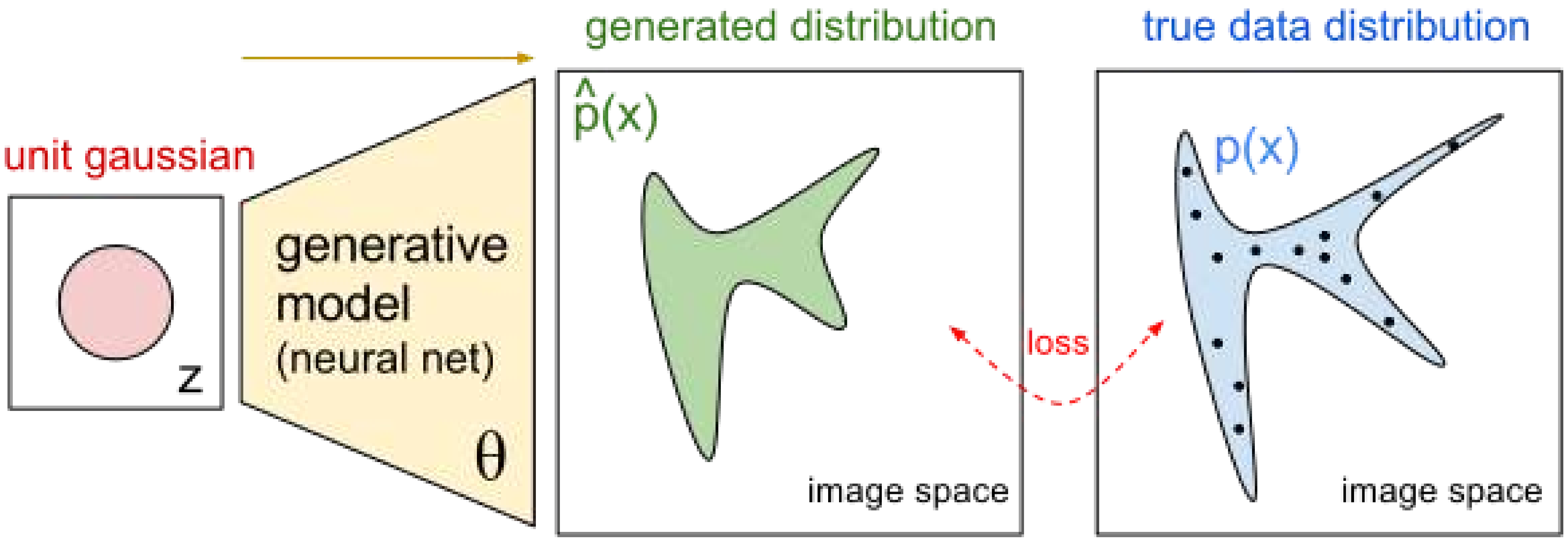
Генеративные модели для изображений

Что вообще есть генеративные модели?

"What I cannot create, I do not understand."

