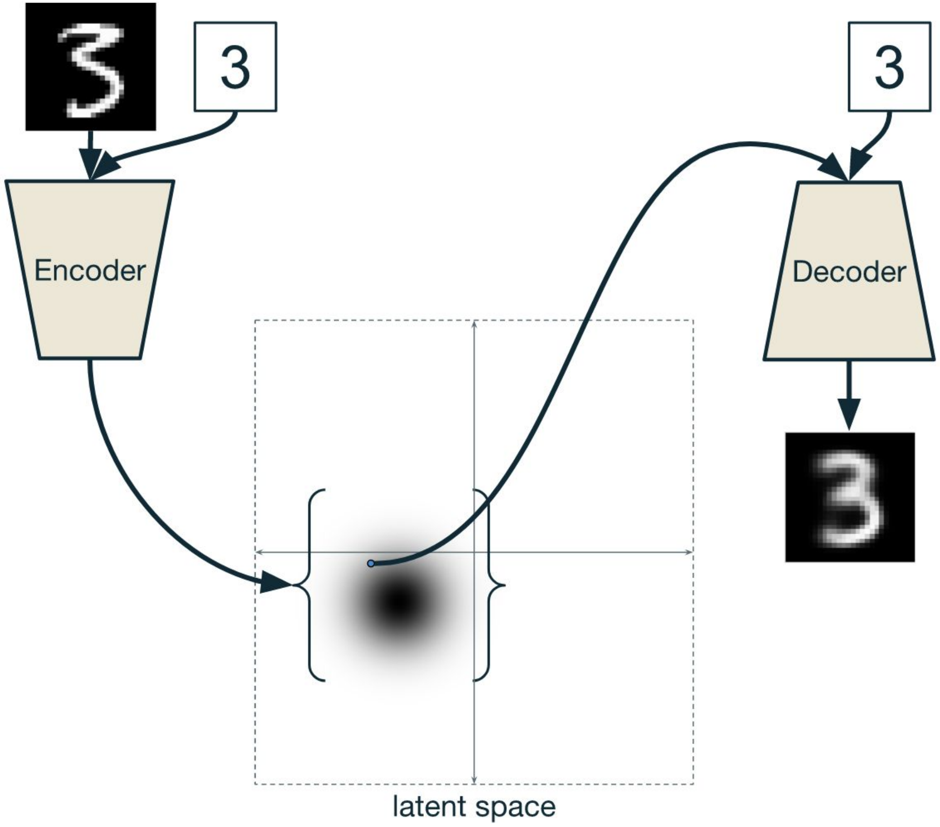


Image source: [Conditional Variational Autoencoders by Isaac Dykeman, 2016]

