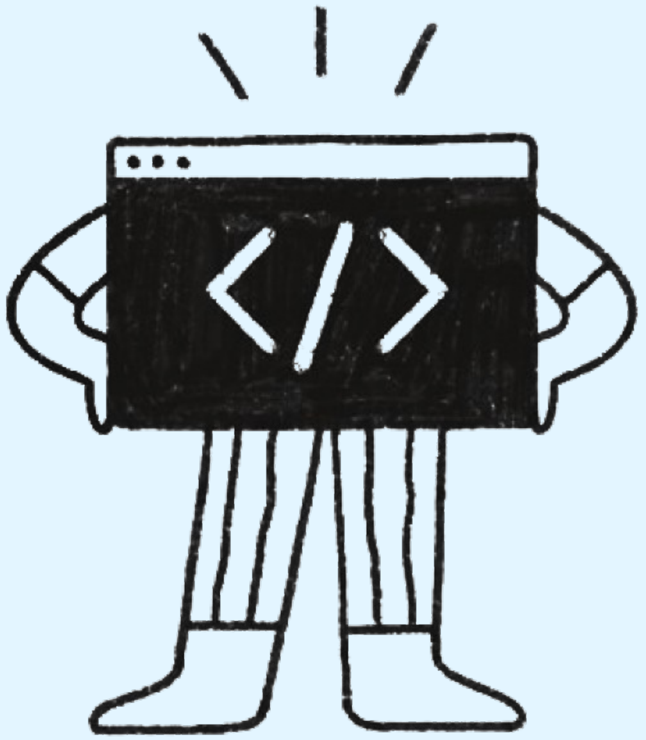
DETR outro,
VAE,
GAN

Радослав Нейчев

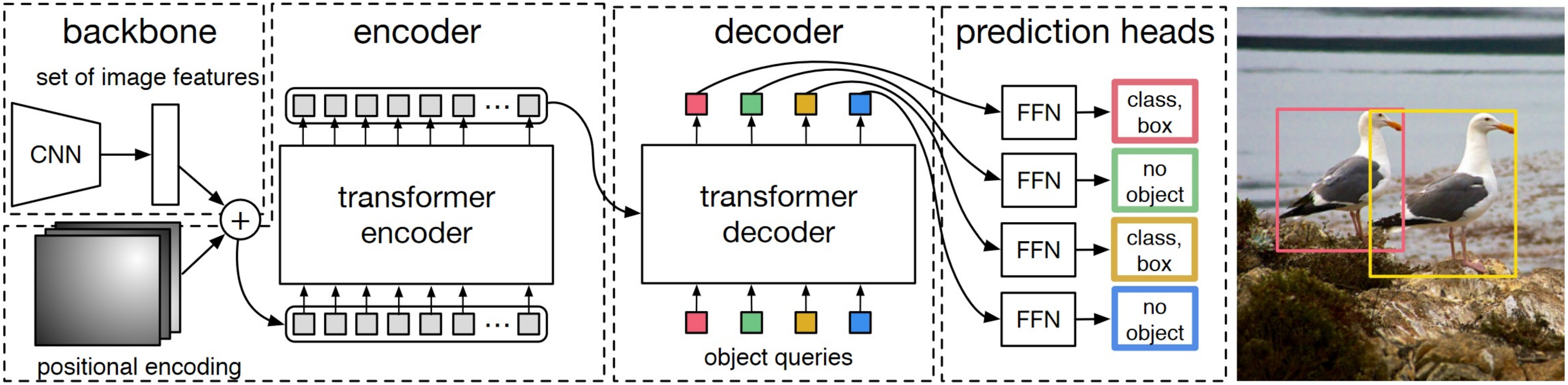
Выпускник и преподаватель ШАД и МФТИ,
руководитель группы ML-разработки в Яндексе,
основатель girafe ai

How DETR decoder works



01

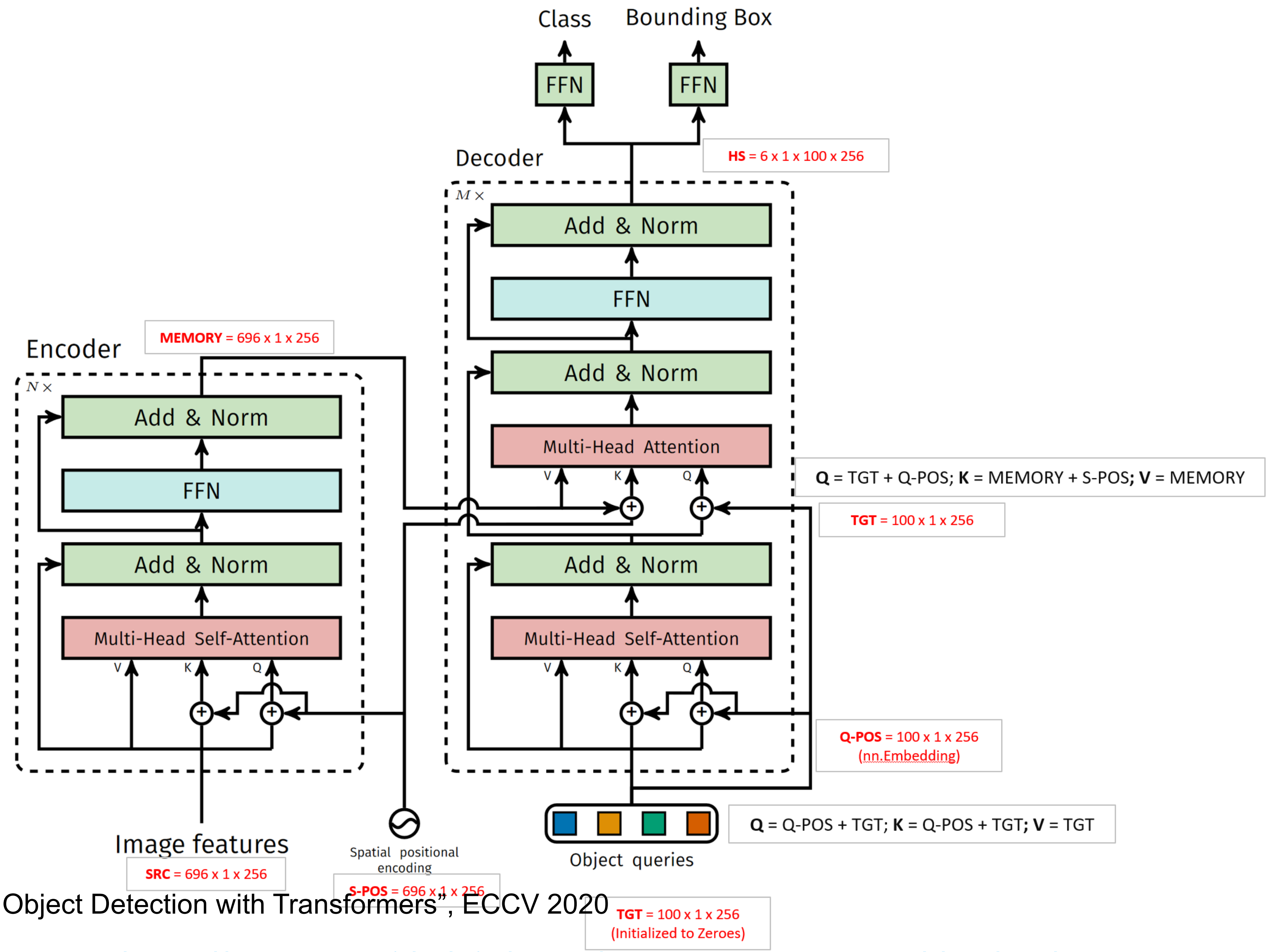
DETR



Carion et al, “End-to-End Object Detection with Transformers”, ECCV 2020

Image source: https://amaarora.github.io/posts/2021-07-26-annotateddetr.html

DETR



Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

Image source: https://amaarora.github.io/posts/2021-07-26-annotateddetr.html

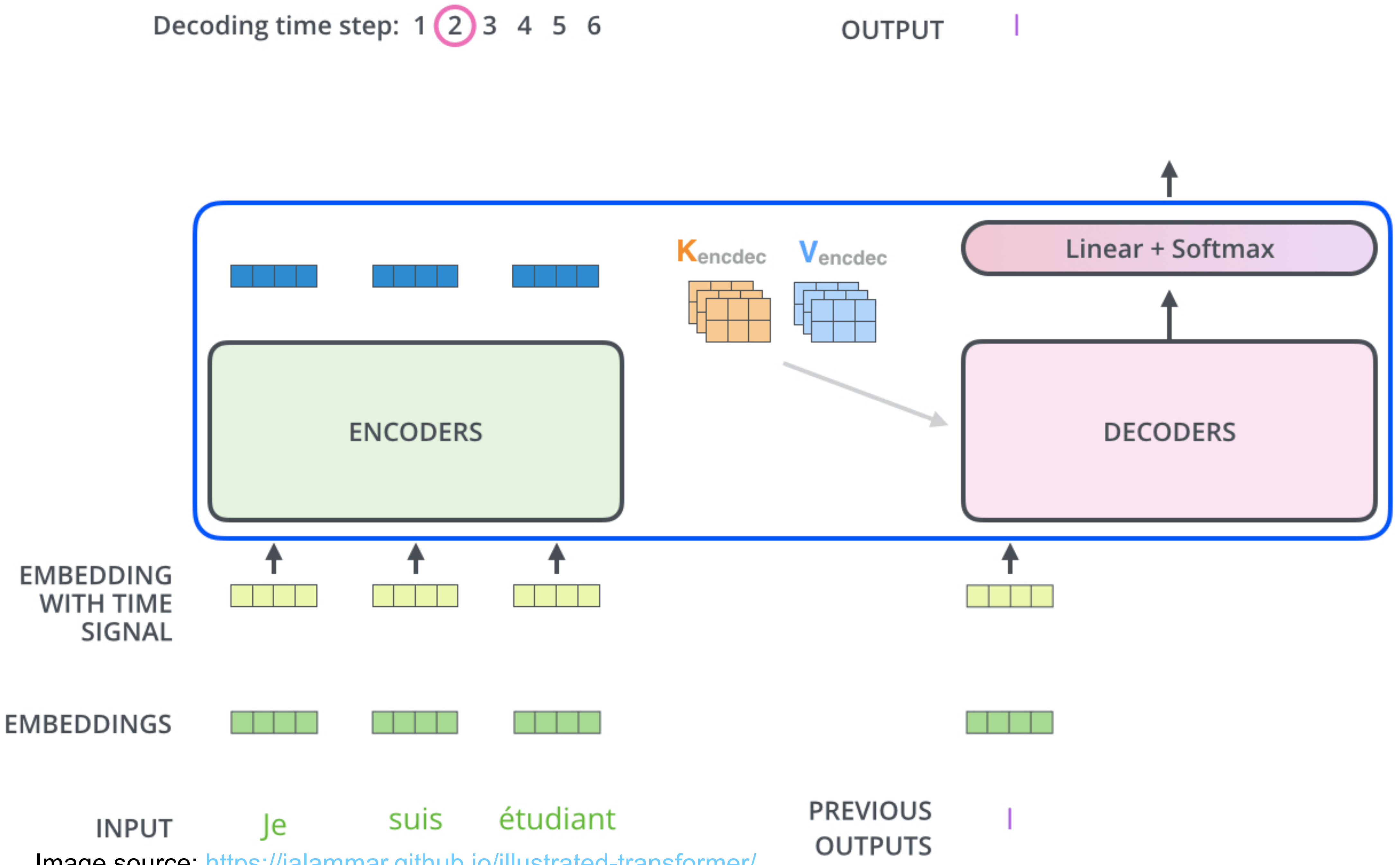
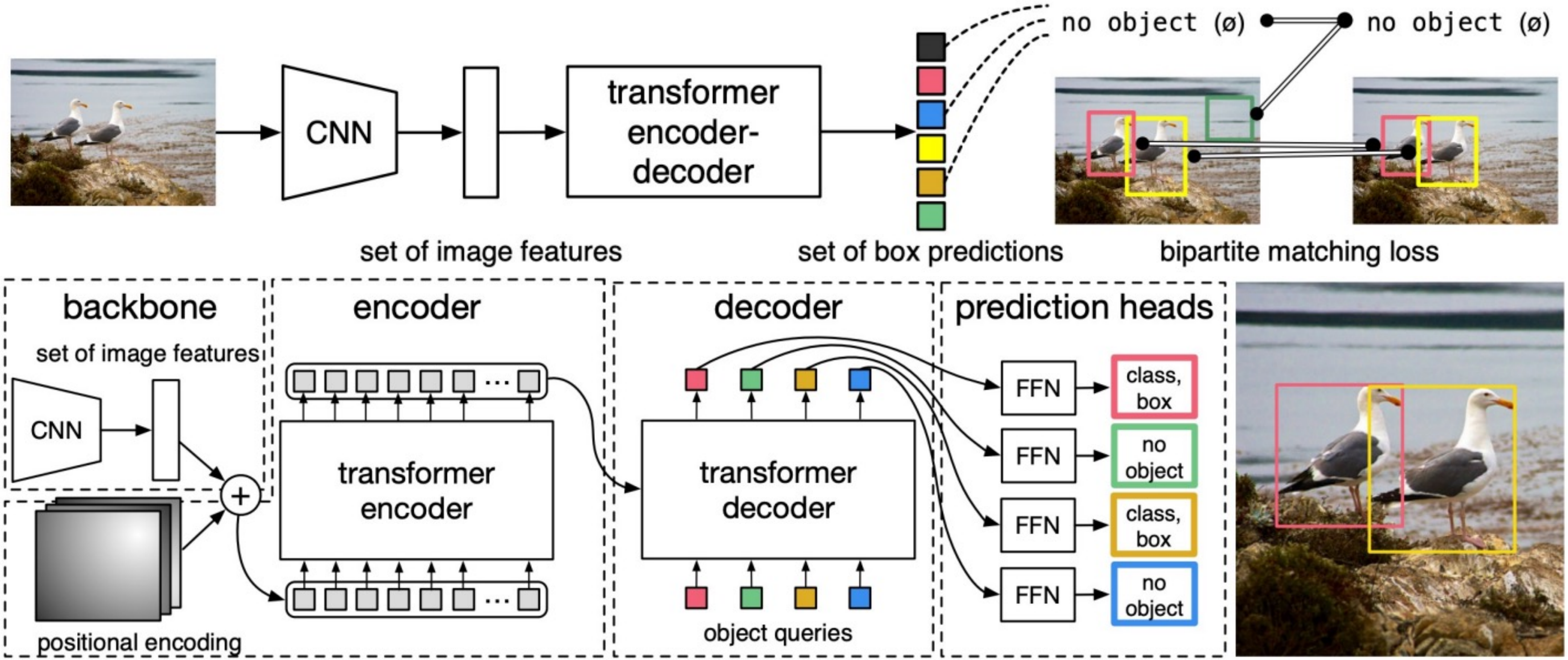


Image source: https://jalammar.github.io/illustrated-transformer/

Object Detection with Transformers: DETR



Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

Slide original source: [Deep Learning for Computer Vision 2022 Lecture 18]