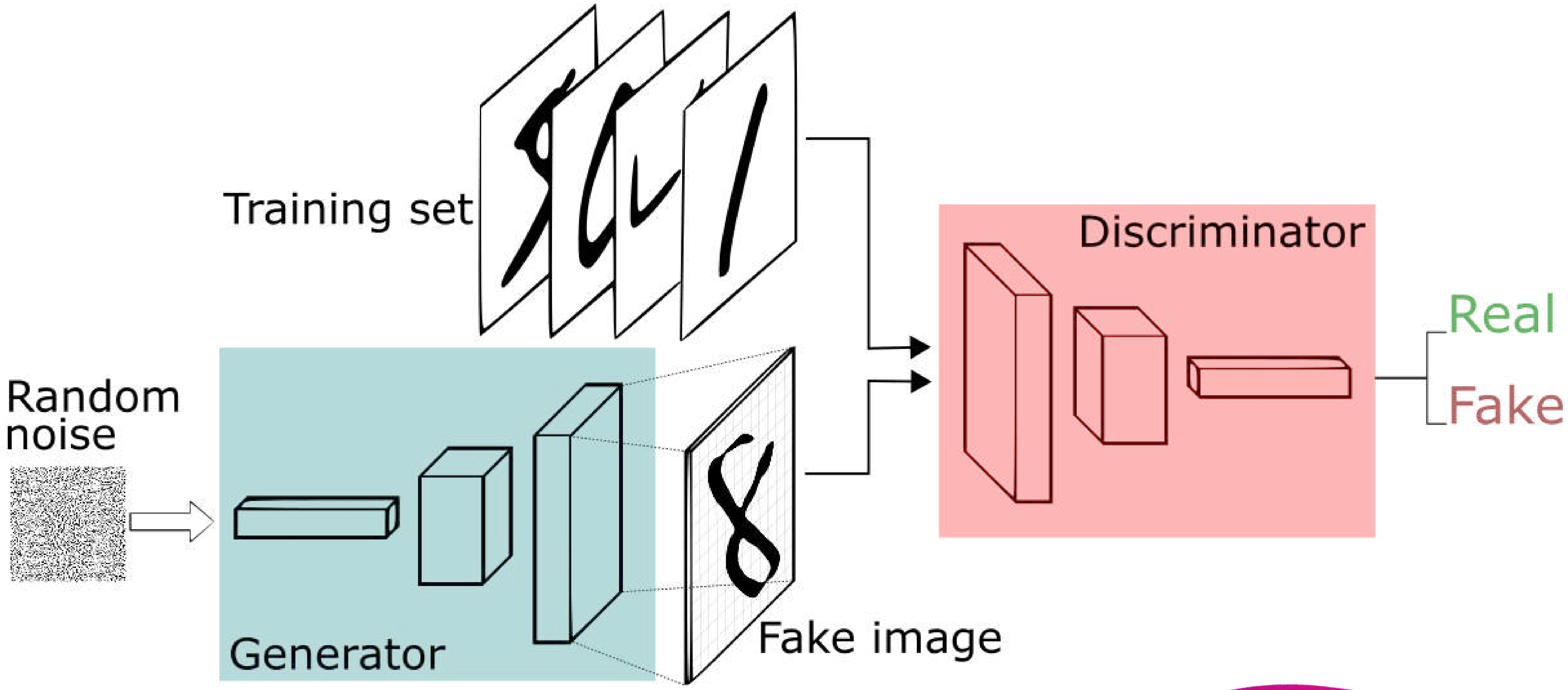
—Richard Feynman

Формулируем задачу



Generative Adversarial Networks

GAN



Training set
Discriminator
Real
Fake
Random noise
Generator
Fake image

Игра

- $p_{data}(x)$ - реальное распределение
- $p_z(x)$ - распределение нашего шума
- G - генератор
- D - дискриминатор
- $G(x)$ - выход нашего генератора, x - случайный шум
- $D(x)$ - выход нашего дискриминатора, $D(x) \in [0, 1]$, (1 - настоящая картинка, 0 - фейк)

Обучаем генератор, чтоб минимизировать

$$E_{z \in p_z(x)}(\log(1 - D(G(z)))$$

Обучаем дискриминатор, чтоб максимизировать

$$E_{x \in p_{data}(x)}[\log(D(x)] + E_{z \in p_z(x)}[\log(1 - D(G(z))]$$

Итого получаем minimax игру с такой функцией $V(G, D)$

$$\min_G \min_D V(D,G) = E_{x \in p_{data}(x)}[\log(D(x)] + E_{z \in p_z(x)}[\log(1 - D(G(z))]$$