Conditional VAE

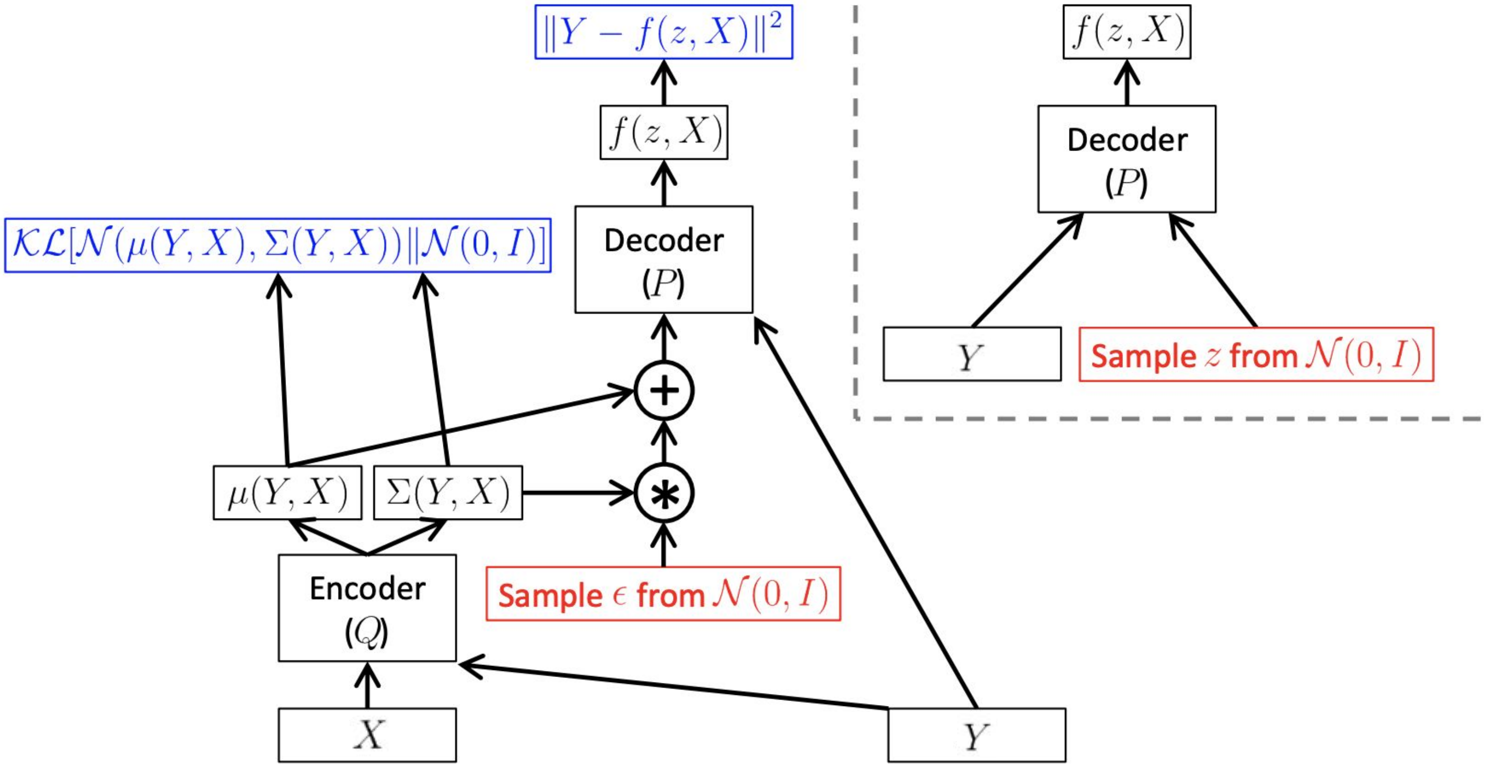
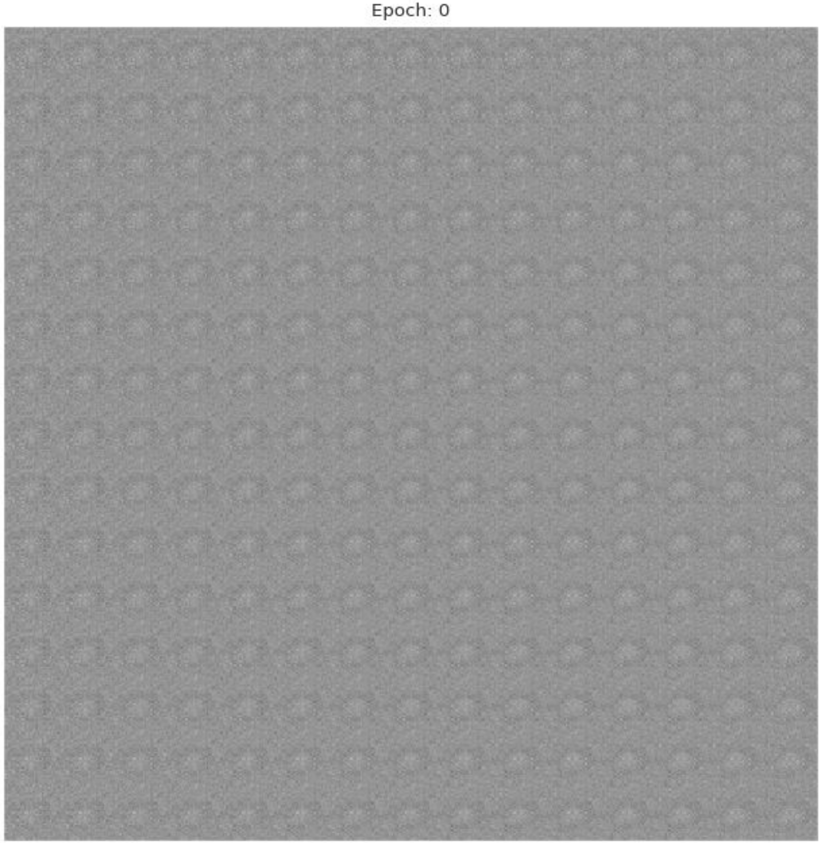
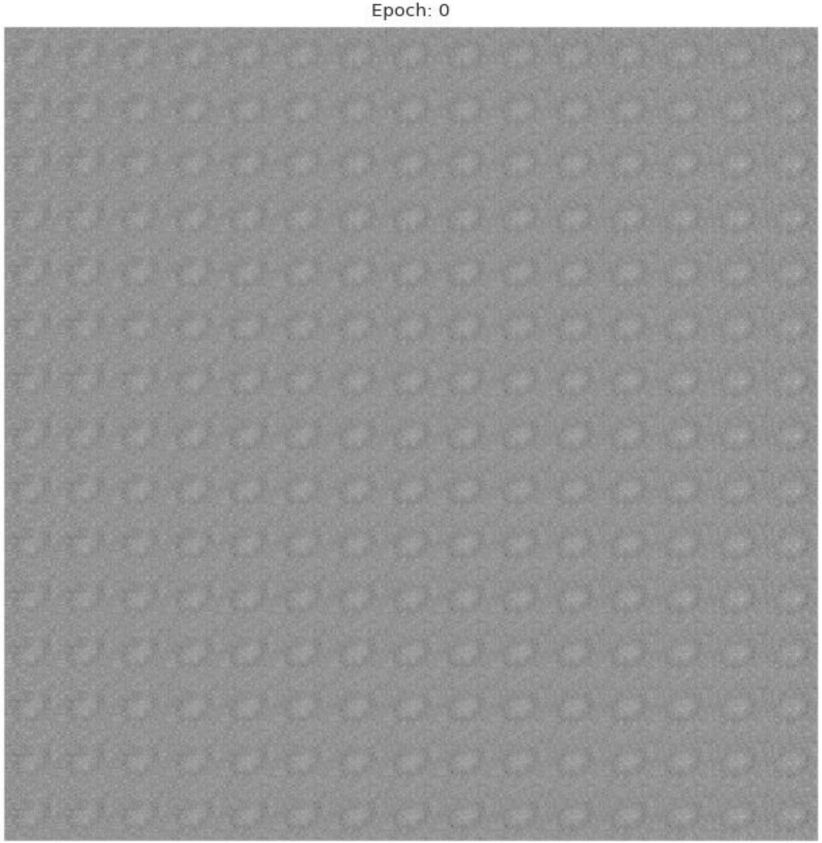


Image source: [Tutorial on Variational Autoencoders by Carl Doersch, 2016]