Autoencoders & VAE

02

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

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

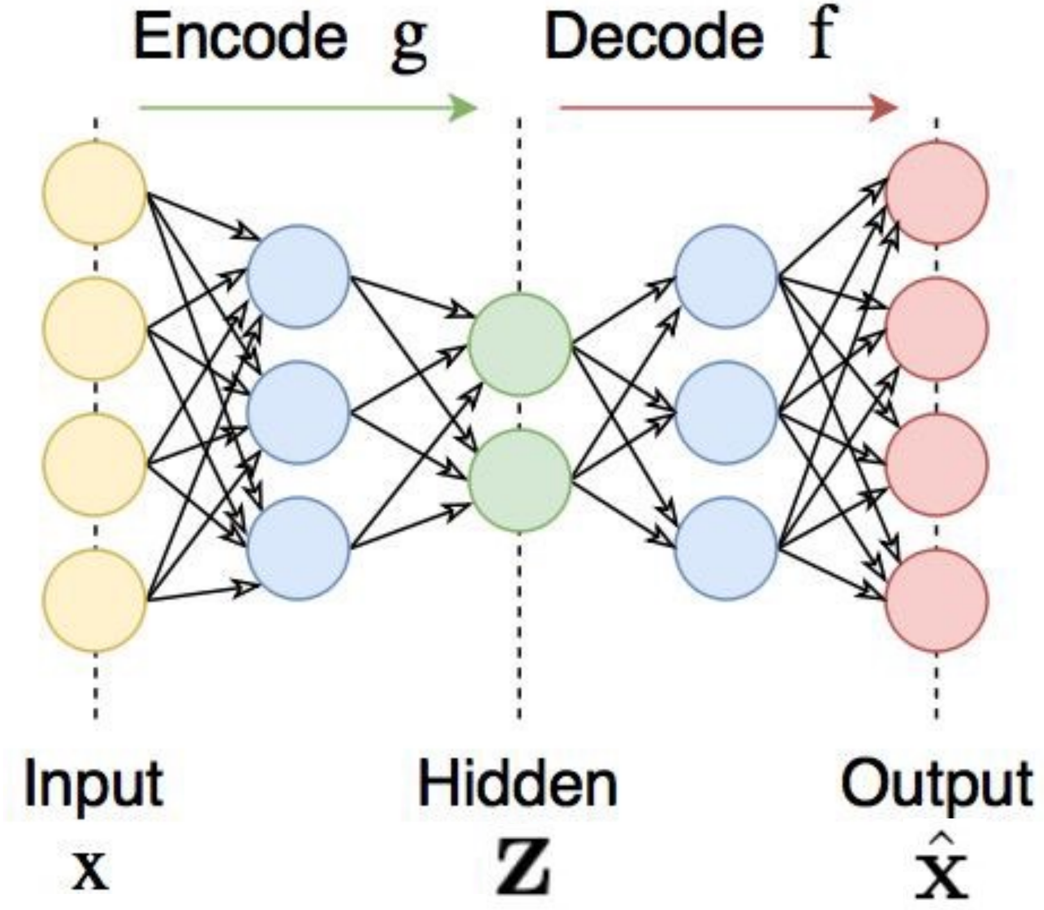


Image source: [Habr post on autoencoders and GANs]

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

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

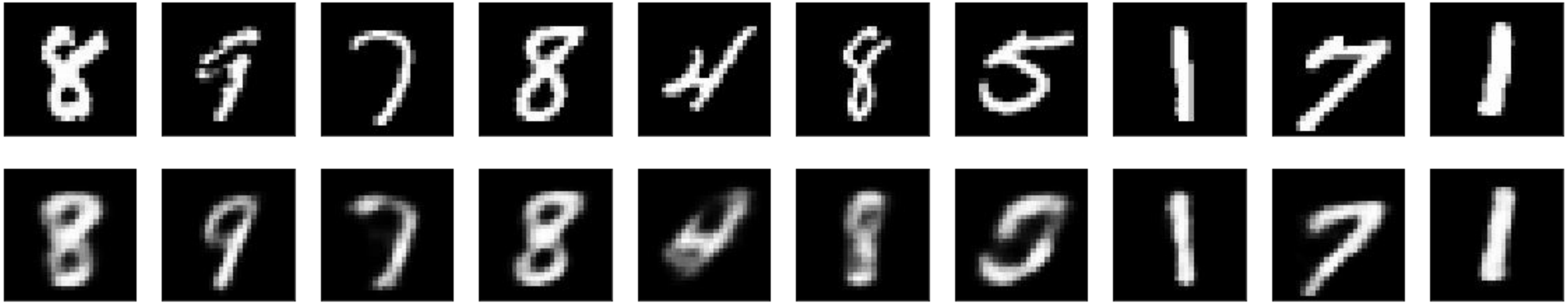
Optimal parameters learned w.r.t. loss function L

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

Simple example: PCA

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

PCA performance on MNIST



16 components

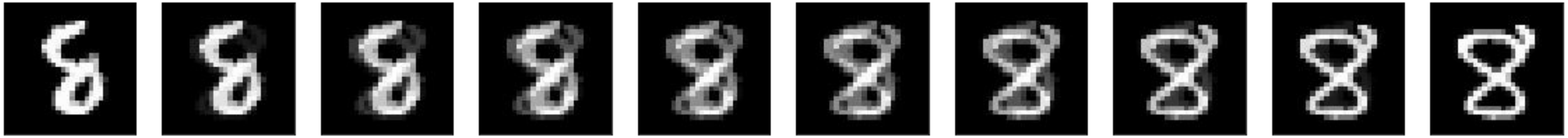
Convolutional performance on MNIST

7 x 7 latent space

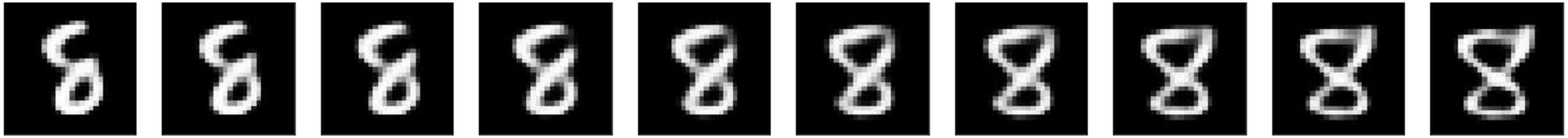
Homotopy between samples

10 steps between samples

- In original feature space (28 x 28):



- In latent space (7 x 7):



Latent space structure

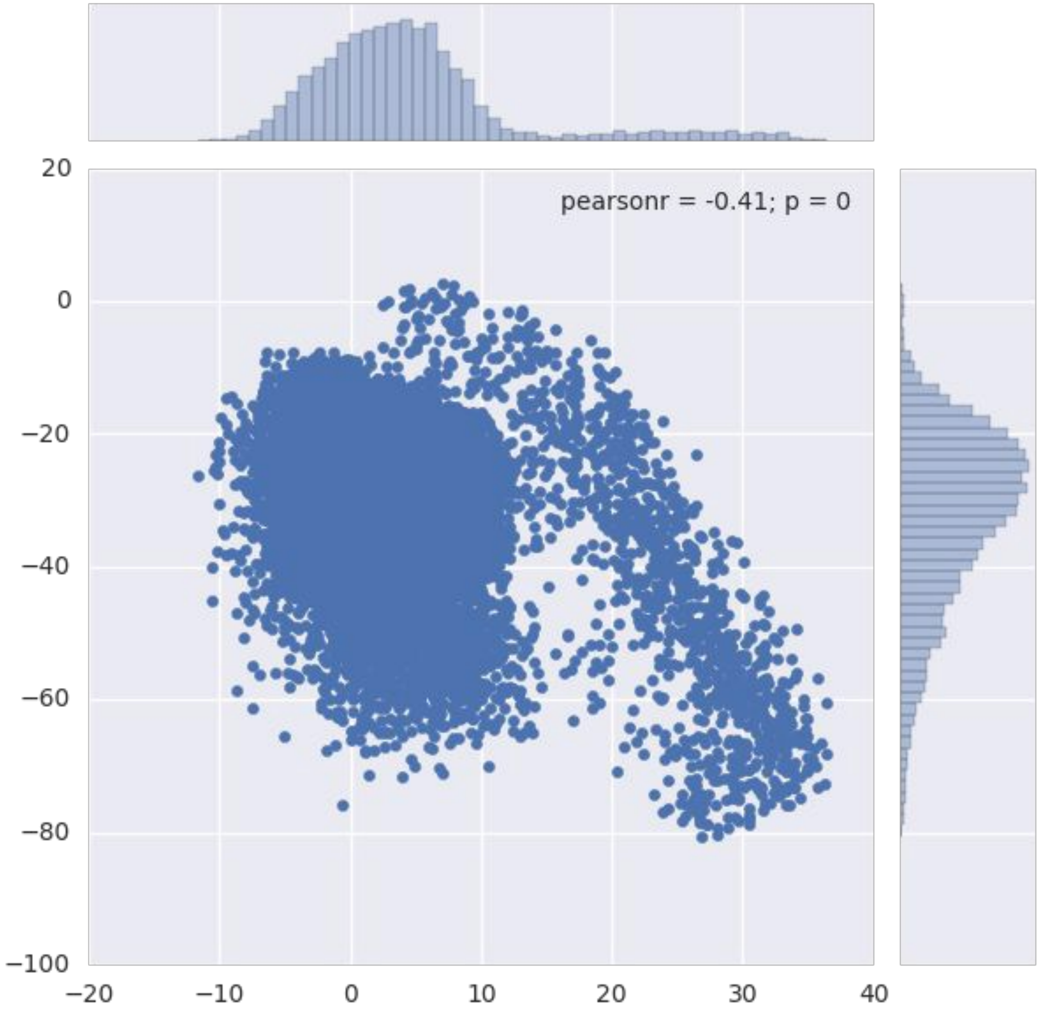


Image source: [Habr post on autoencoders and GANs]()

VAE intuition

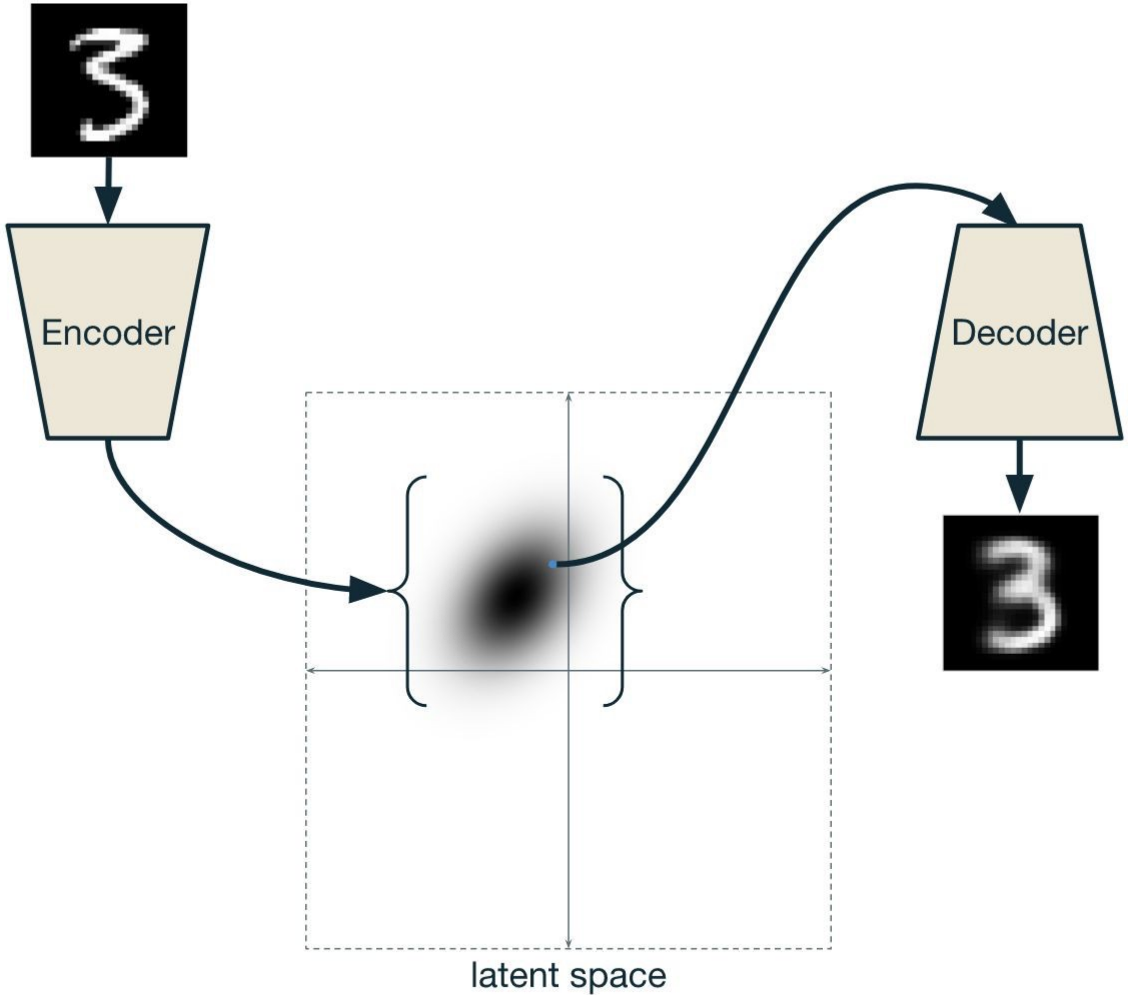


Image source: [Conditional Variational Autoencoders by Isaac Dykeman, 2016]

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

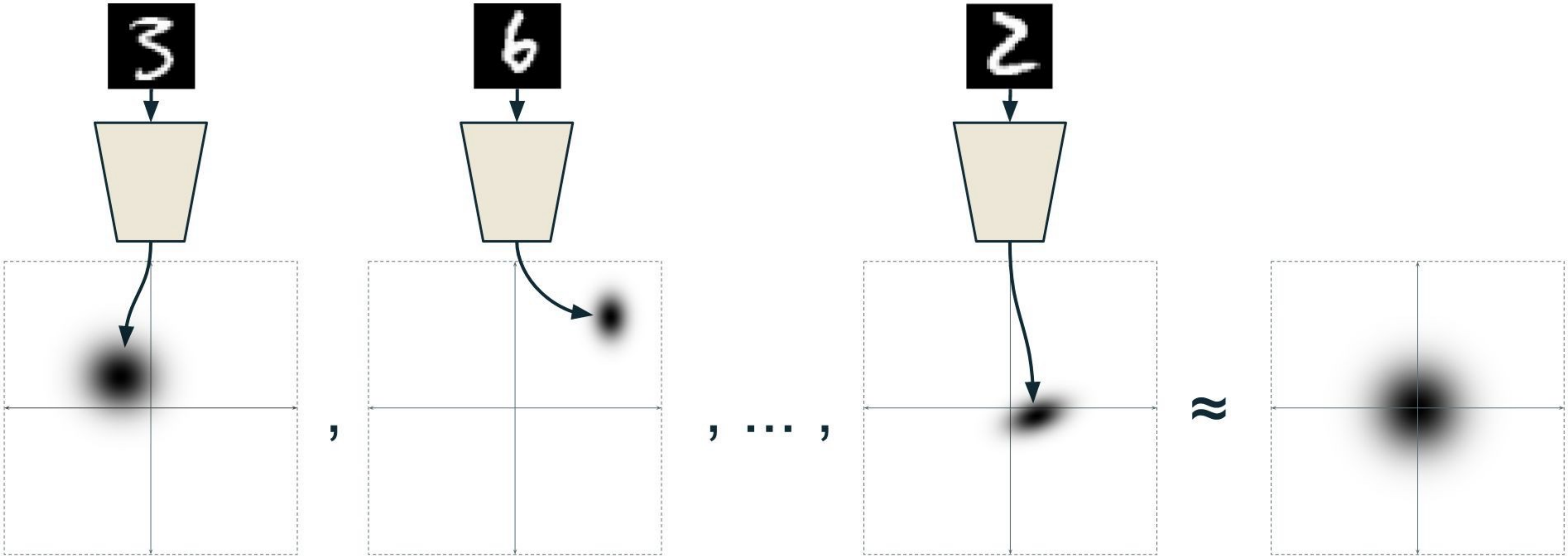


Image source: [Conditional Variational Autoencoders by Isaac Dykeman, 2016]