Обучение GAN'ов

for number of training iterations **do**

for k steps **do**

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

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

end for

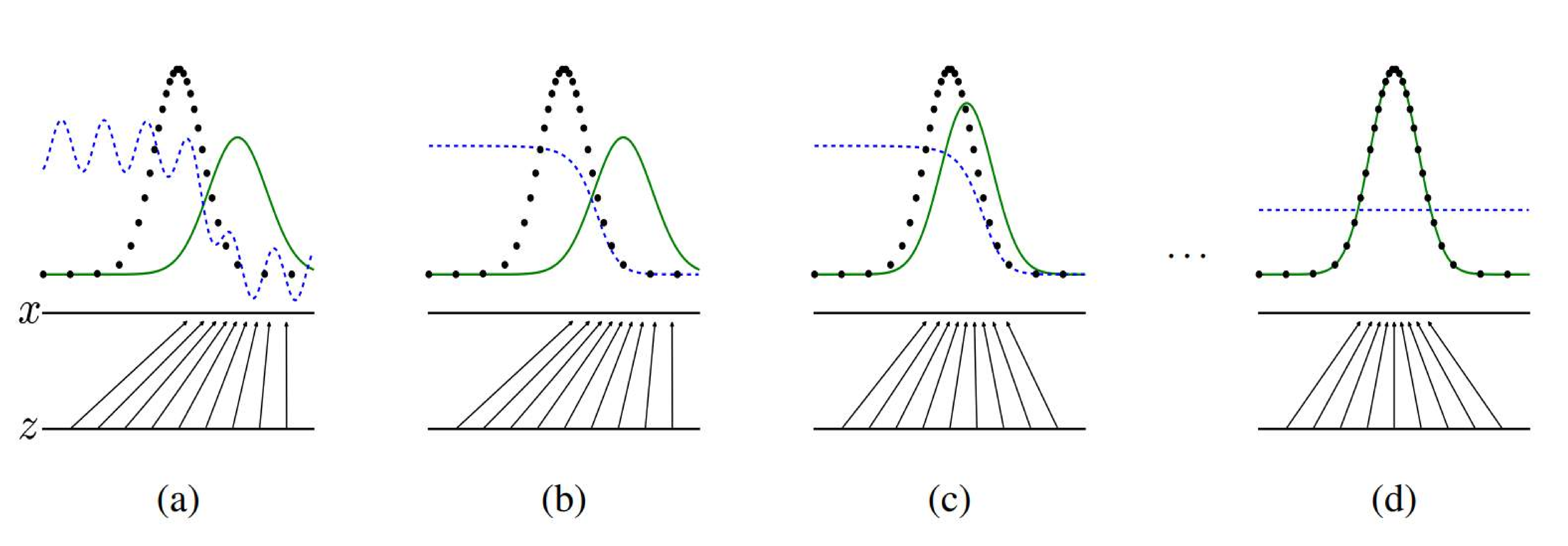
- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

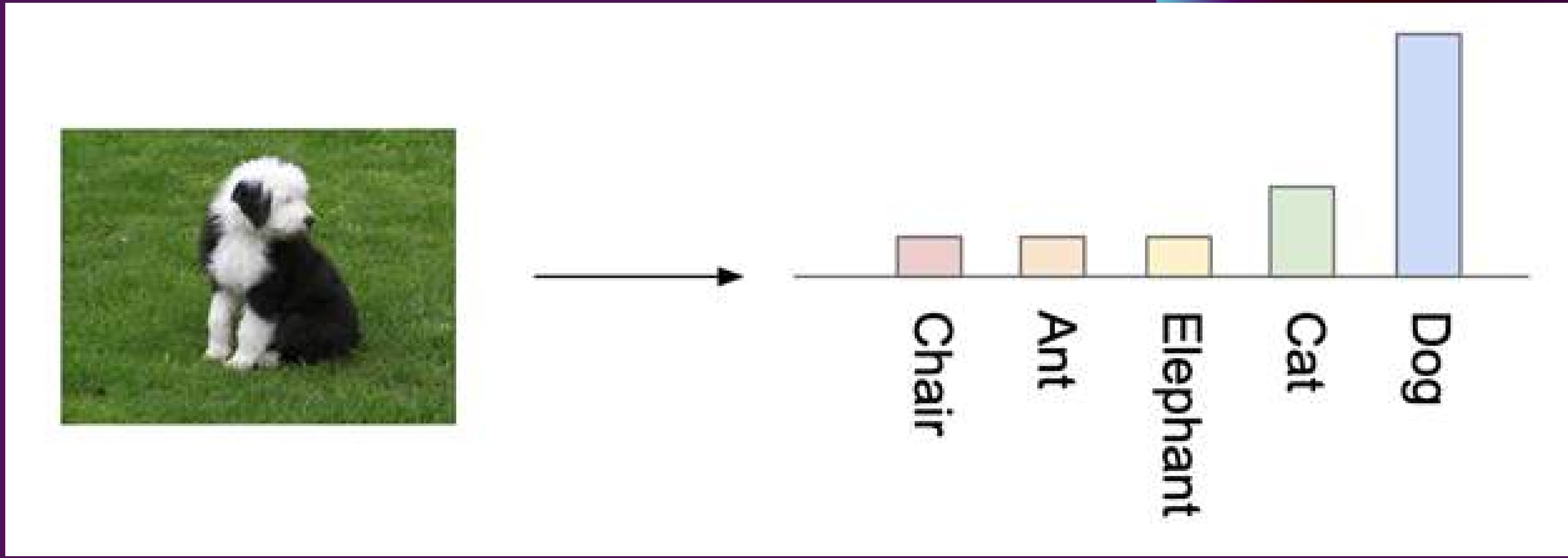
end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