cVAE manifold



cVAE manifold



Transferring style with cVAE

Seed label is 1

Once again

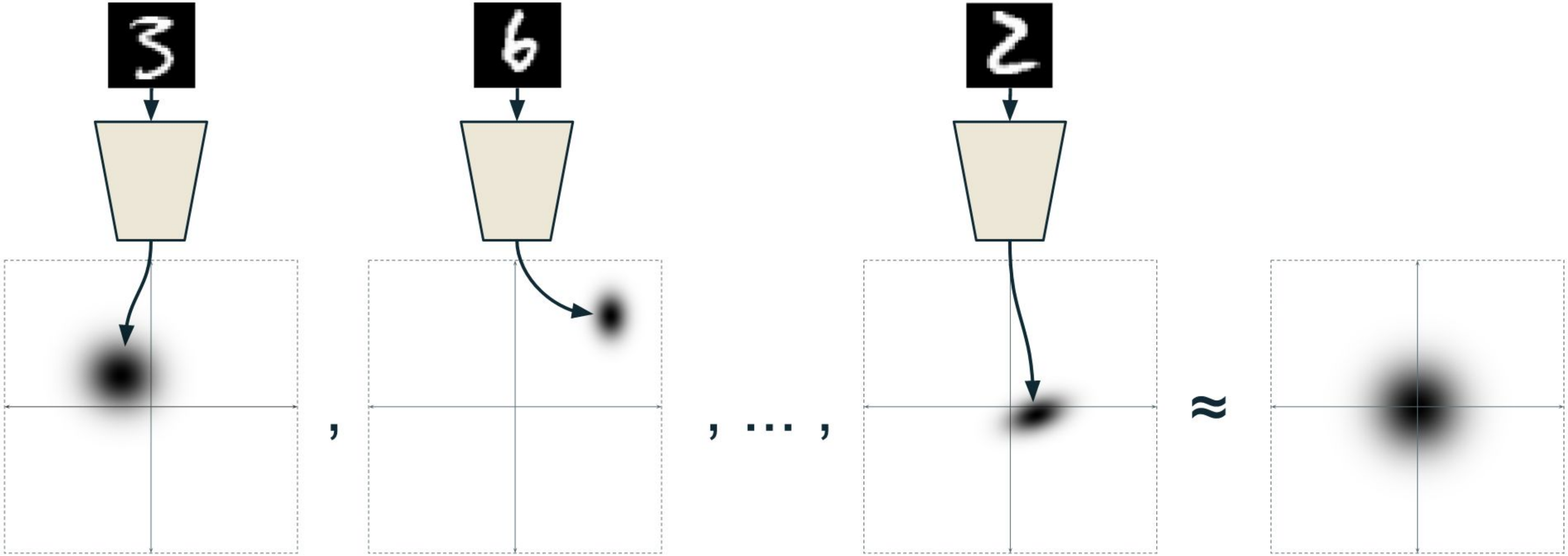
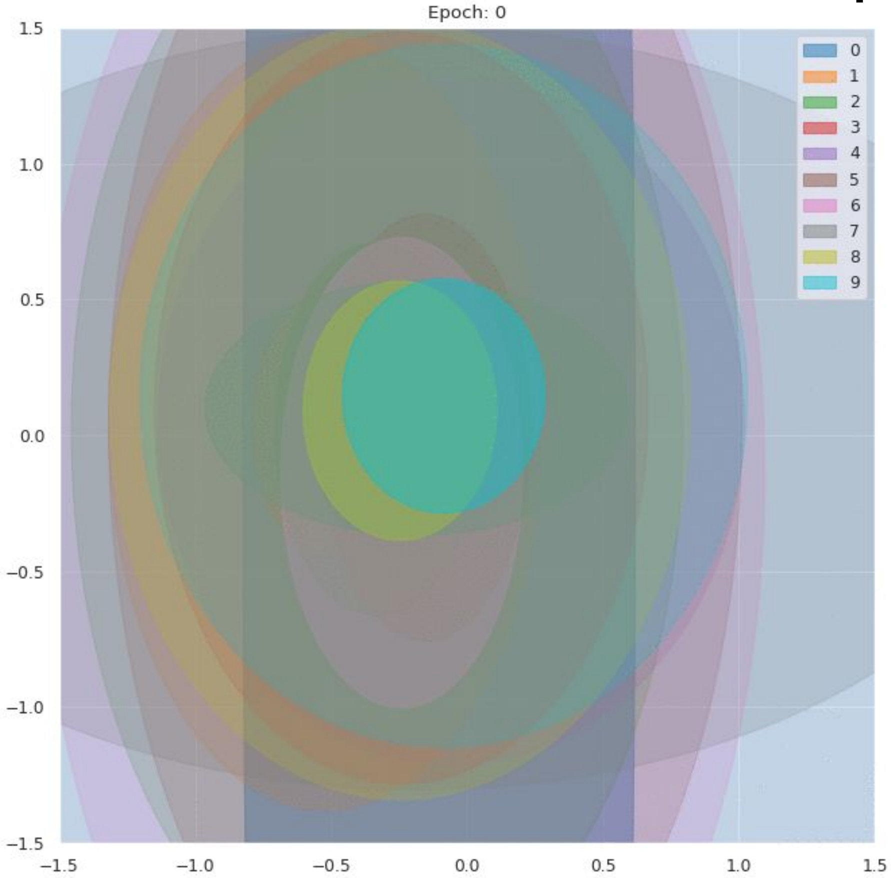


Image source: [Conditional Variational Autoencoders by Isaac Dykeman, 2016]

cVAE latent space distribution



GAN

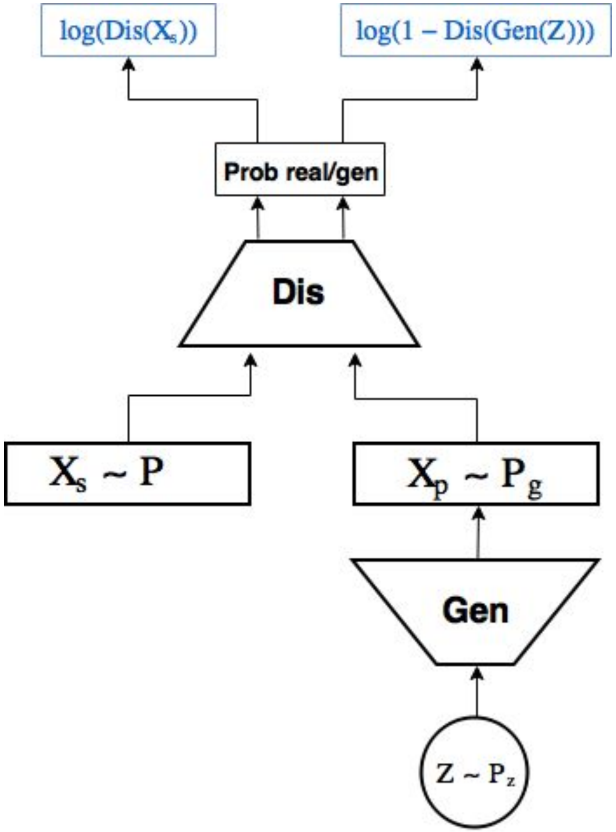
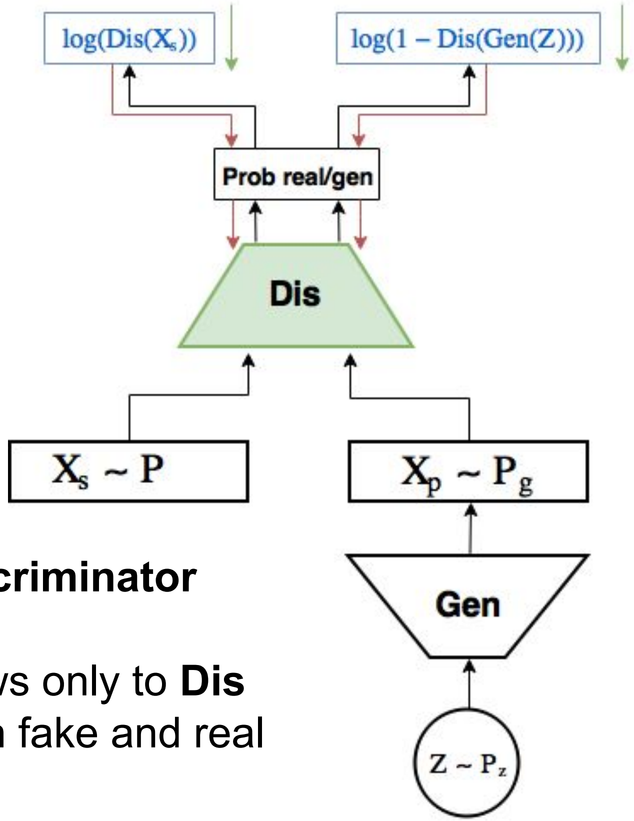
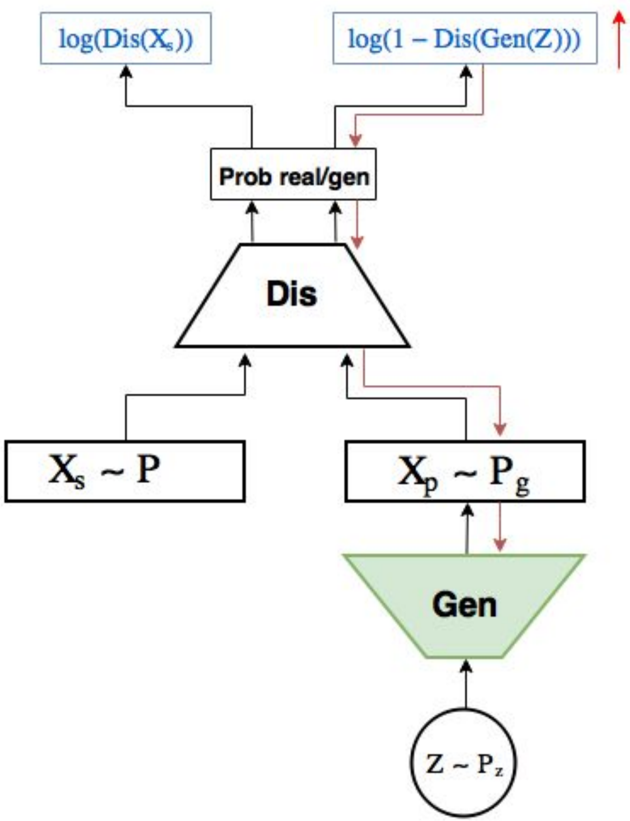