VAE so far

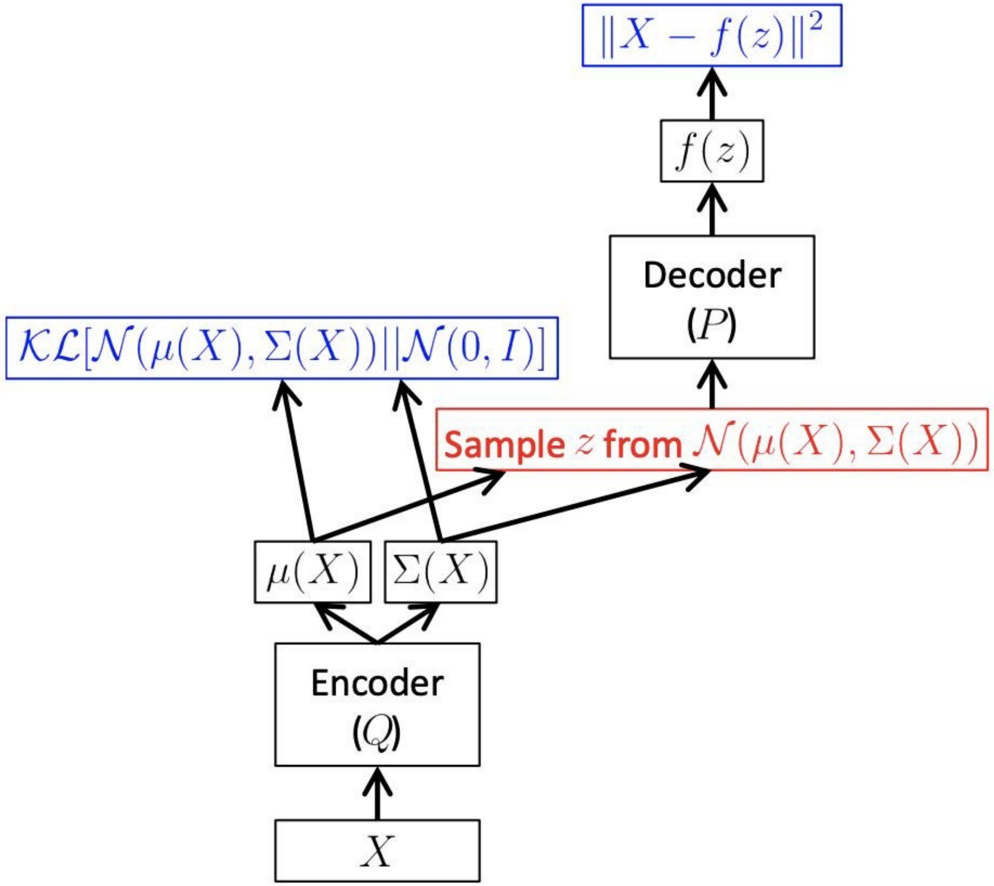


Image source: [Tutorial on Variational Autoencoders by Carl Doersch, 2016]

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

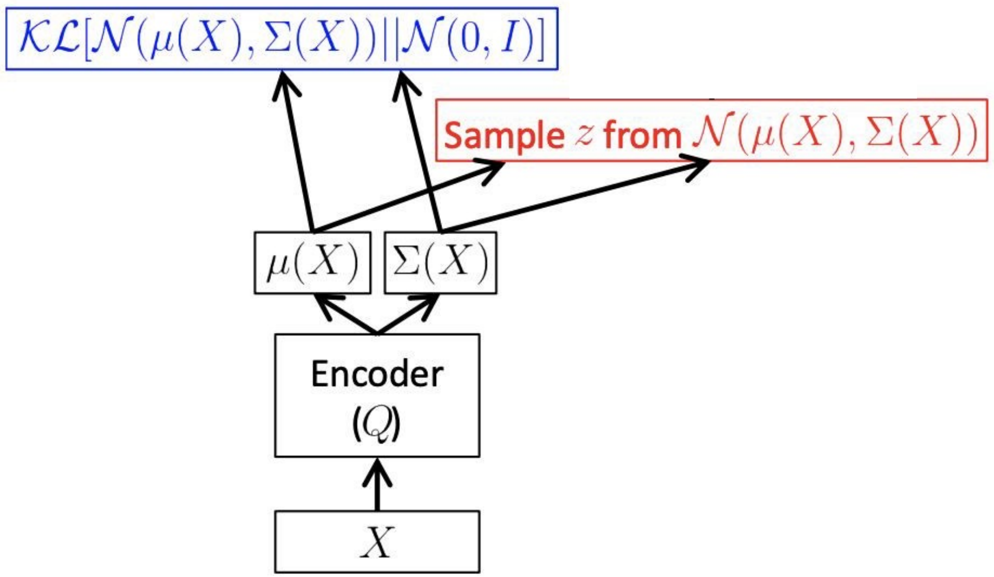


Image source: [Tutorial on Variational Autoencoders by Carl Doersch, 2016]

VAE so far

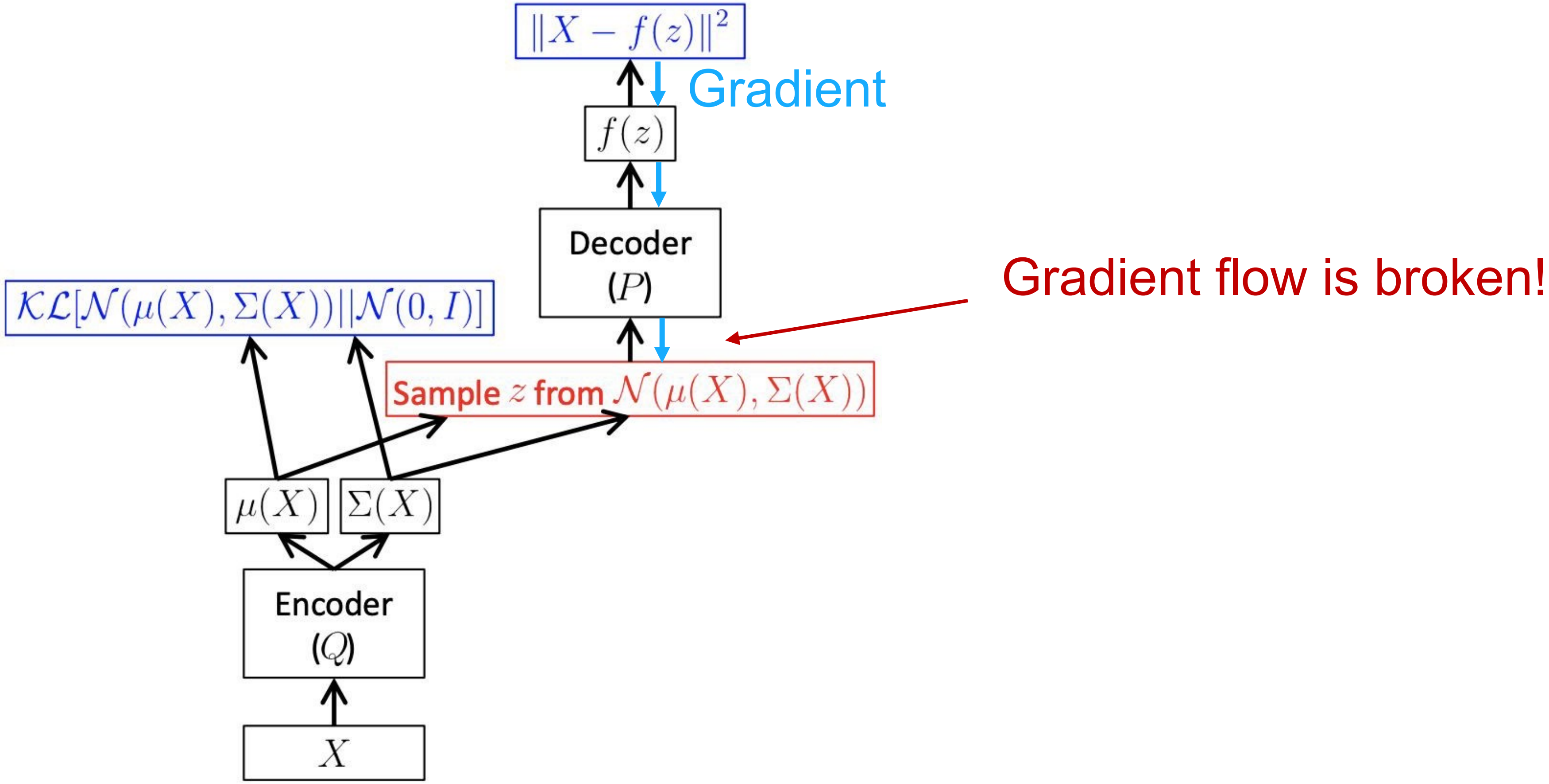
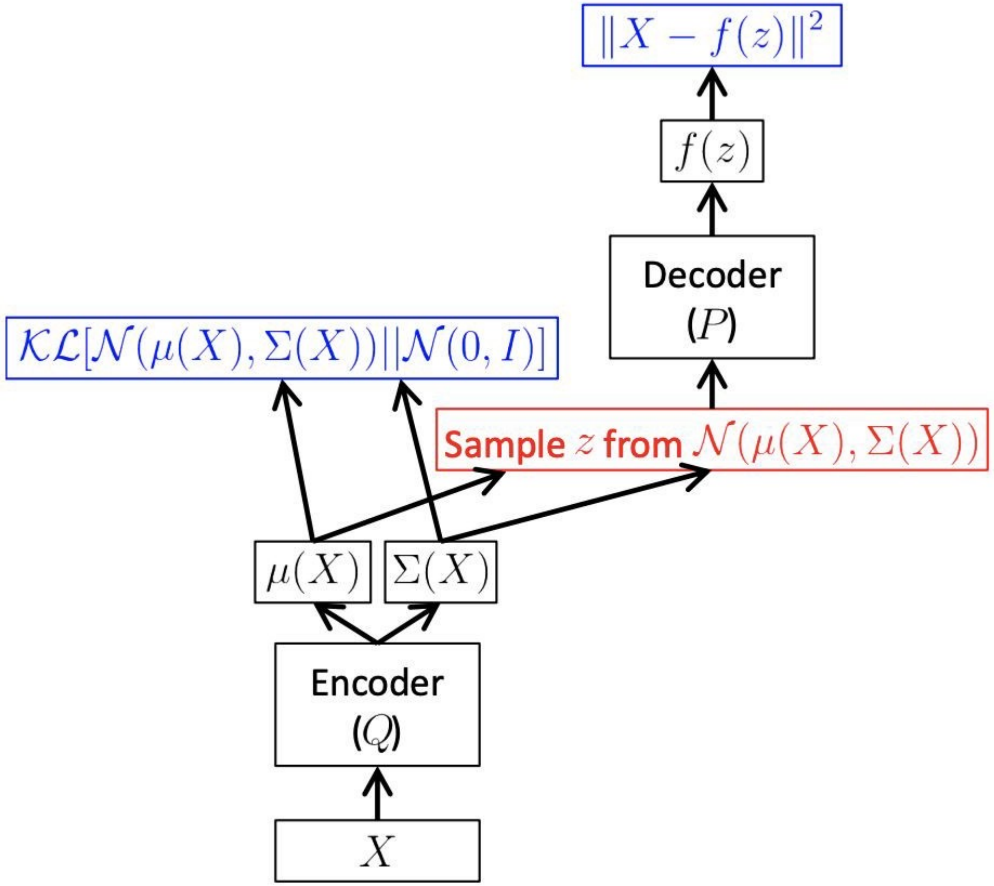


Image source: [Tutorial on Variational Autoencoders by Carl Doersch, 2016]

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

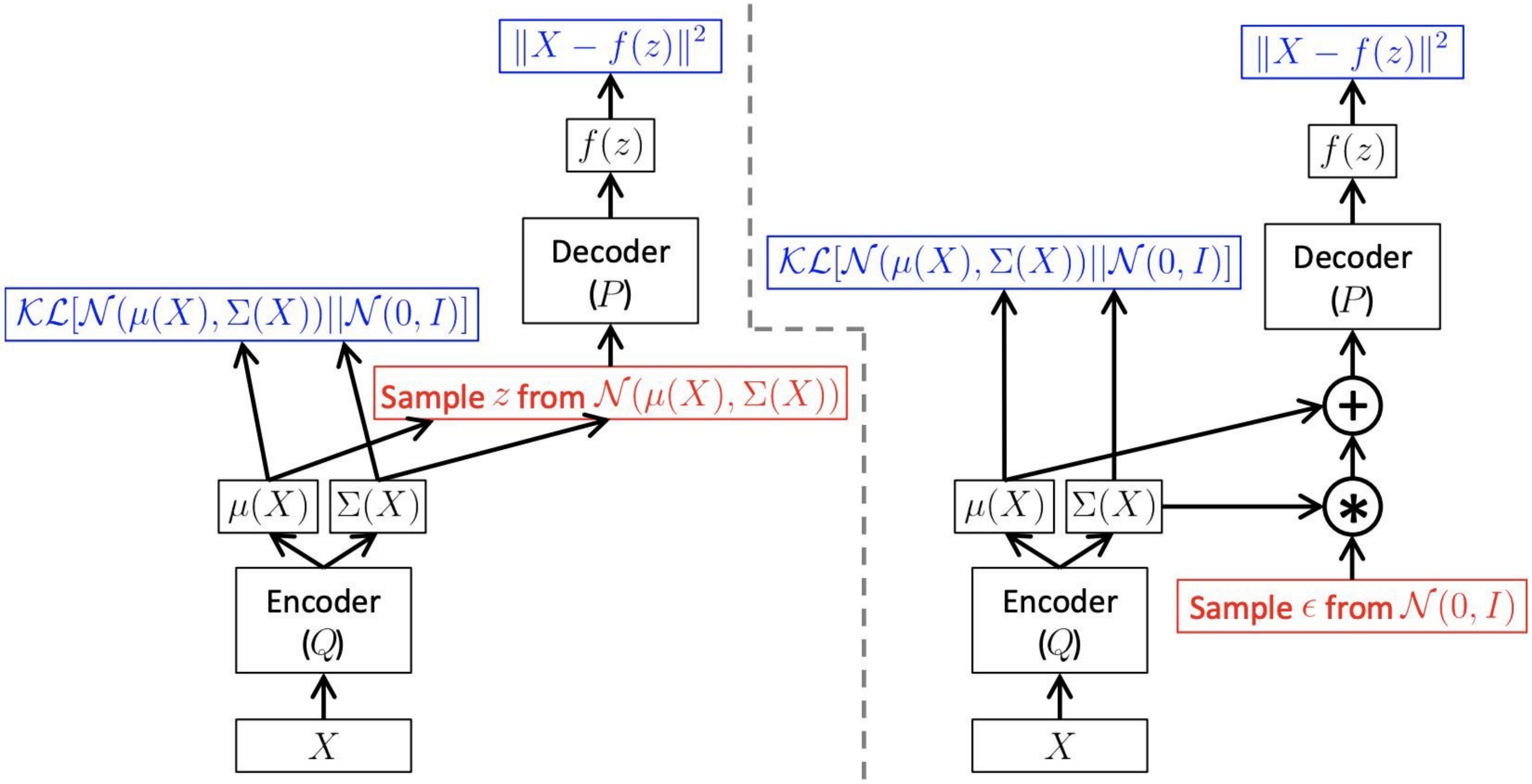


Image source: [Tutorial on Variational Autoencoders by Carl Doersch, 2016]