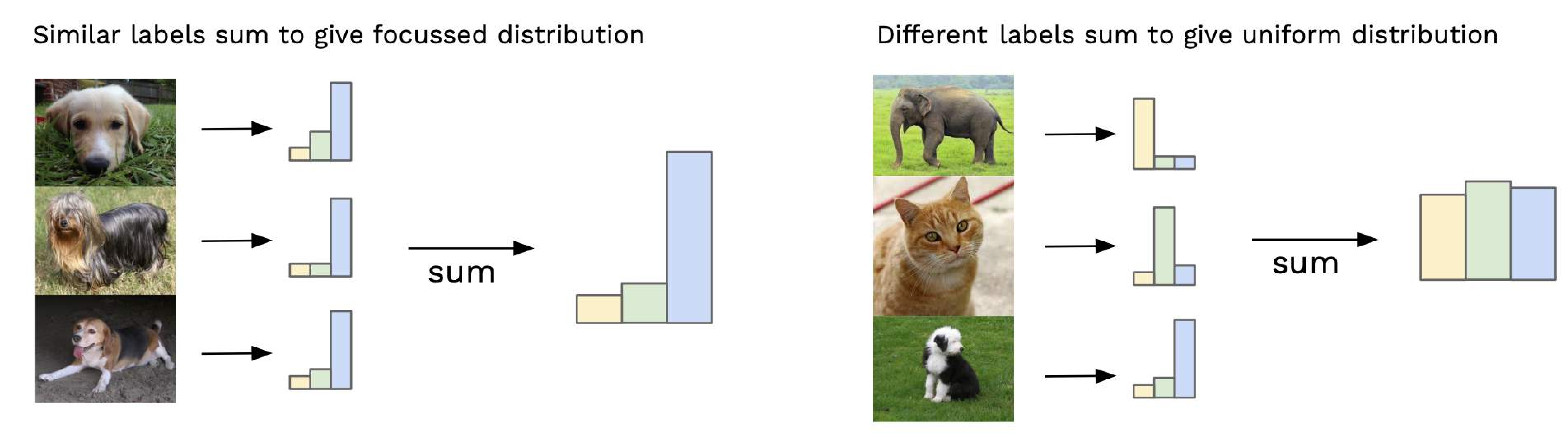
Обучение GAN'ов



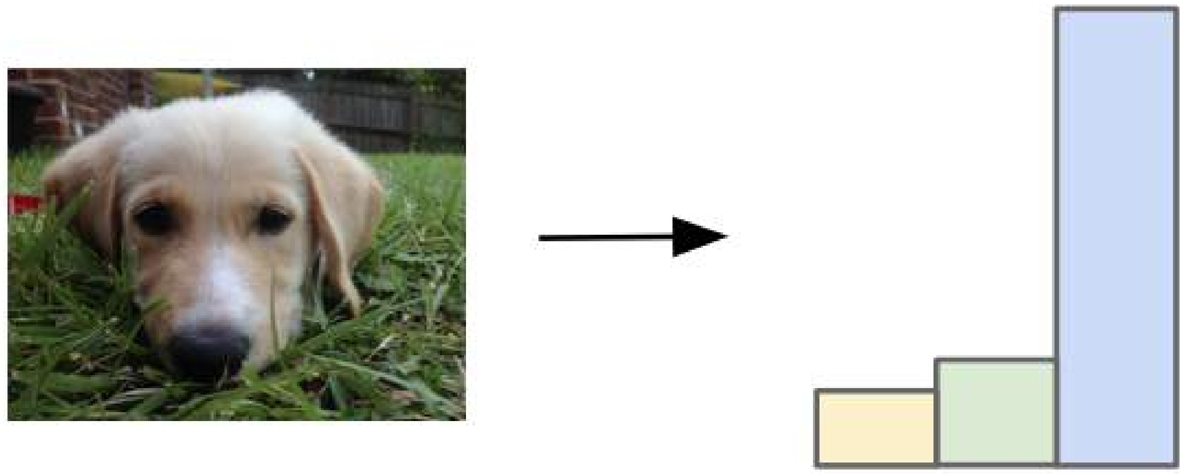
Inception Score



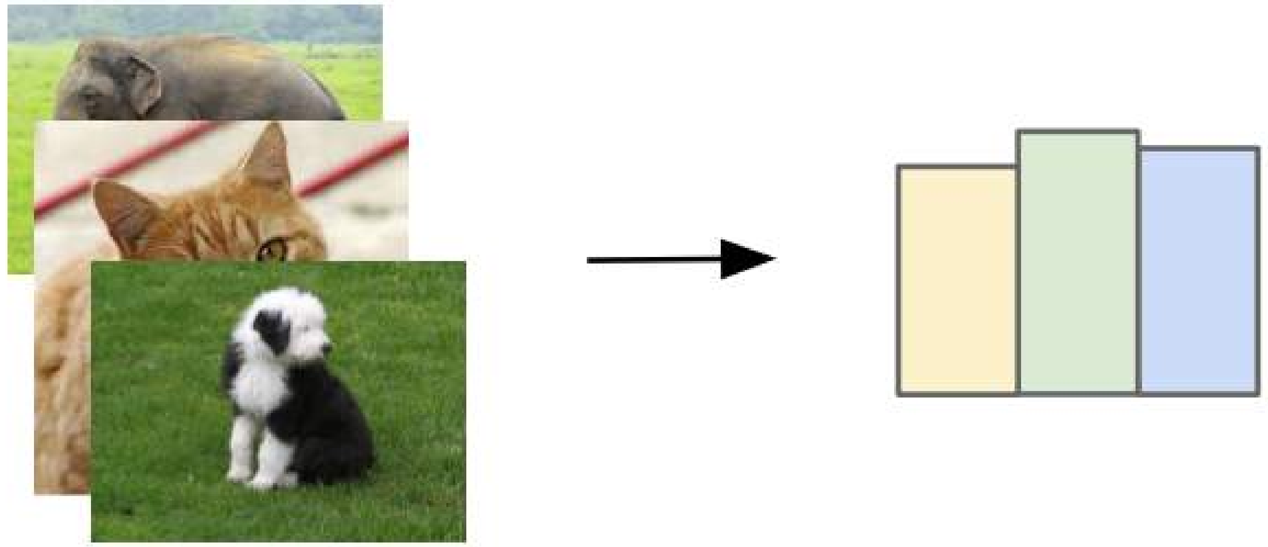