Image source: [Habr post on autoencoders and GANs]

Training GAN



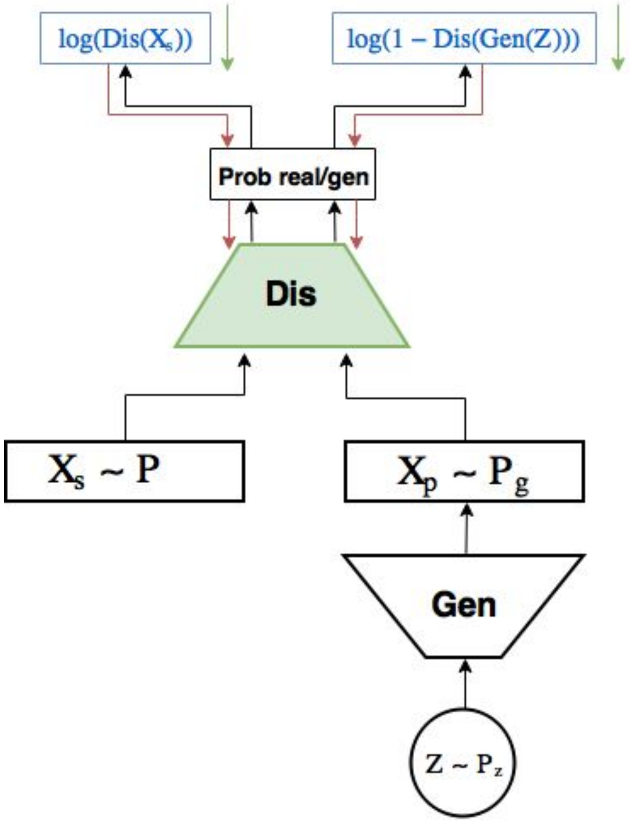


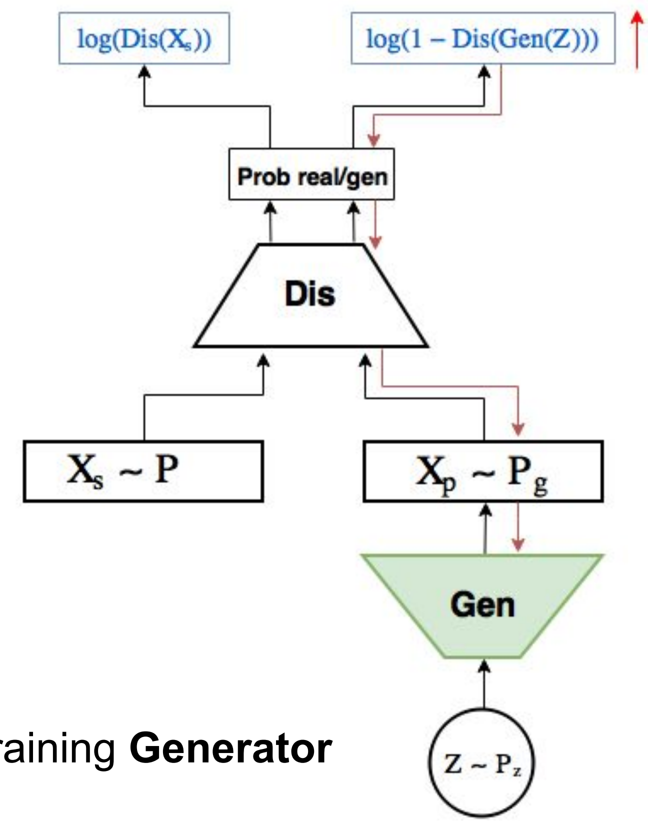
Training **Discriminator**

Gradient flows only to **Dis** to distinguish fake and real examples

Image source: [Habr post on autoencoders and GANs]

Training GAN





Training **Generator**

Gradient flows to **Gen** with **Dis** weights freezed to fool the Discriminator

Image source: [Habr post on autoencoders and GANs]