Structure of the latent space

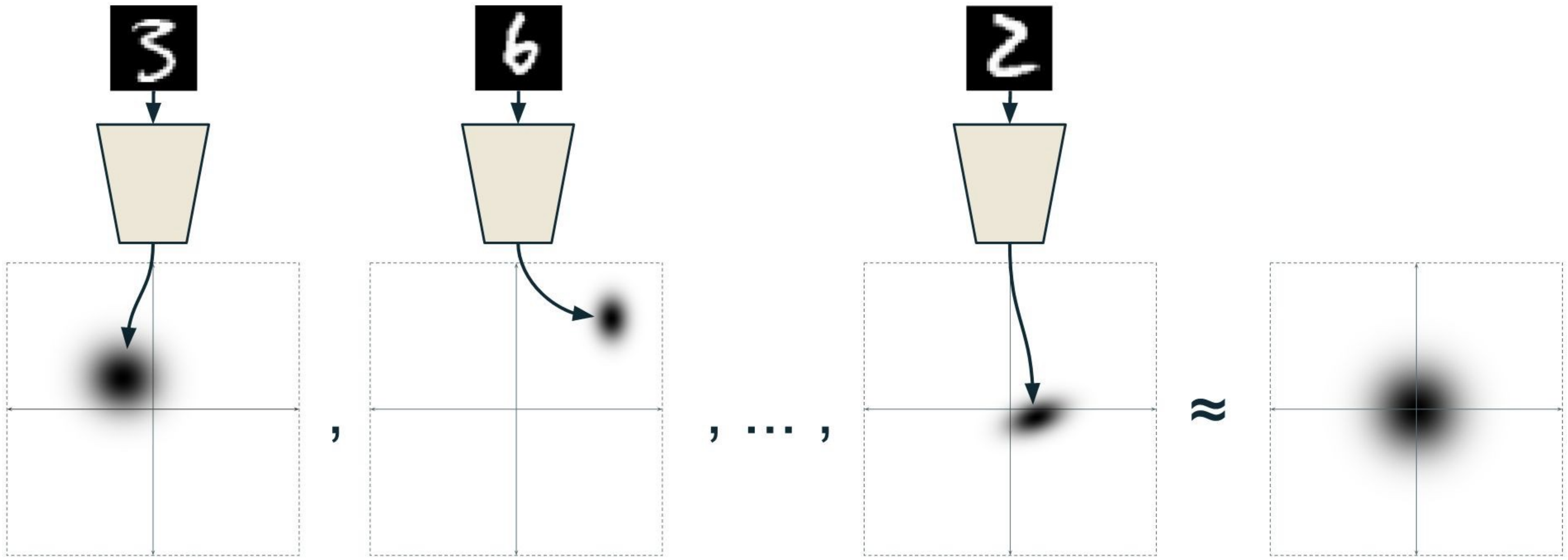


Image source: [Conditional Variational Autoencoders by Isaac Dykeman, 2016]

Conditional VAE intuition

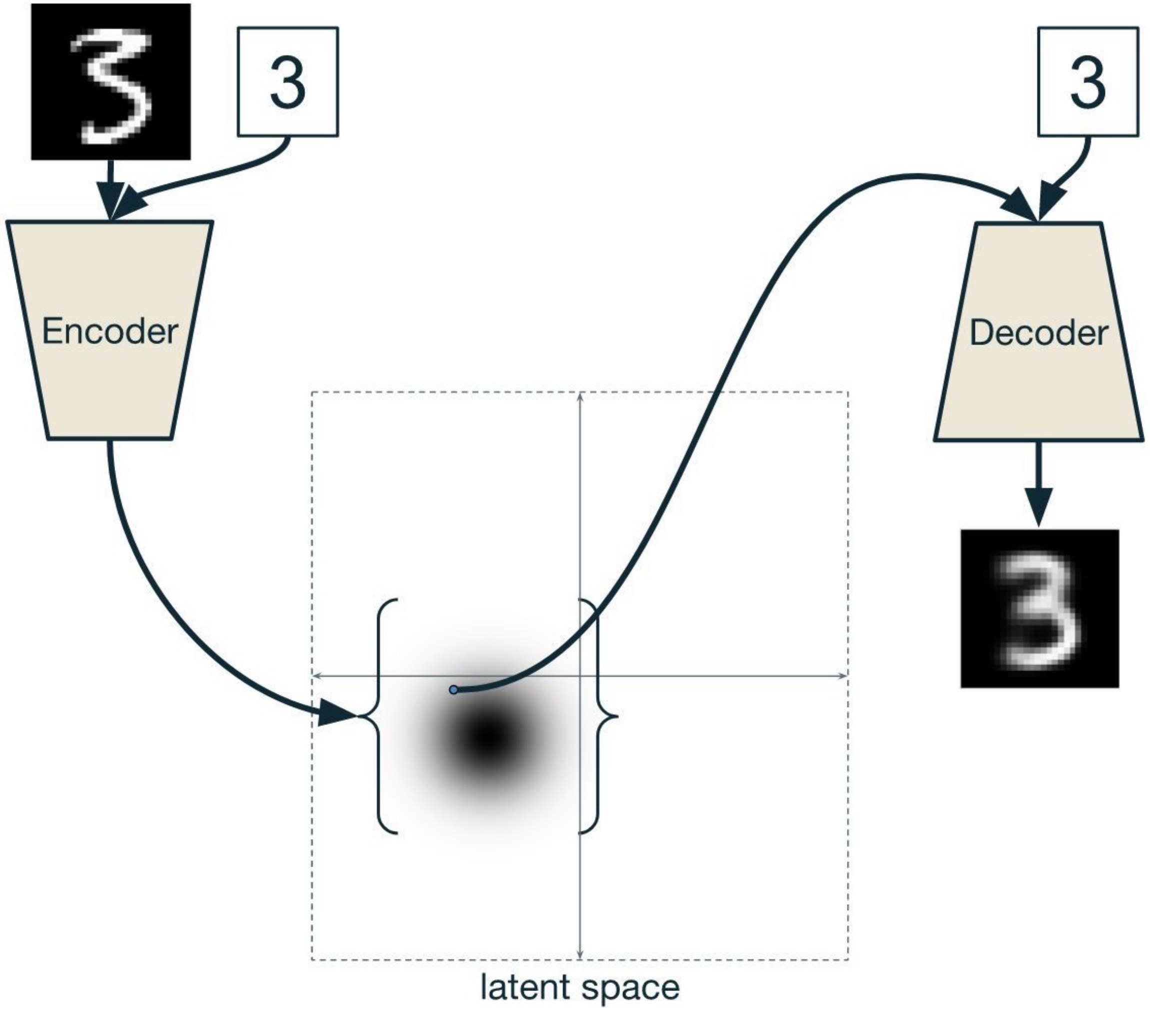


Image source: [Conditional Variational Autoencoders by Isaac Dykeman, 2016]

Conditional VAE

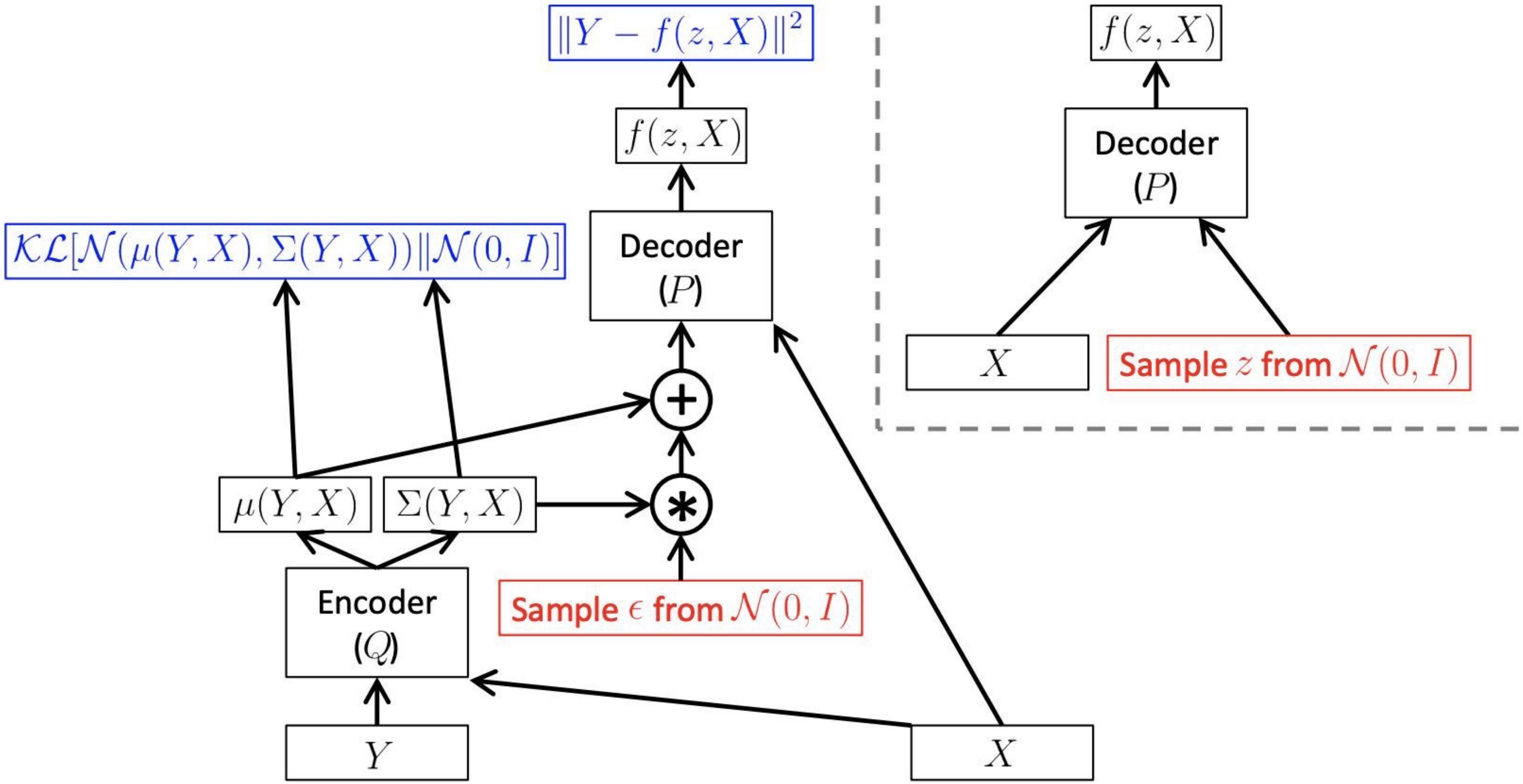


Image source: [Tutorial on Variational Autoencoders by Carl Doersch, 2016]()