Inception Score



Inception Score

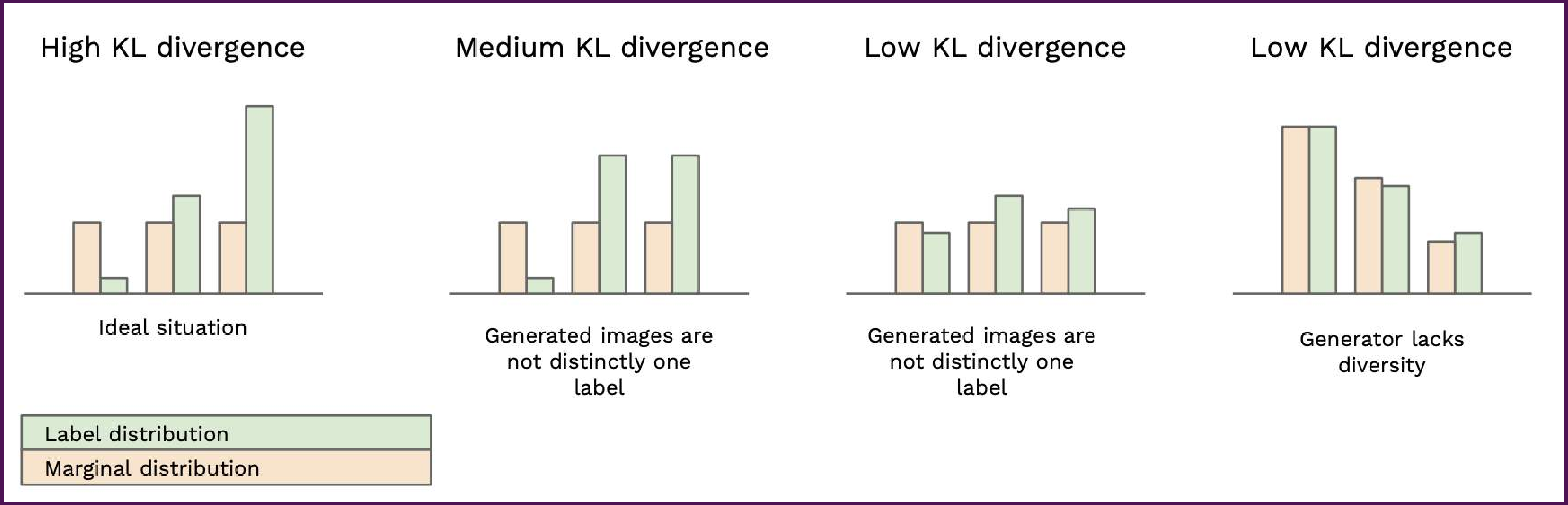


Ideal label distribution



Ideal marginal distribution

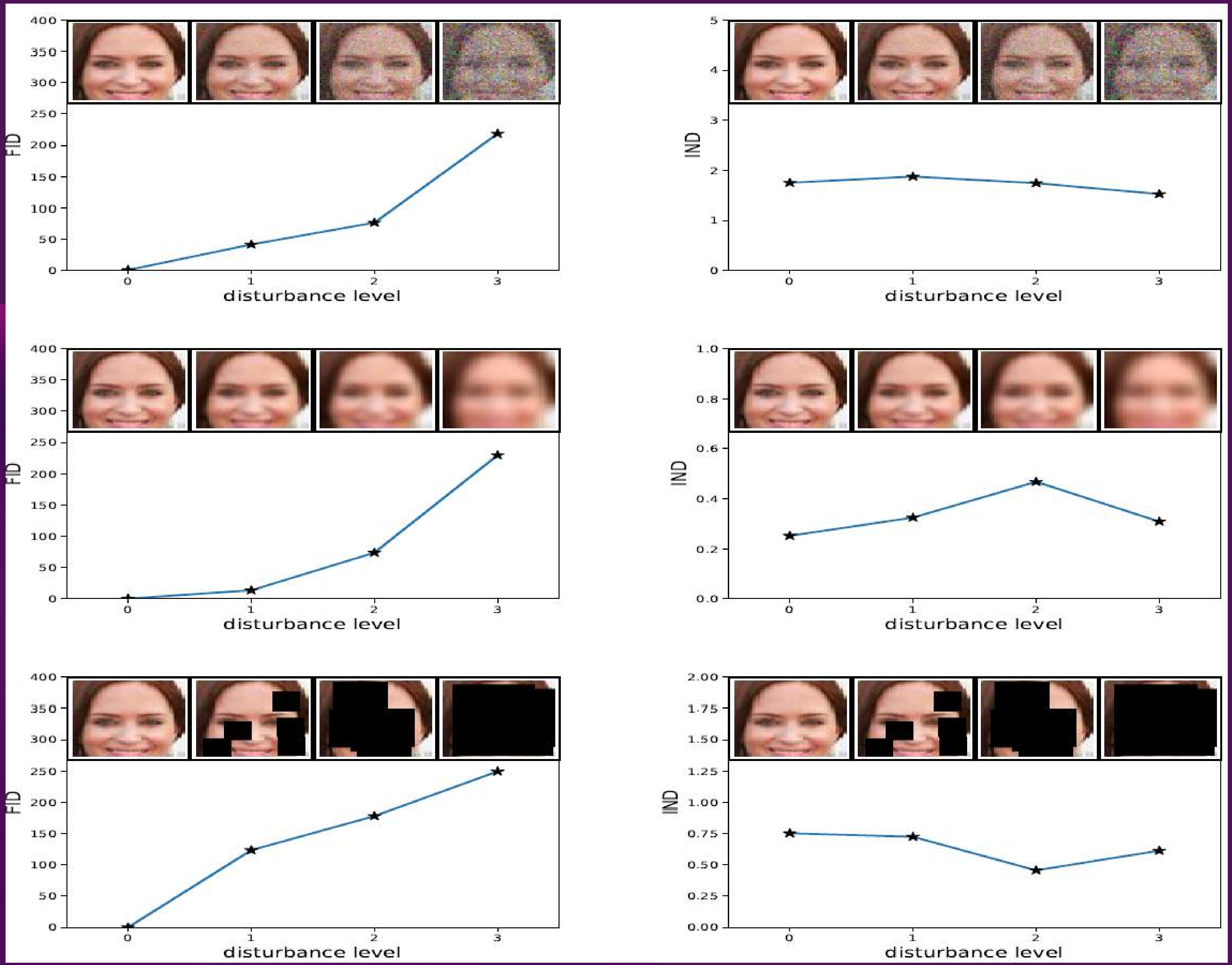