Optimization process in GAN

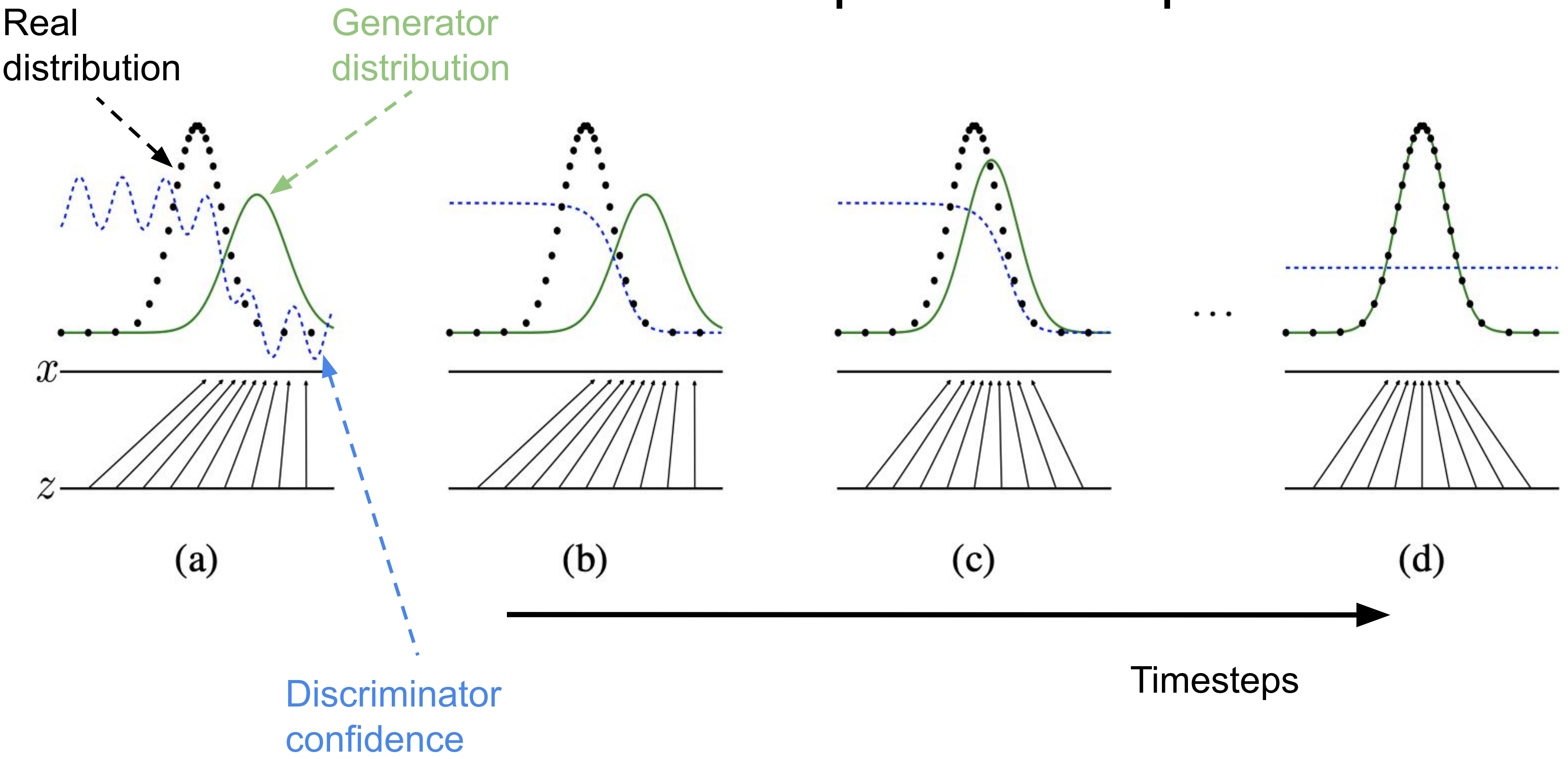


Image source: [Generative Adversarial Nets, Ian J. Goodfellow et al, 2014]

GAN manifold

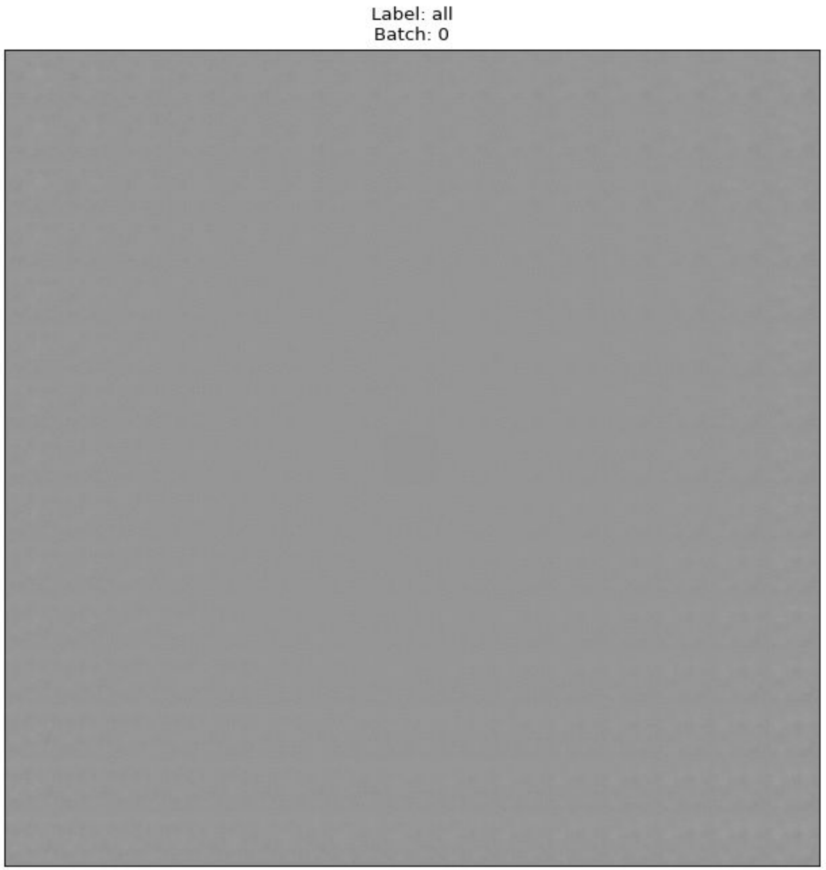


Image source: [Habr post on autoencoders and GANs]