Transferring style with cVAE

Seed label is 1

Once again

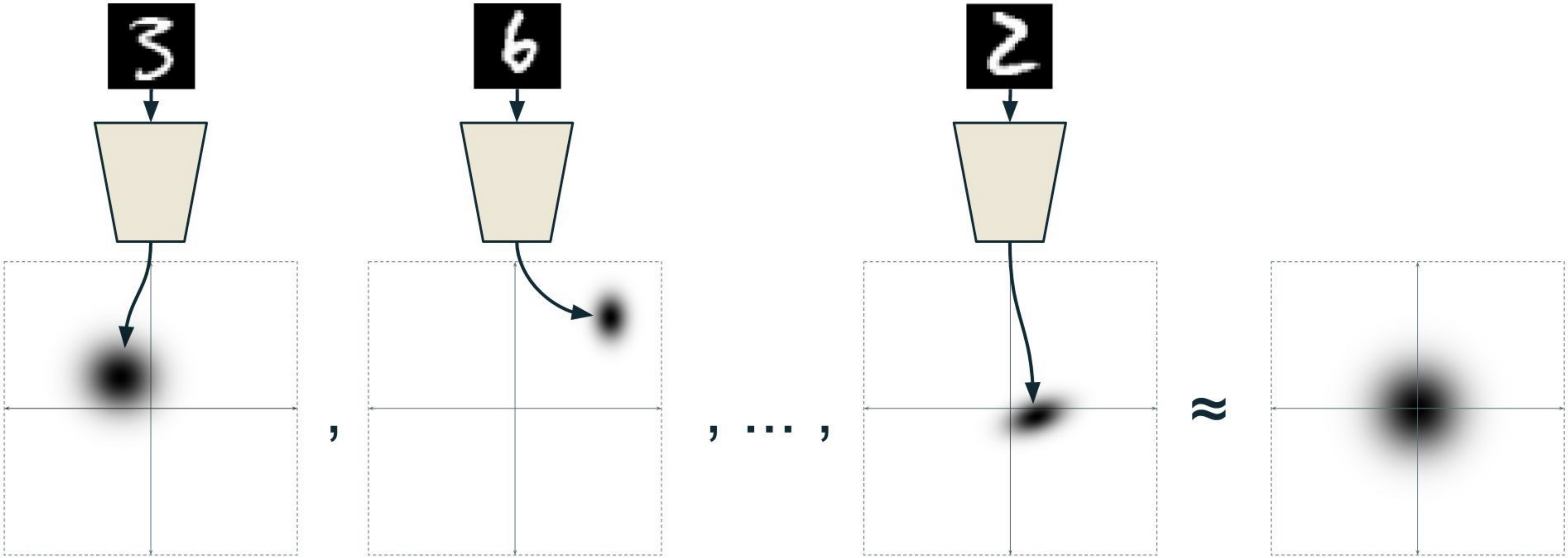
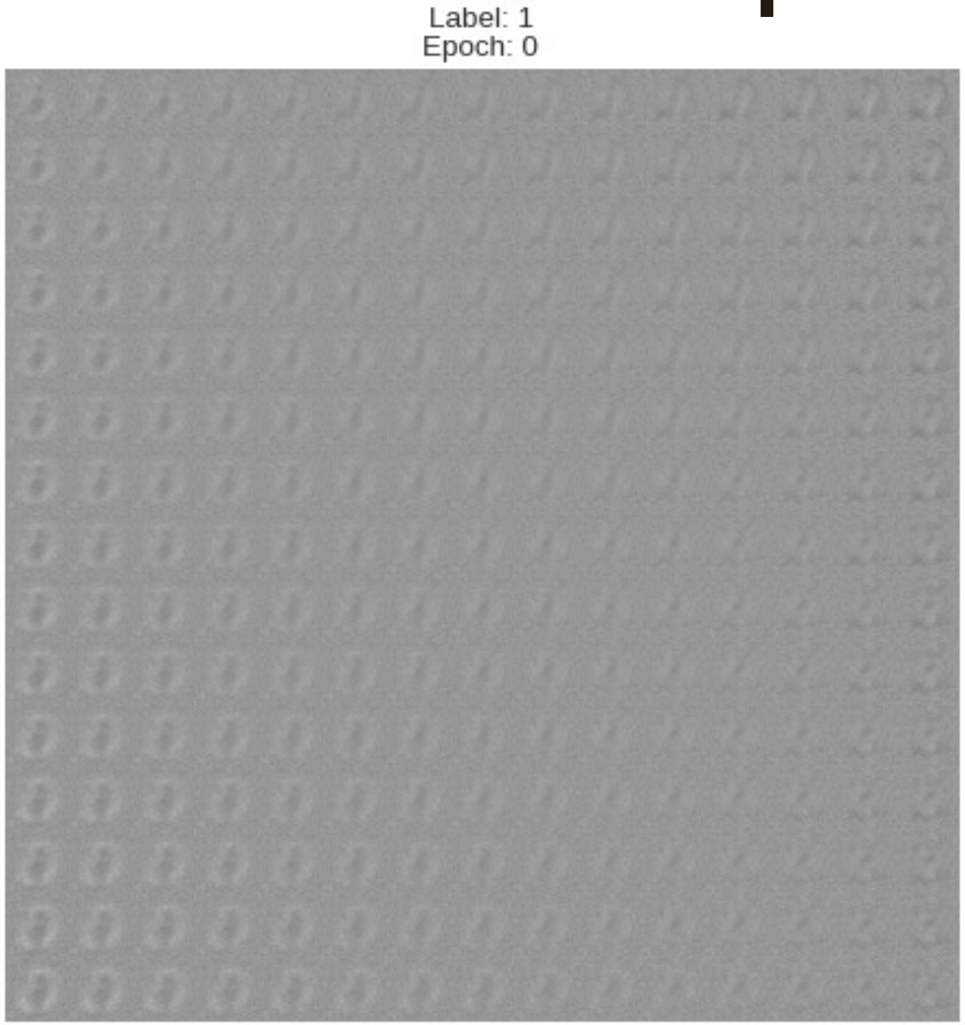
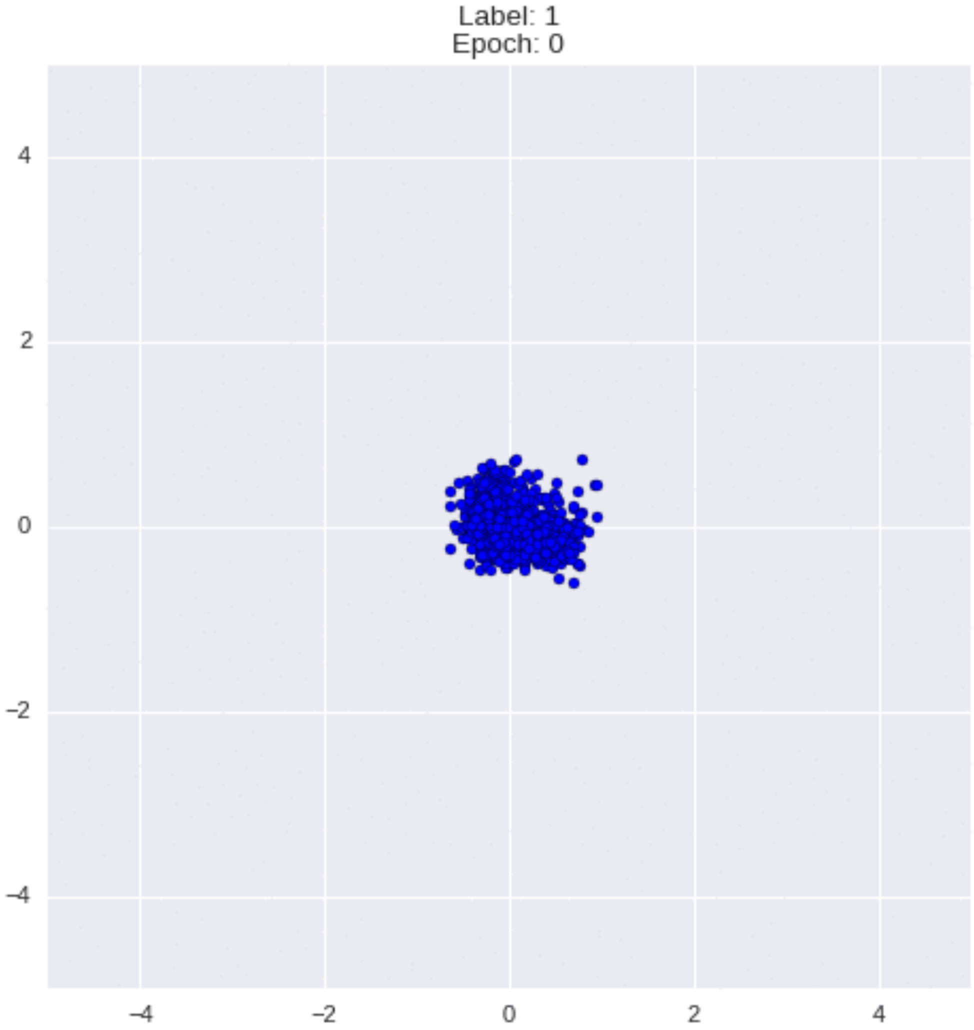


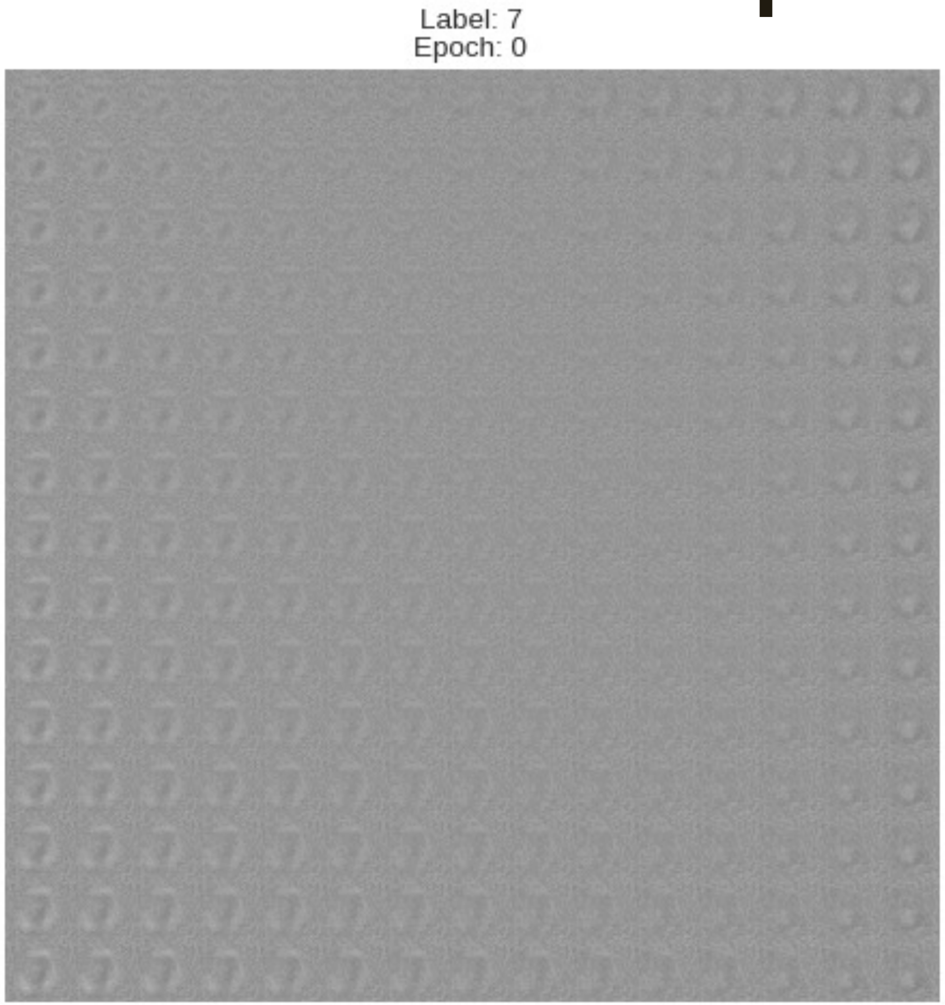
Image source: [Conditional Variational Autoencoders by Isaac Dykeman, 2016]