Inception Score

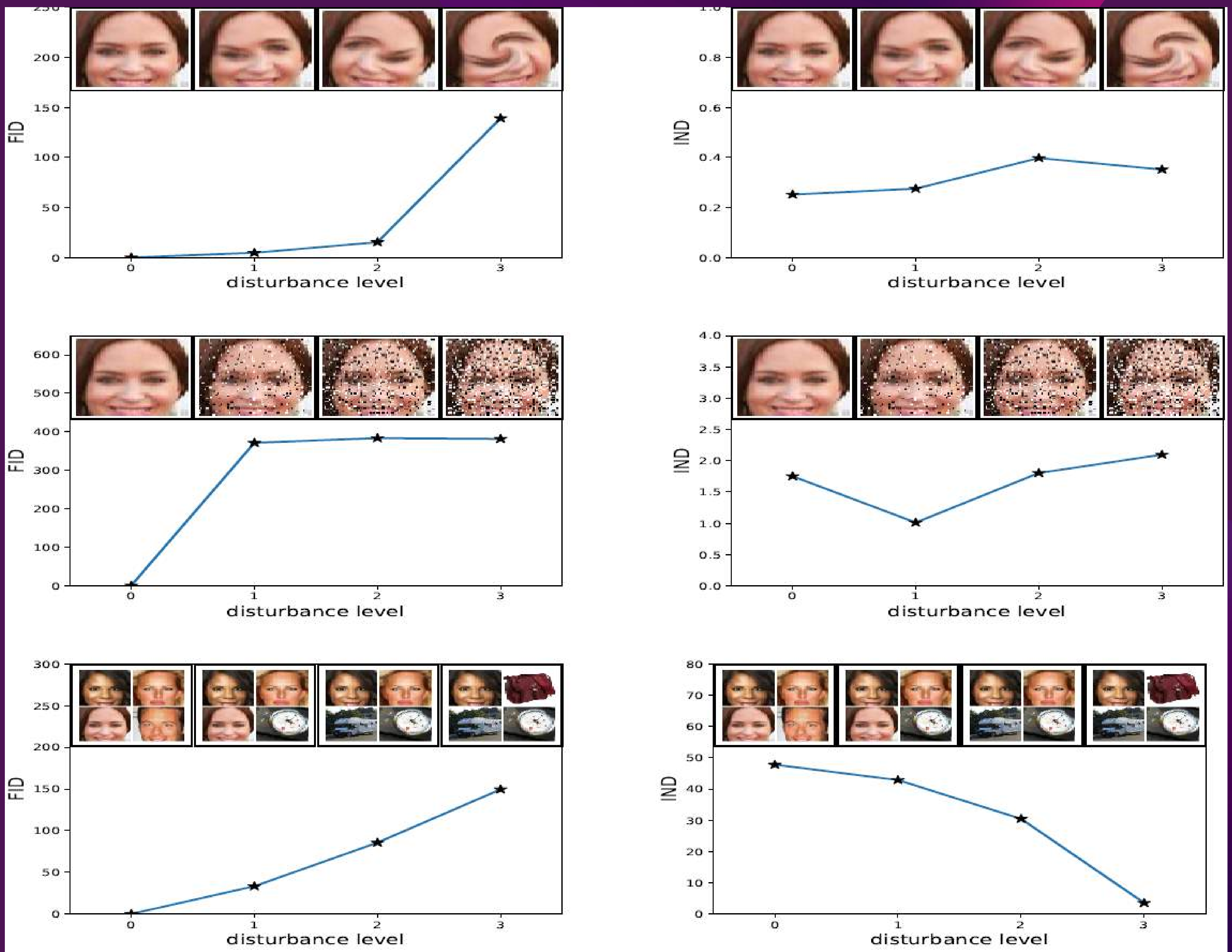


Fréchet Inception Distance

$$d^2((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_w, \boldsymbol{C}_w)) = \|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2 + \mathrm{Tr}\big(\boldsymbol{C} + \boldsymbol{C}_w - 2(\boldsymbol{C}\boldsymbol{C}_w)^{1/2}\big) \, .$$