Conditional GAN

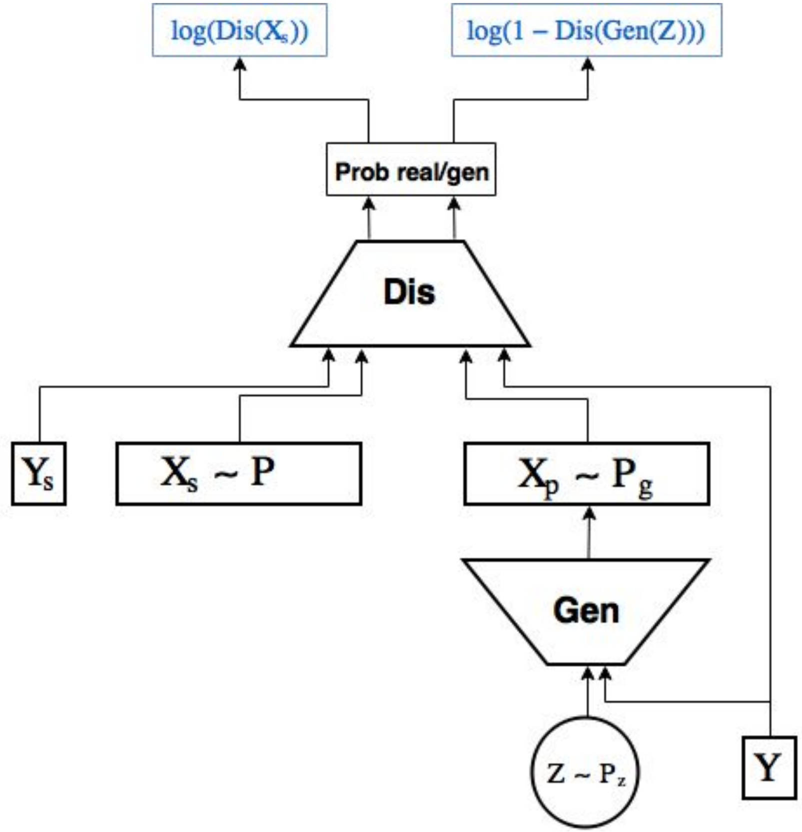
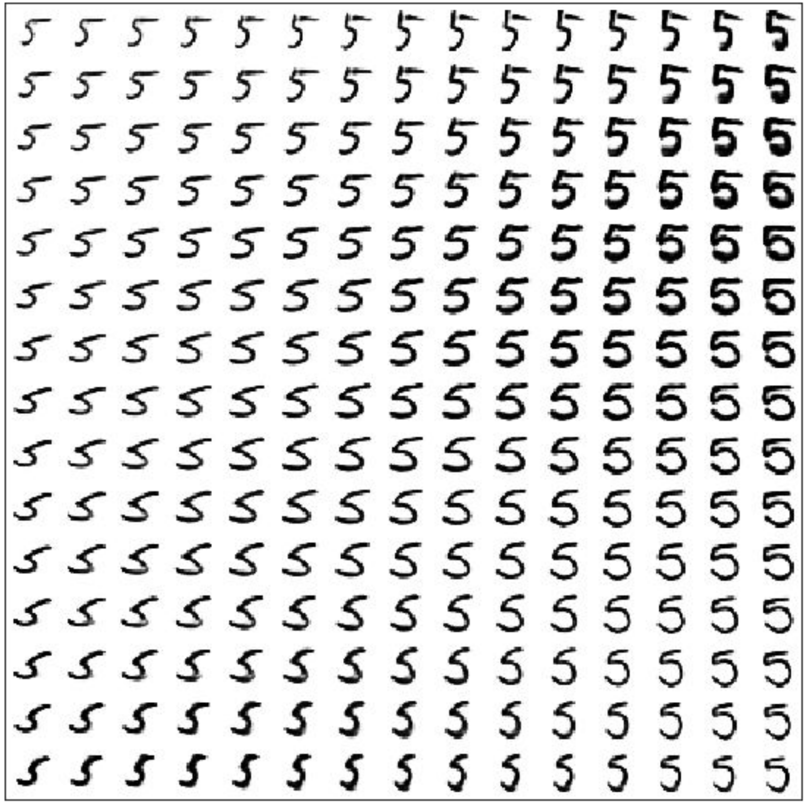


Image source: Habr post on autoencoders and GANs

cGAN manifolds



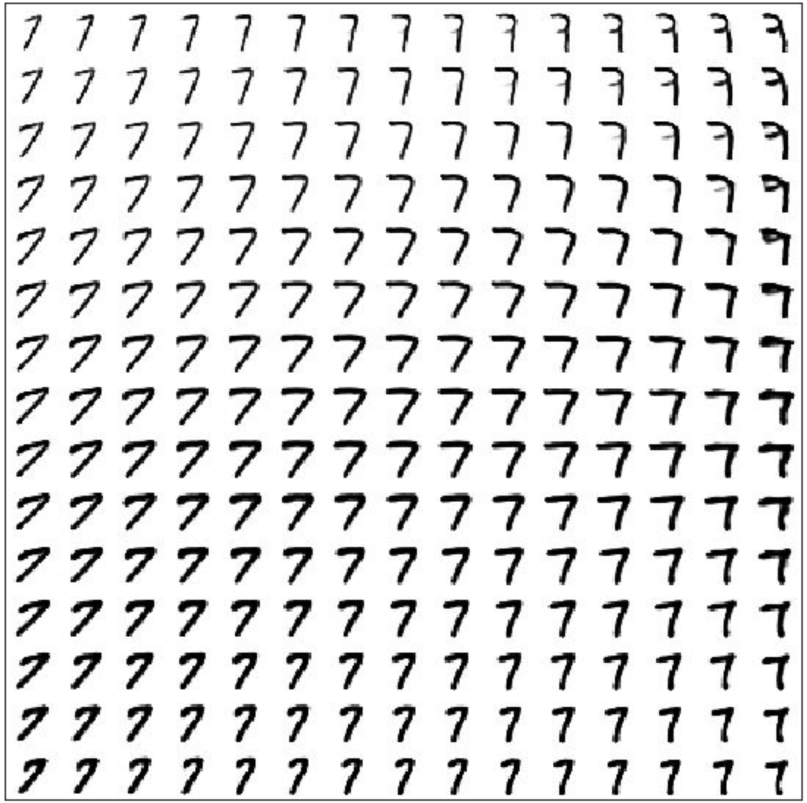


Image source: [Habr post on autoencoders and GANs]

Some more combinations

Image source: [Guiness World Records blog](), original video [PIKOTARO - PPAP (Pen Pineapple Apple Pen)]()