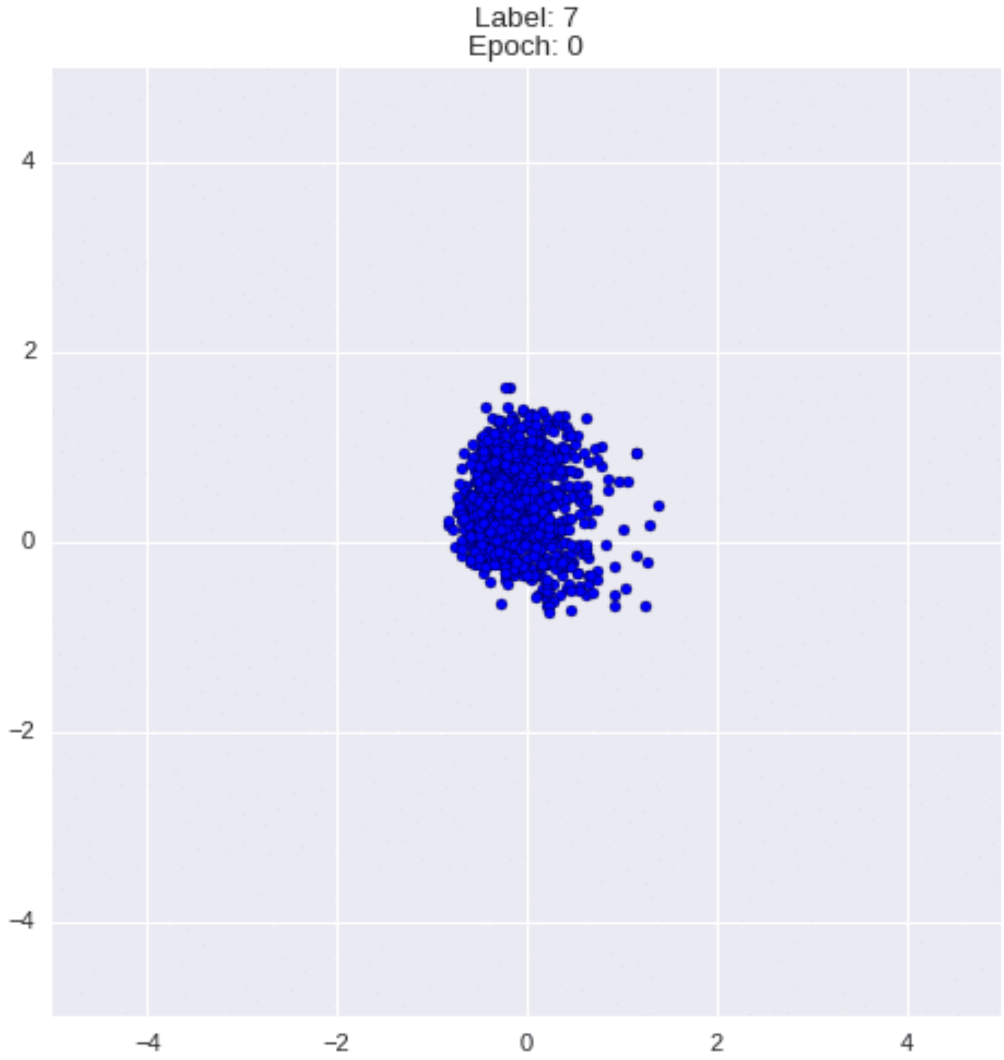
cVAE latent space distribution





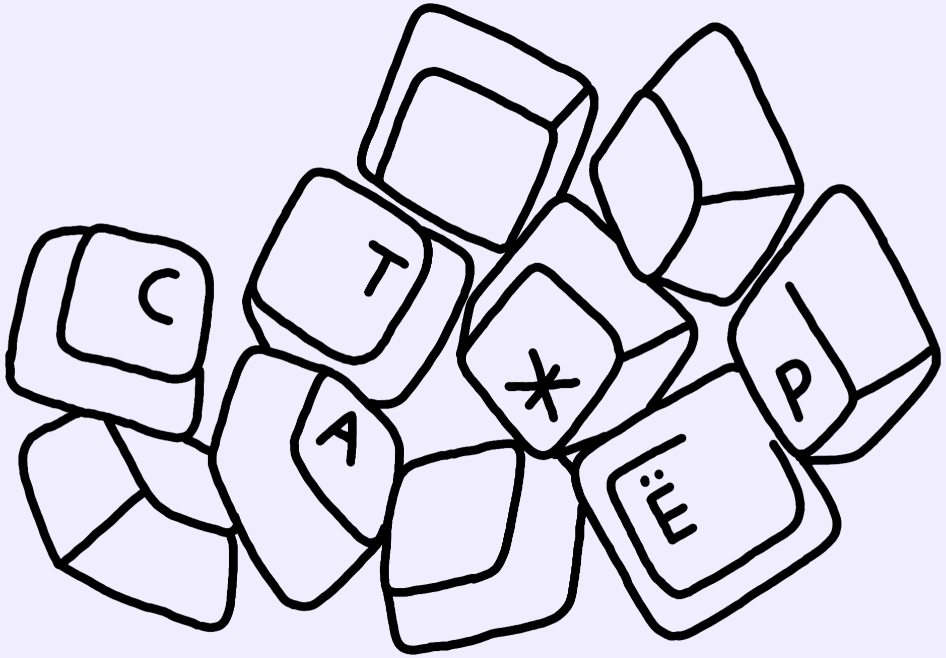
cVAE latent space distribution





GAN – Generative Adversarial Networks

03



GAN

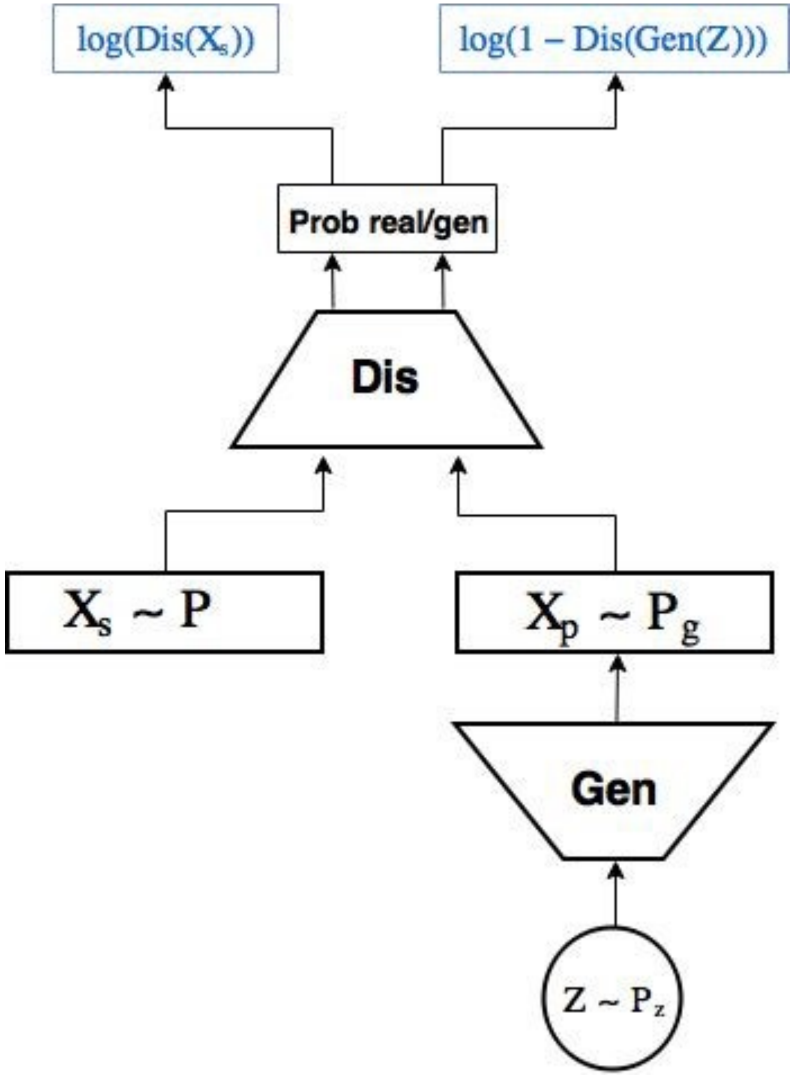
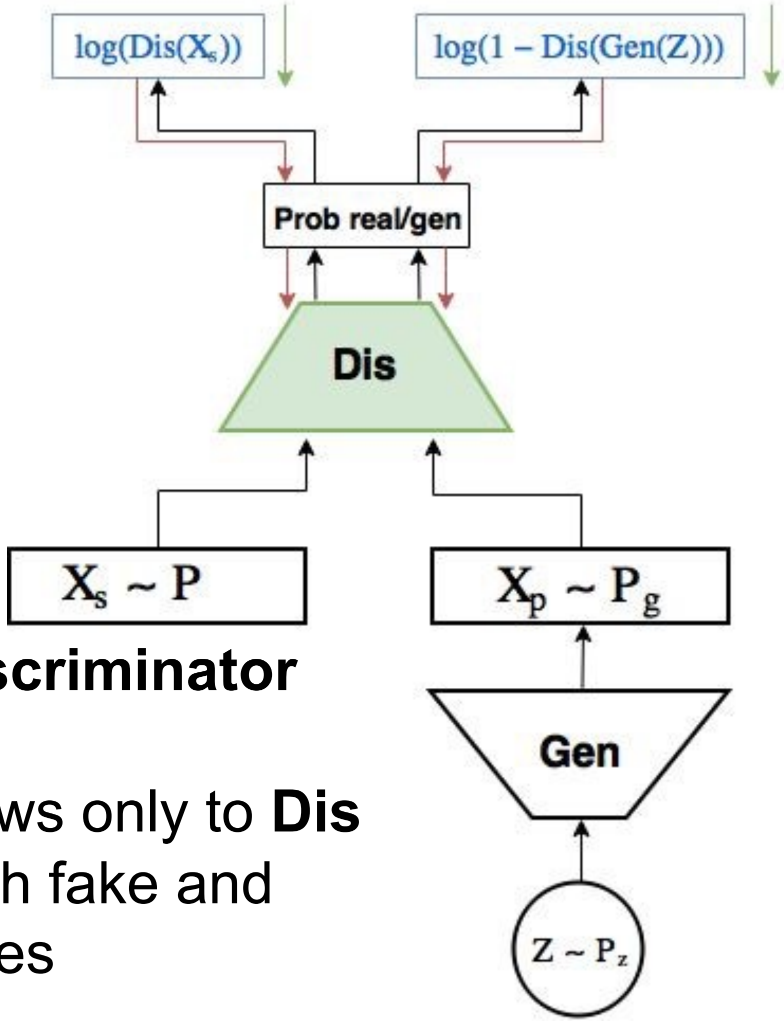
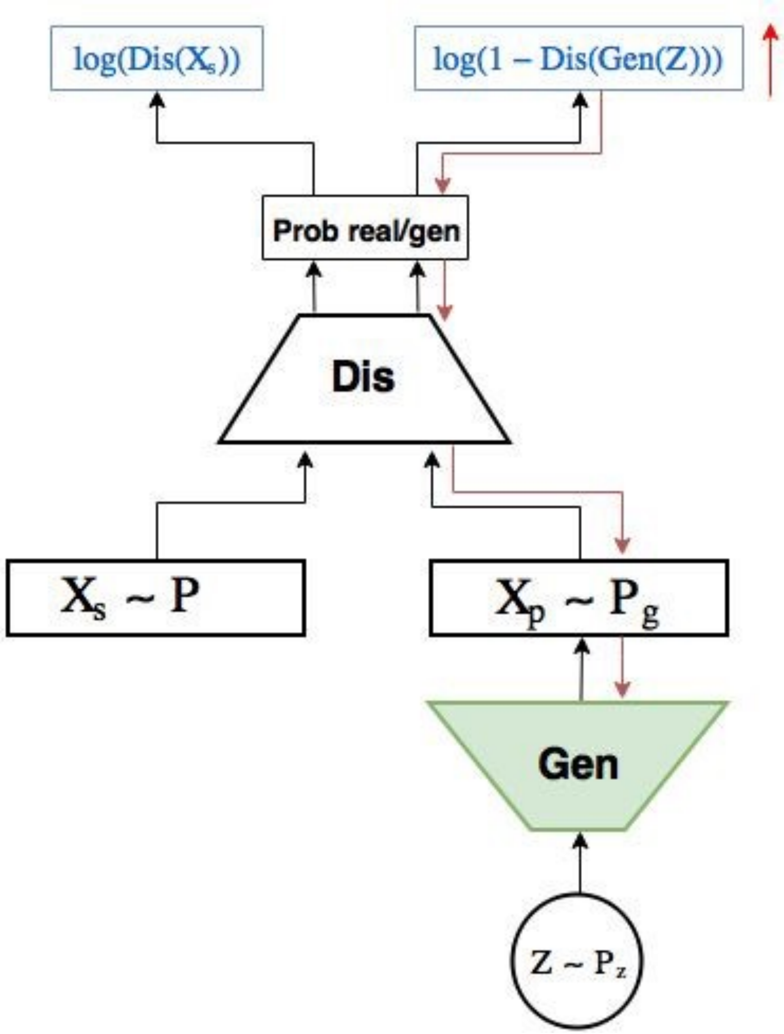


Image source: [Habr post on autoencoders and GANs]()

Training GAN



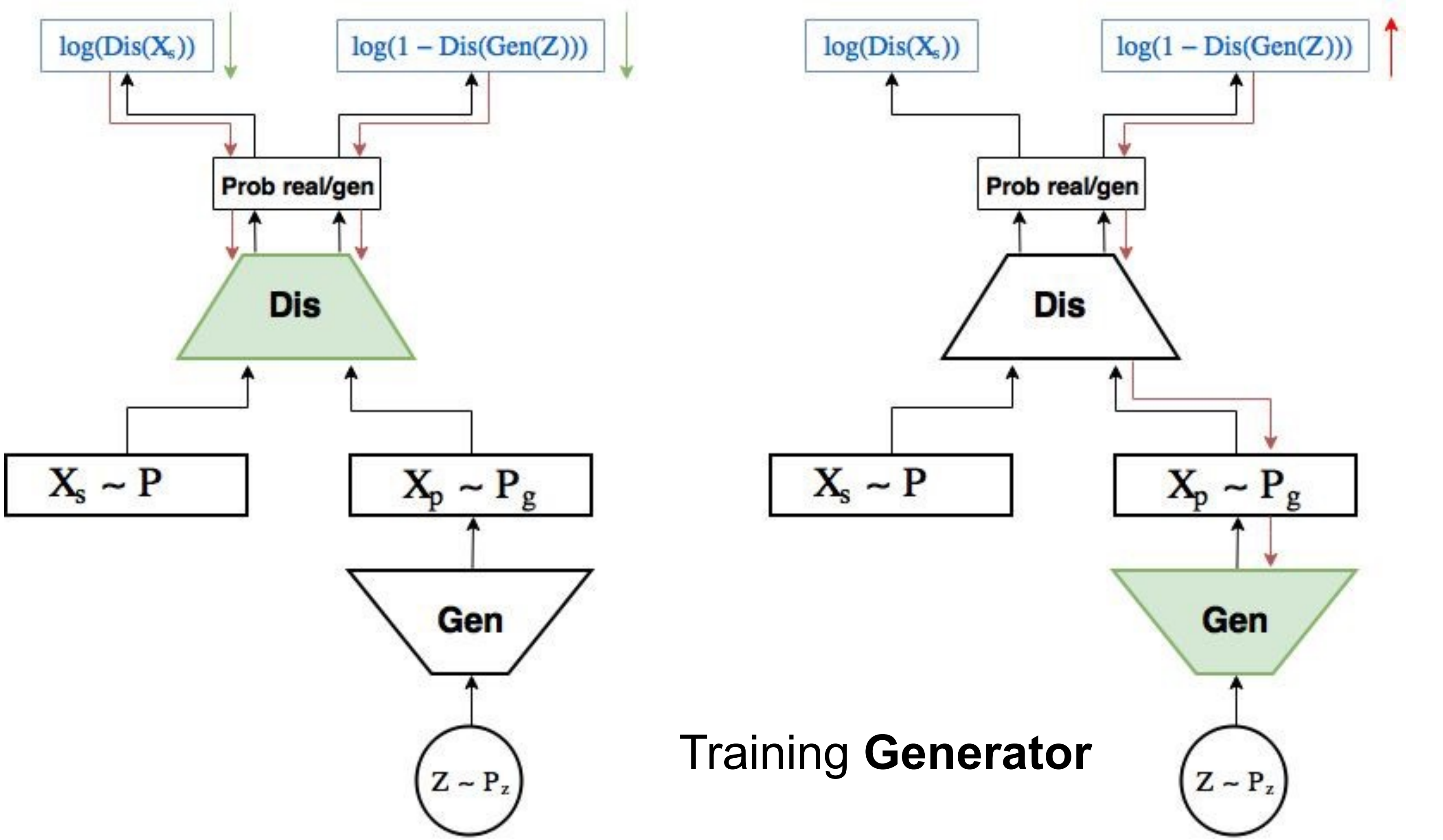


Training **Discriminator**

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

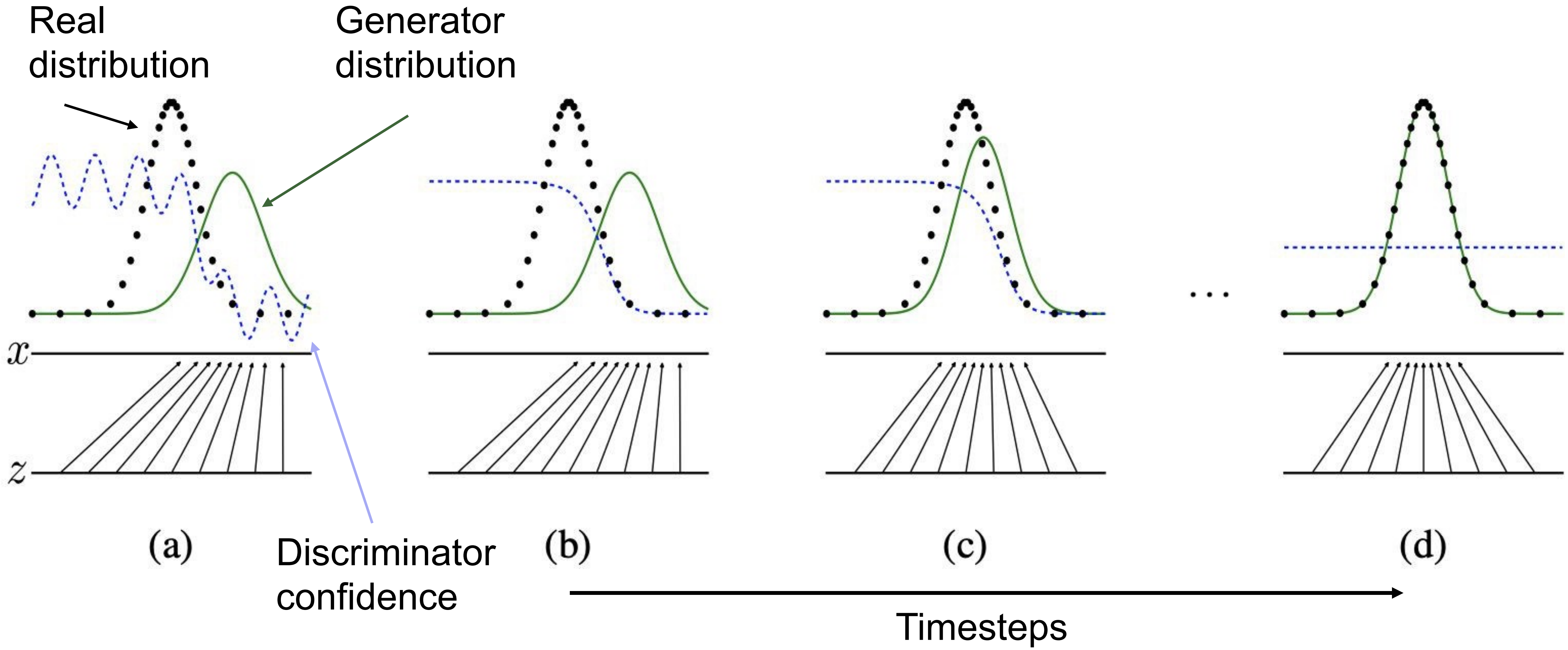


Image source: [Generative Adversarial Nets, Ian J. Goodfellow et al, 2014]

GAN manifold

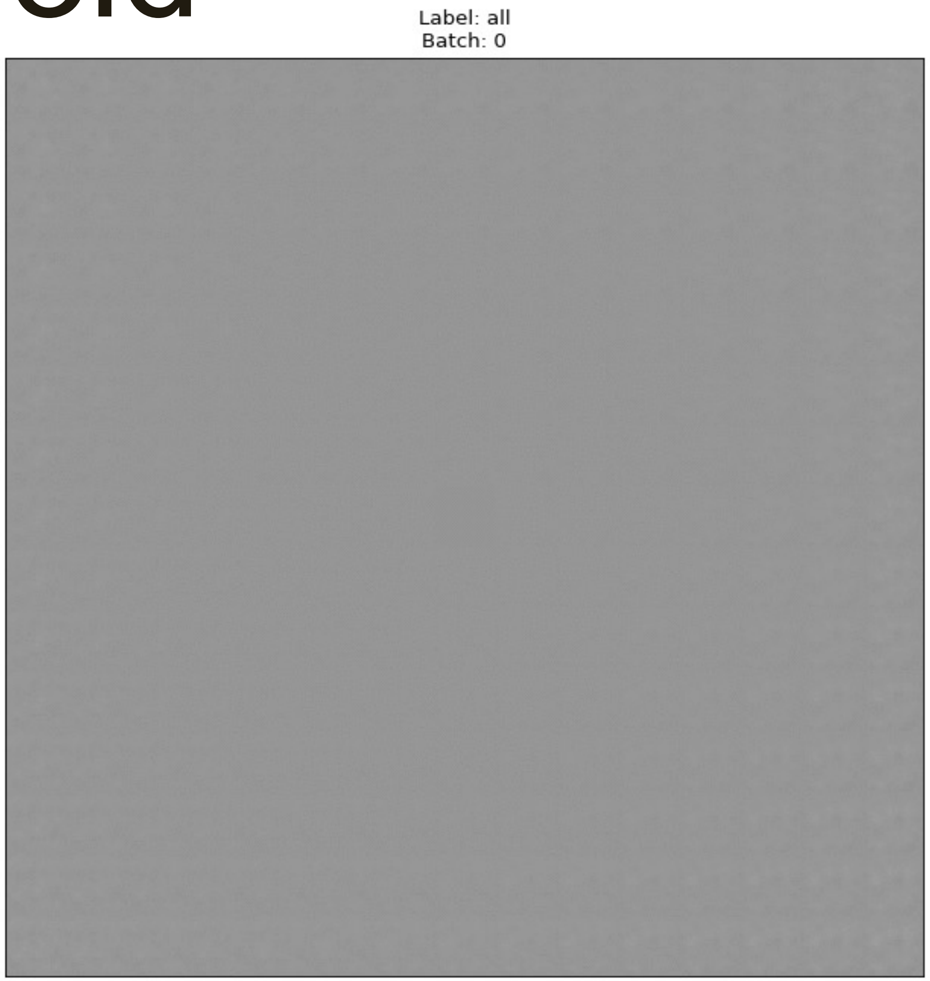


Image source: [Habr post on autoencoders and GANs]