Simple GAN

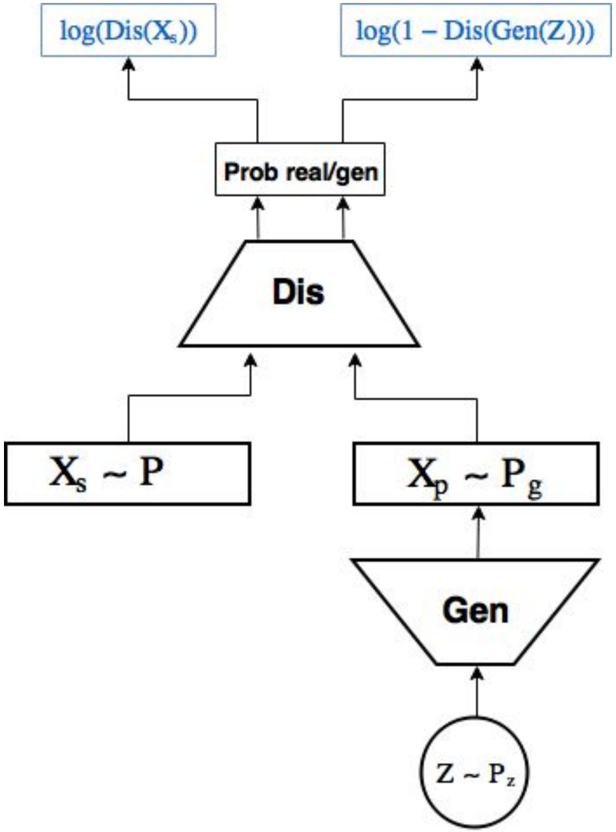
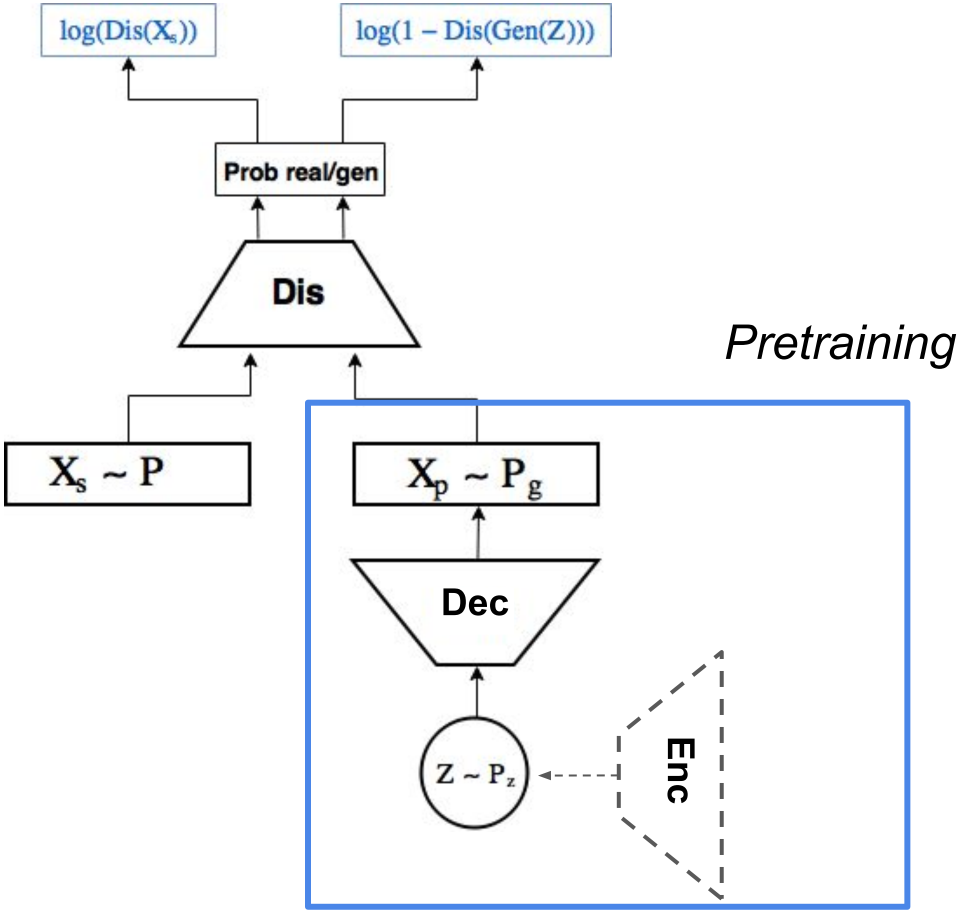


Image source: [Habr post on autoencoders and GANs]()

VAE/GAN



Partial image source: [Habr post on autoencoders and GANs]()

VAE/GAN original illustration

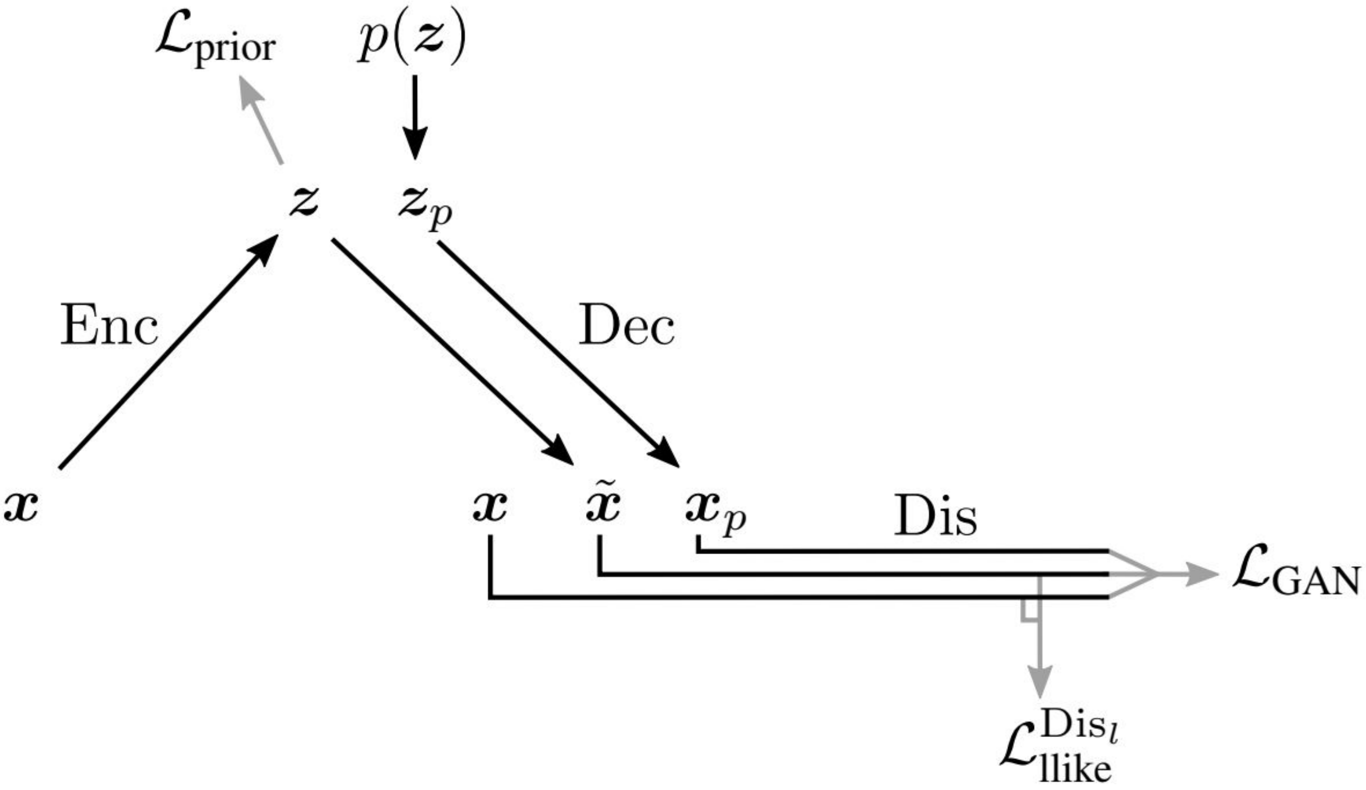
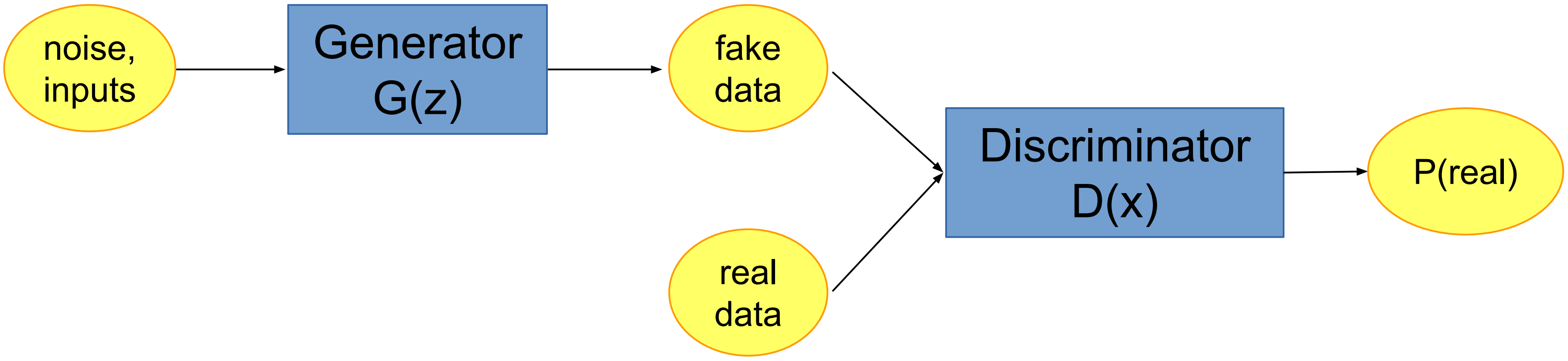