Conditional GAN

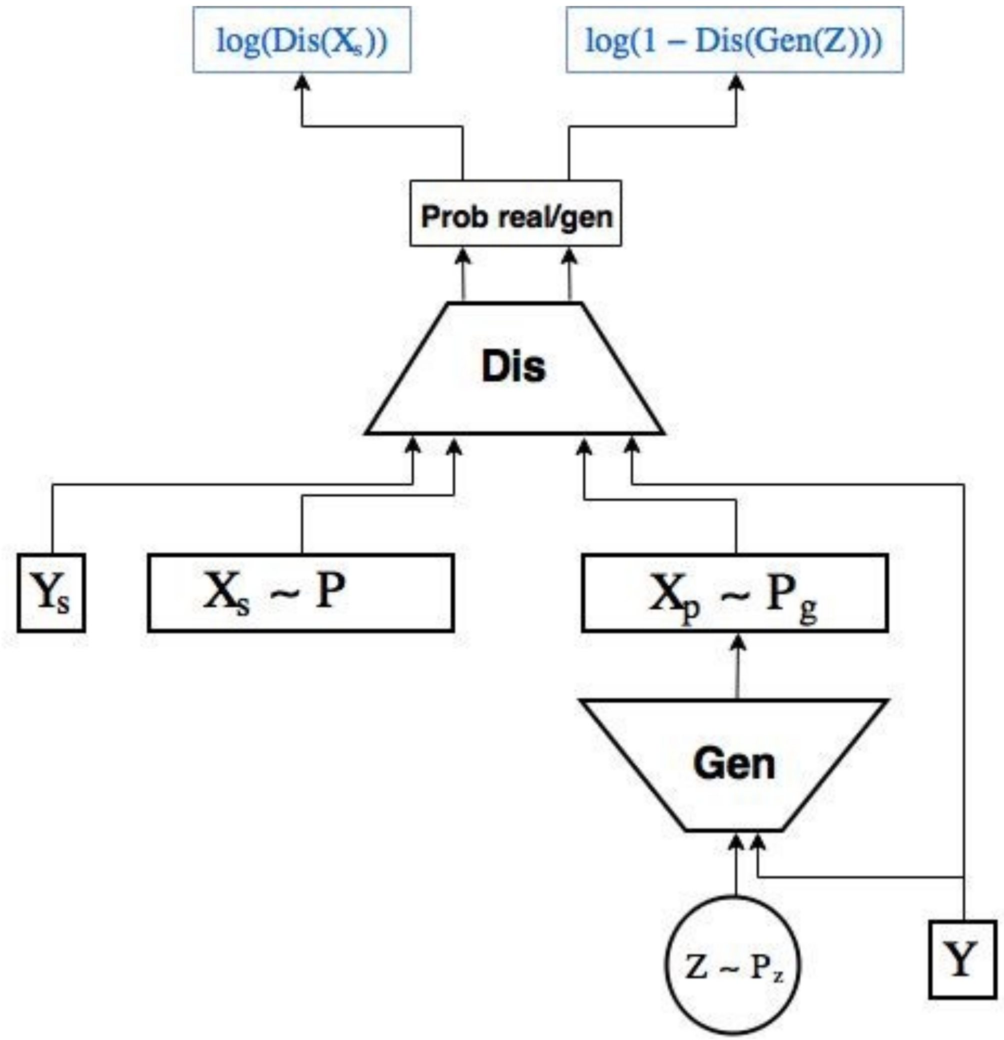
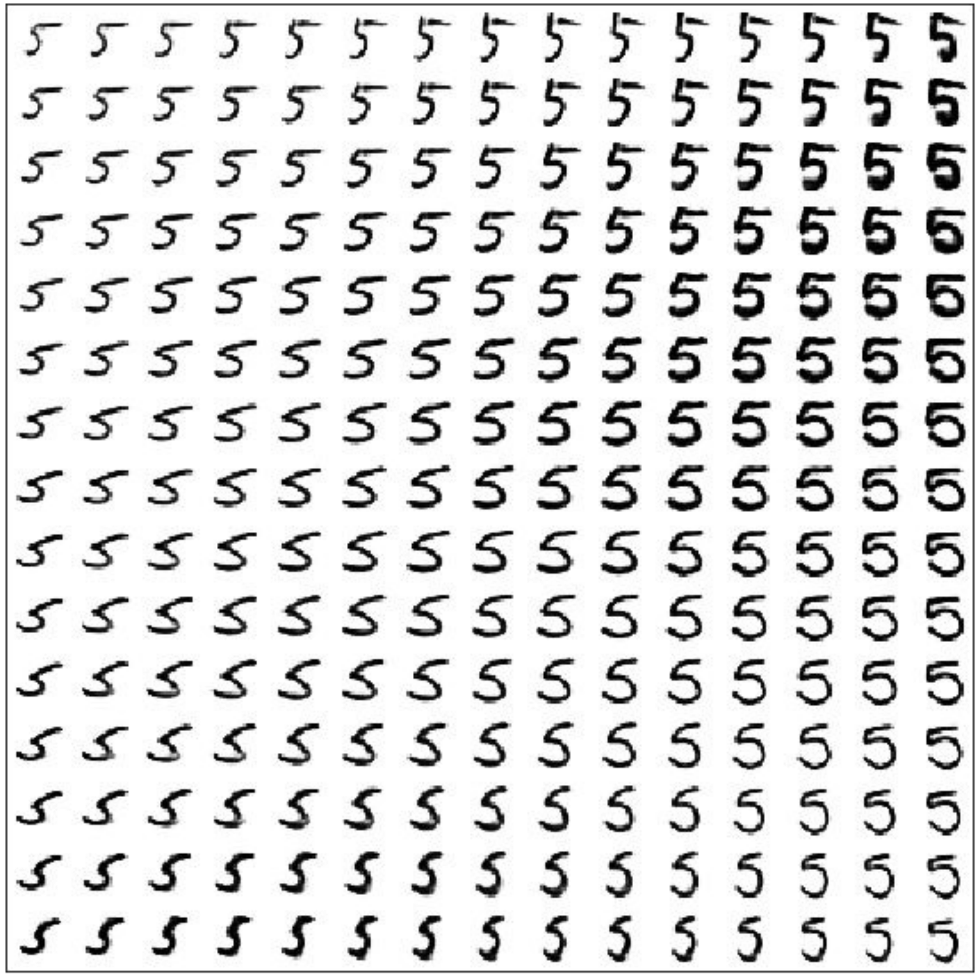


Image source: [Habr post on autoencoders and GANs]()

cGAN manifolds



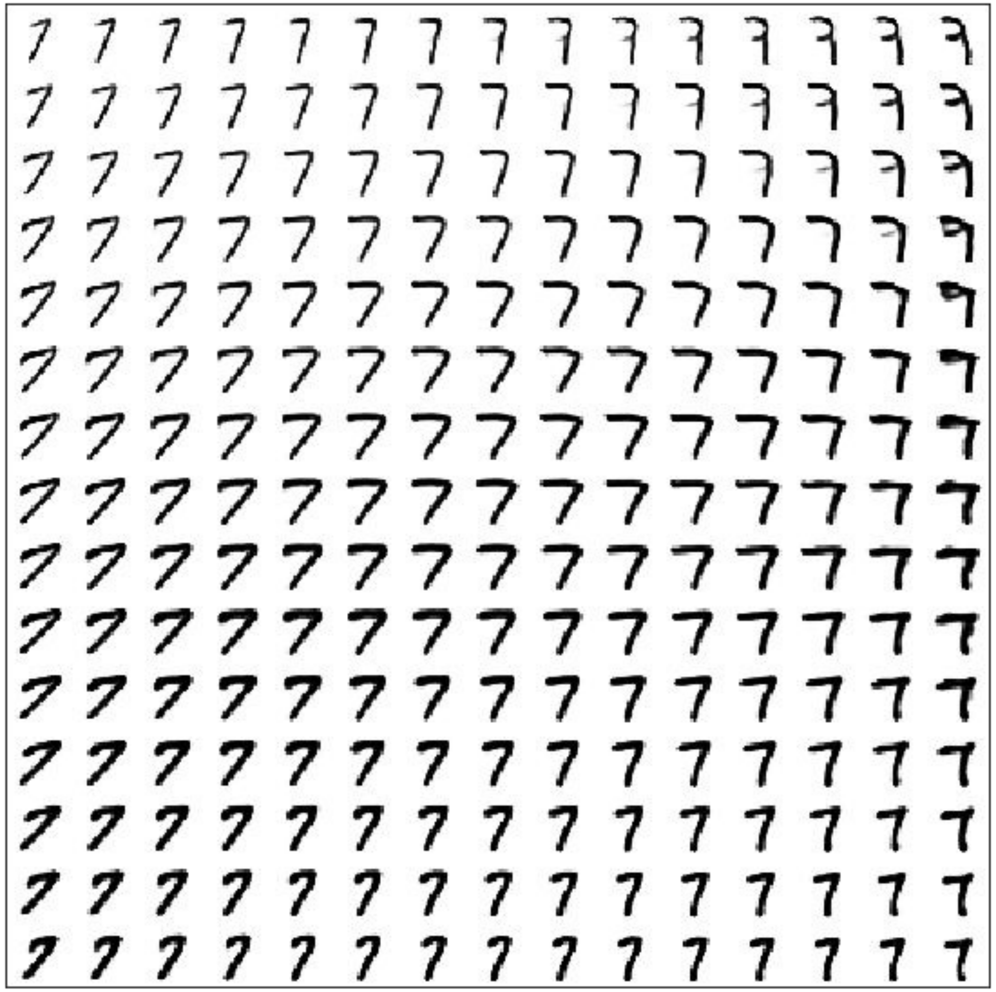
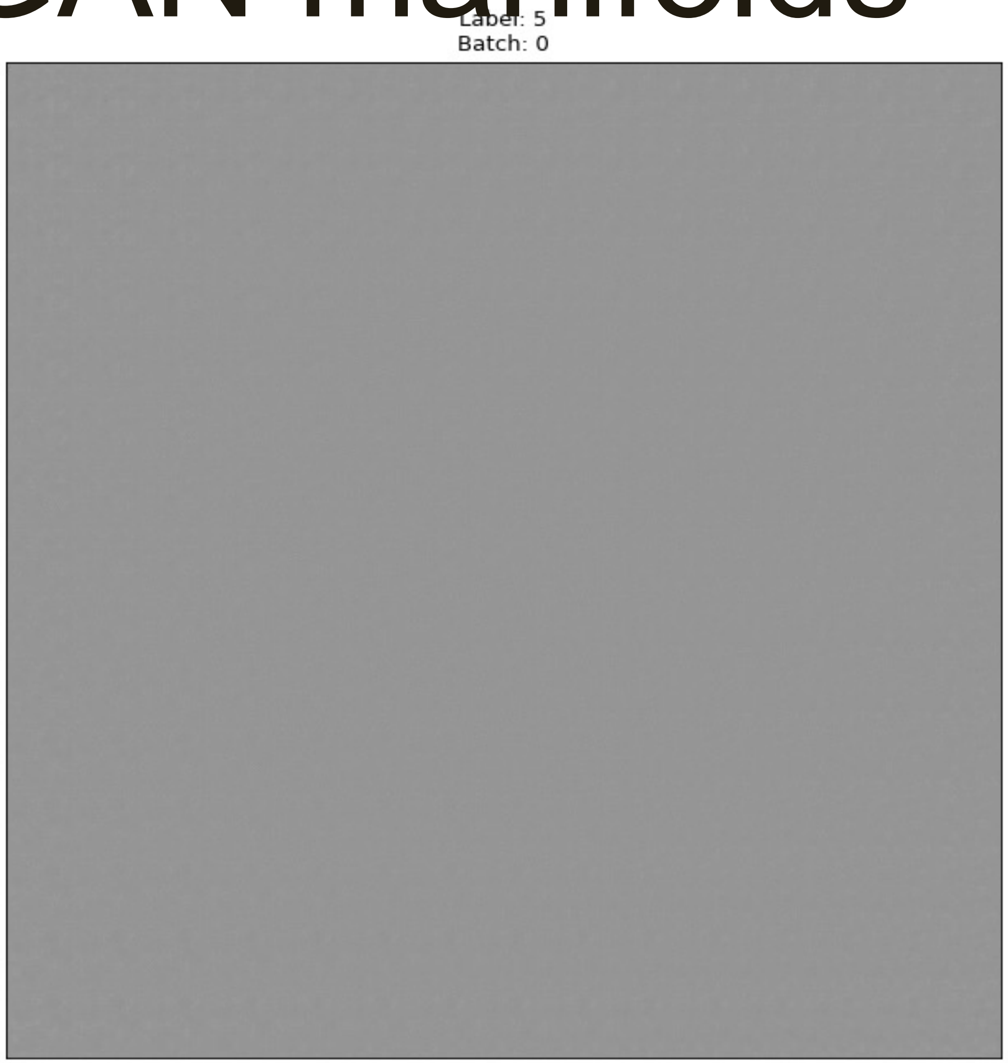


Image source: [Habr post on autoencoders and GANs]

cGAN manifolds



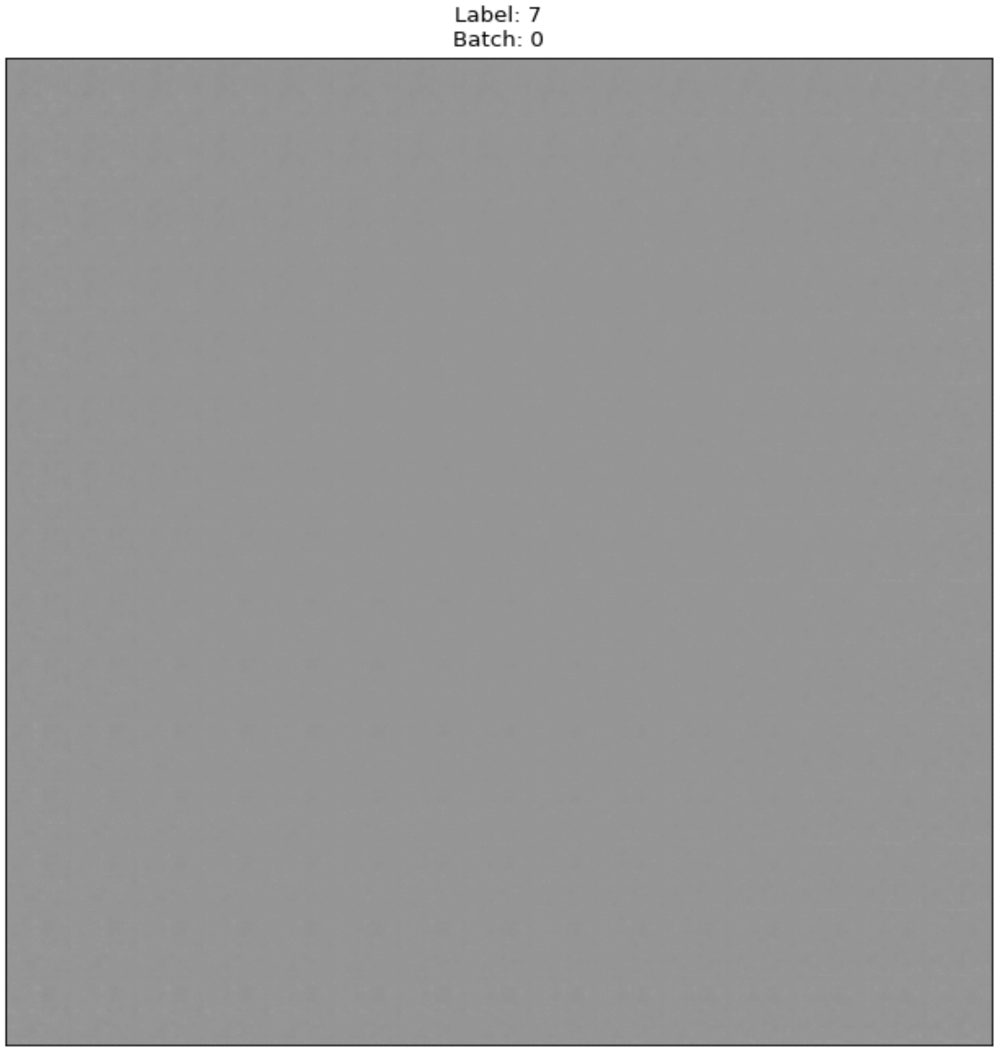


Image source: [Habr post on autoencoders and GANs]

Some more combinations

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

Simple GAN

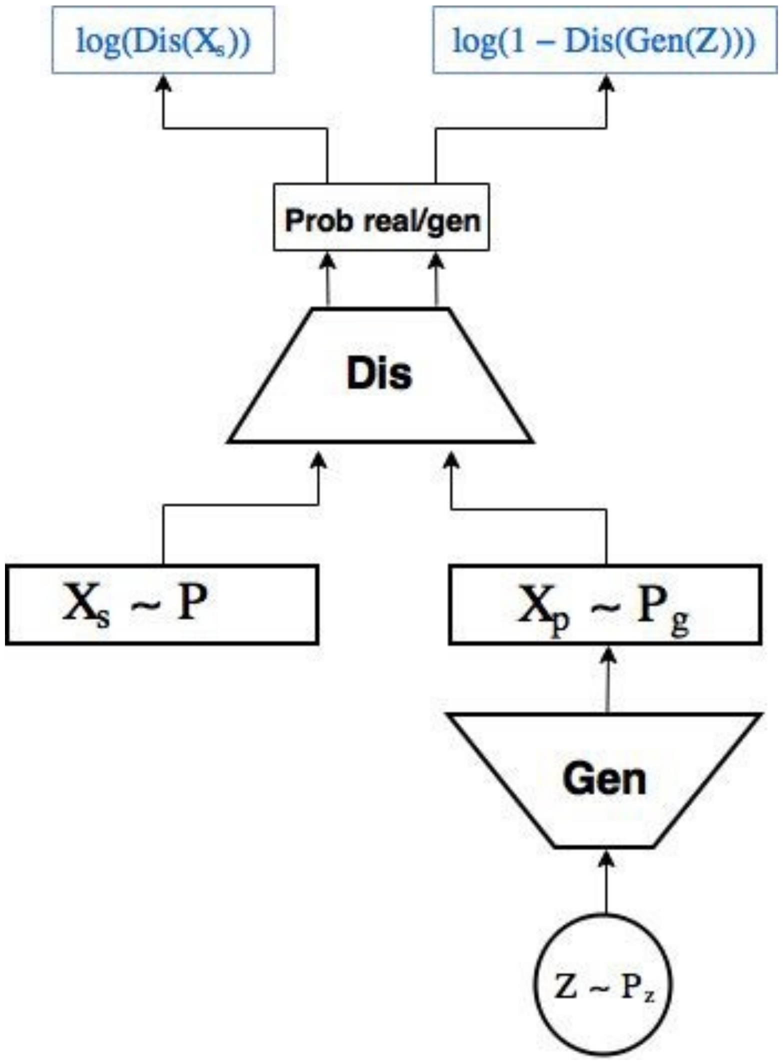
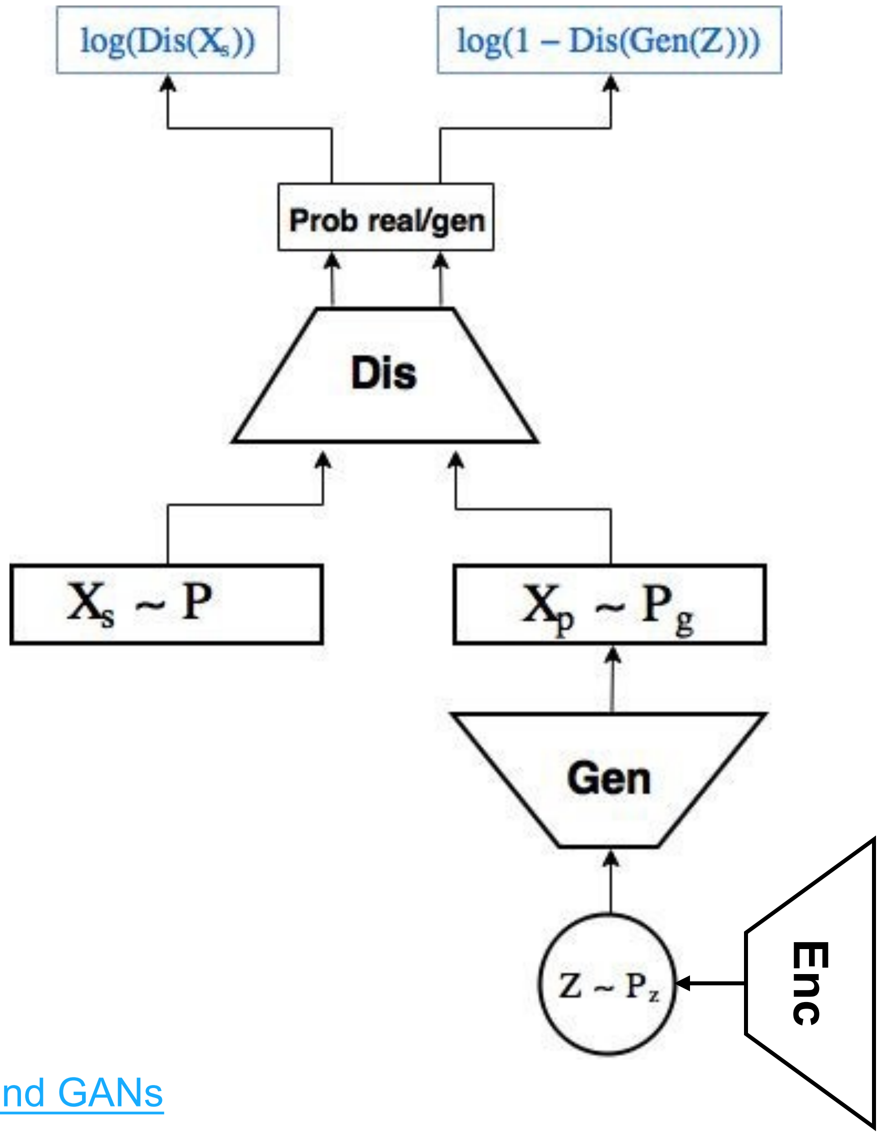


Image source: [Habr post on autoencoders and GANs]

Simple GAN



Pretraining

Image source: [Habr post on autoencoders and GANs]

VAE/GAN original illustration

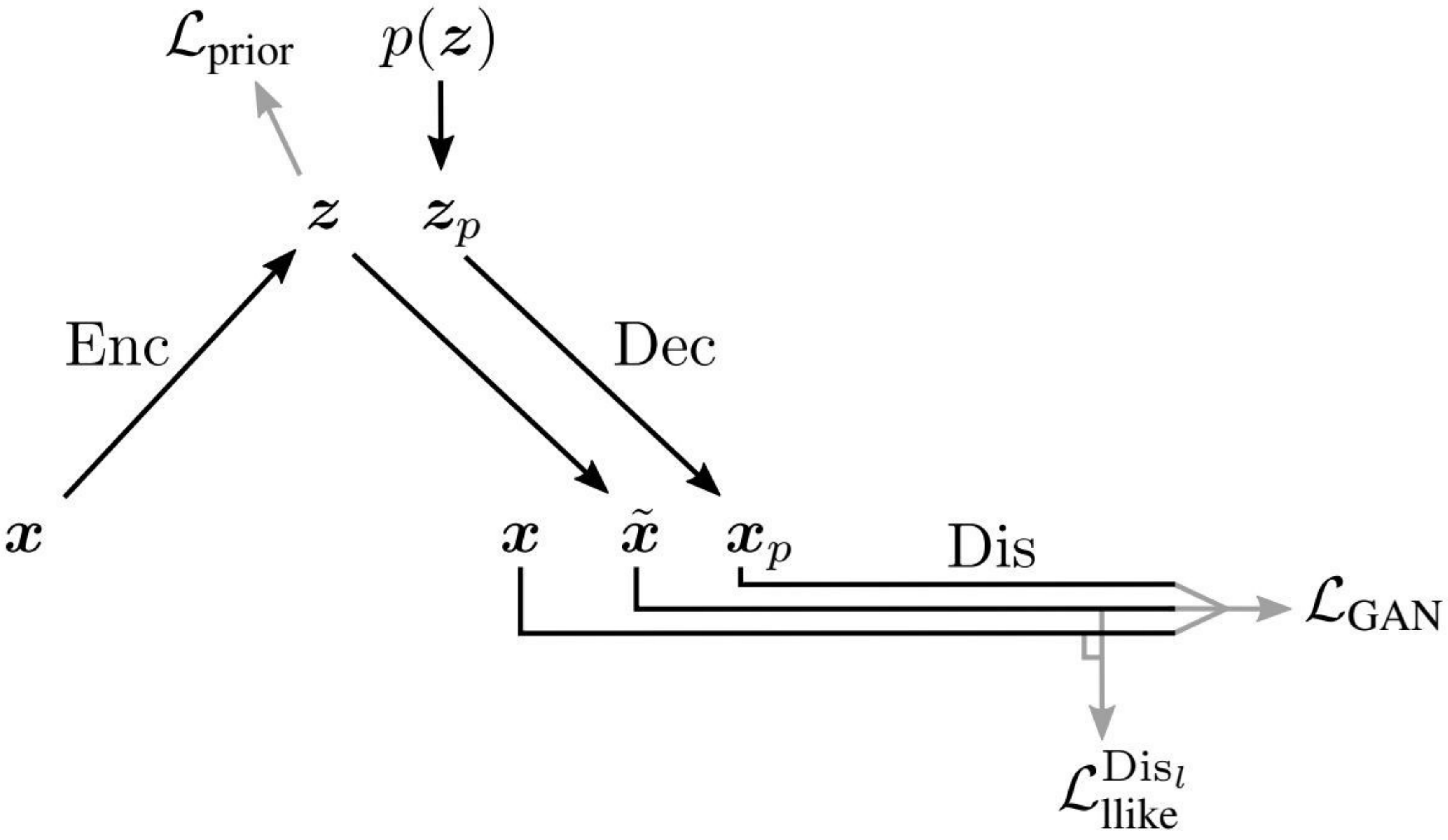
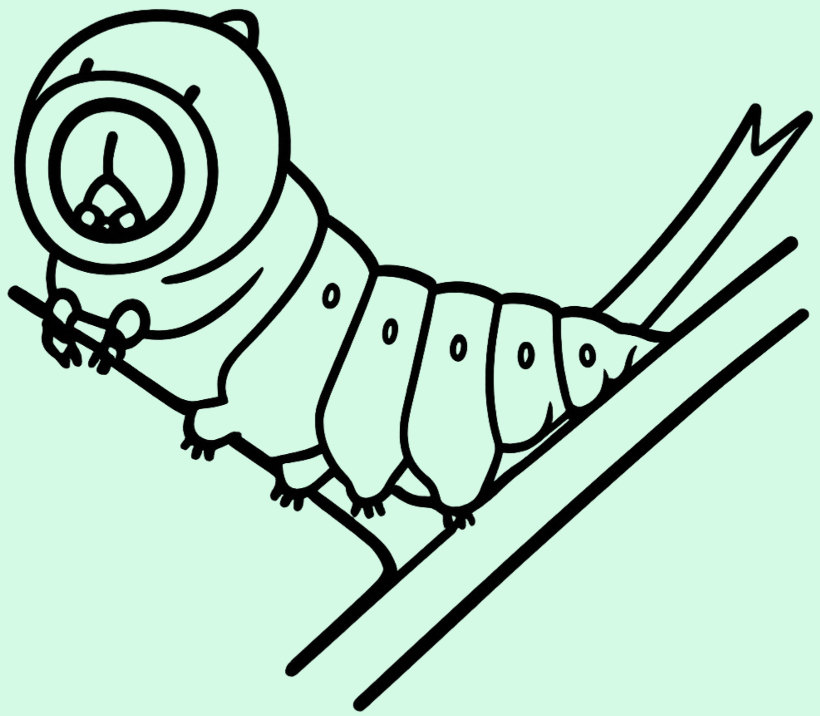


Image source: [Autoencoding beyond pixels using a learned similarity metric, Anders Boesen Lindbo Larsen et al, 2016]

Knowledge distillation



04

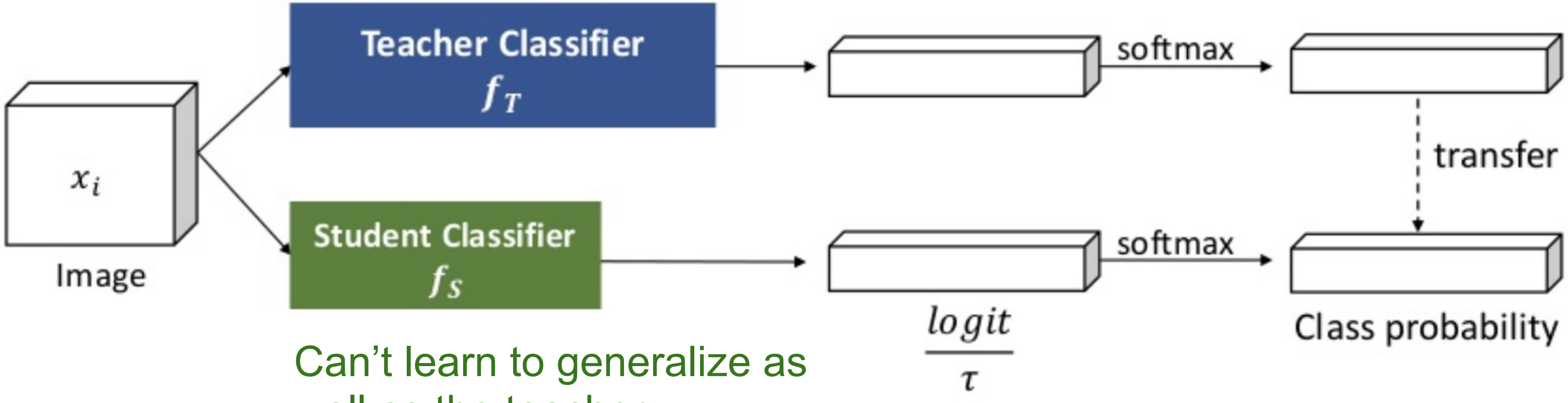
Knowledge distillation

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

Learns to generalize and
perform well



Can’t learn to generalize as
well as the teacher,
but still can perform well

Image 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:

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

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:

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

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

Source: [Distilling the Knowledge in a Neural Network]

Спасибо за внимание

Канал стажировок Яндекса

Доп. материалы