Image source: [Autoencoding beyond pixels using a learned similarity metric, Anders Boesen Lindbo Larsen et al, 2016]

Bonus: discrete GANs

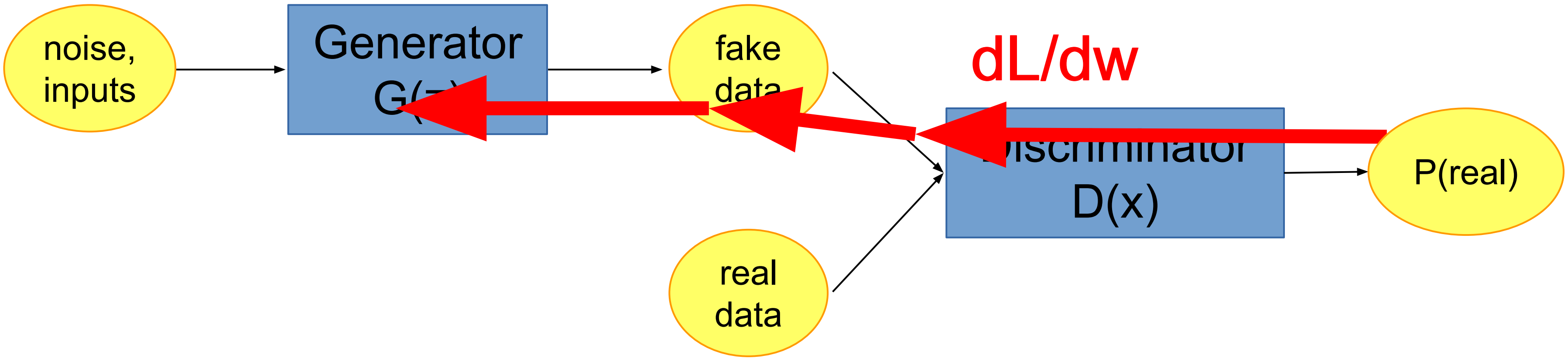
Generalized GAN scheme



source: [Practical RL week07]

Bonus: discrete GANs

Generalized GAN scheme

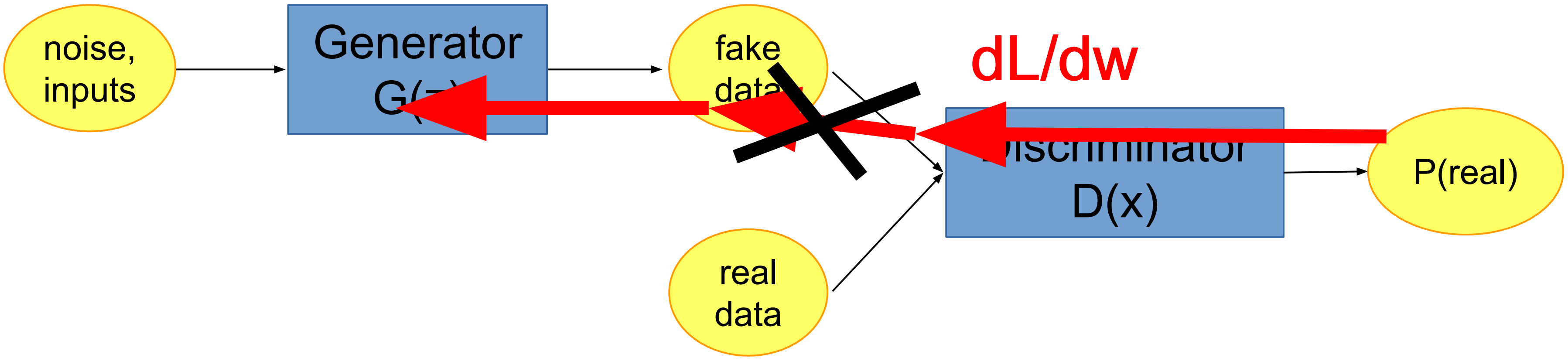


source: [Practical RL week07]

Bonus: discrete GANs

Standard scheme fails if G(z) is discrete

- generating text
- generating music notes
- generating molecules
- binary image masks



source: [Practical RL week07]

Bonus: discrete GANs

We can train generator with Reinforcement Learning methods!

$$\nabla J = \underset{\substack{z \sim p(z) \\ x \sim P(x|G_\theta(z))}}{E} \nabla \log P(x|G_\theta(z)) D(x)$$

source: [Practical RL week07]