FID vs IS

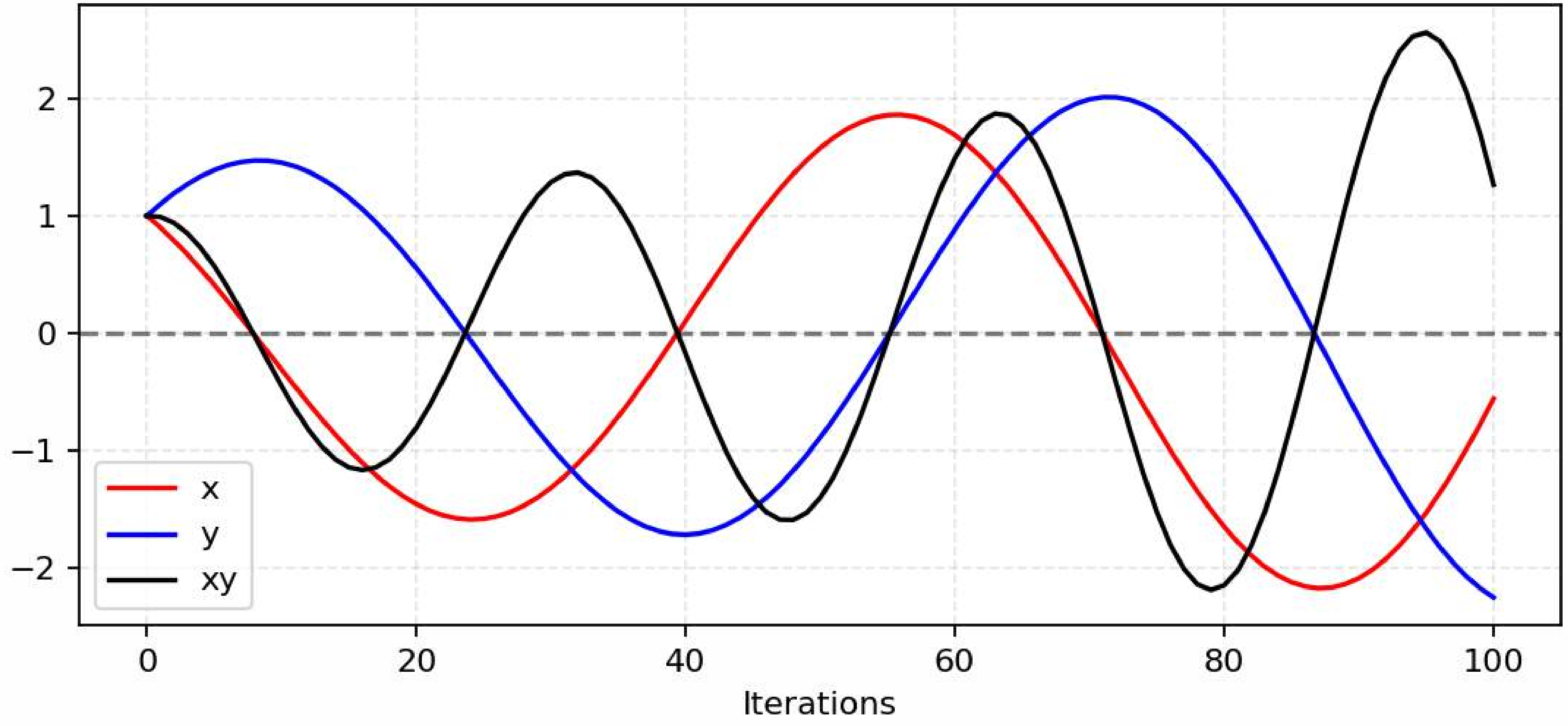




Применение GAN'ам

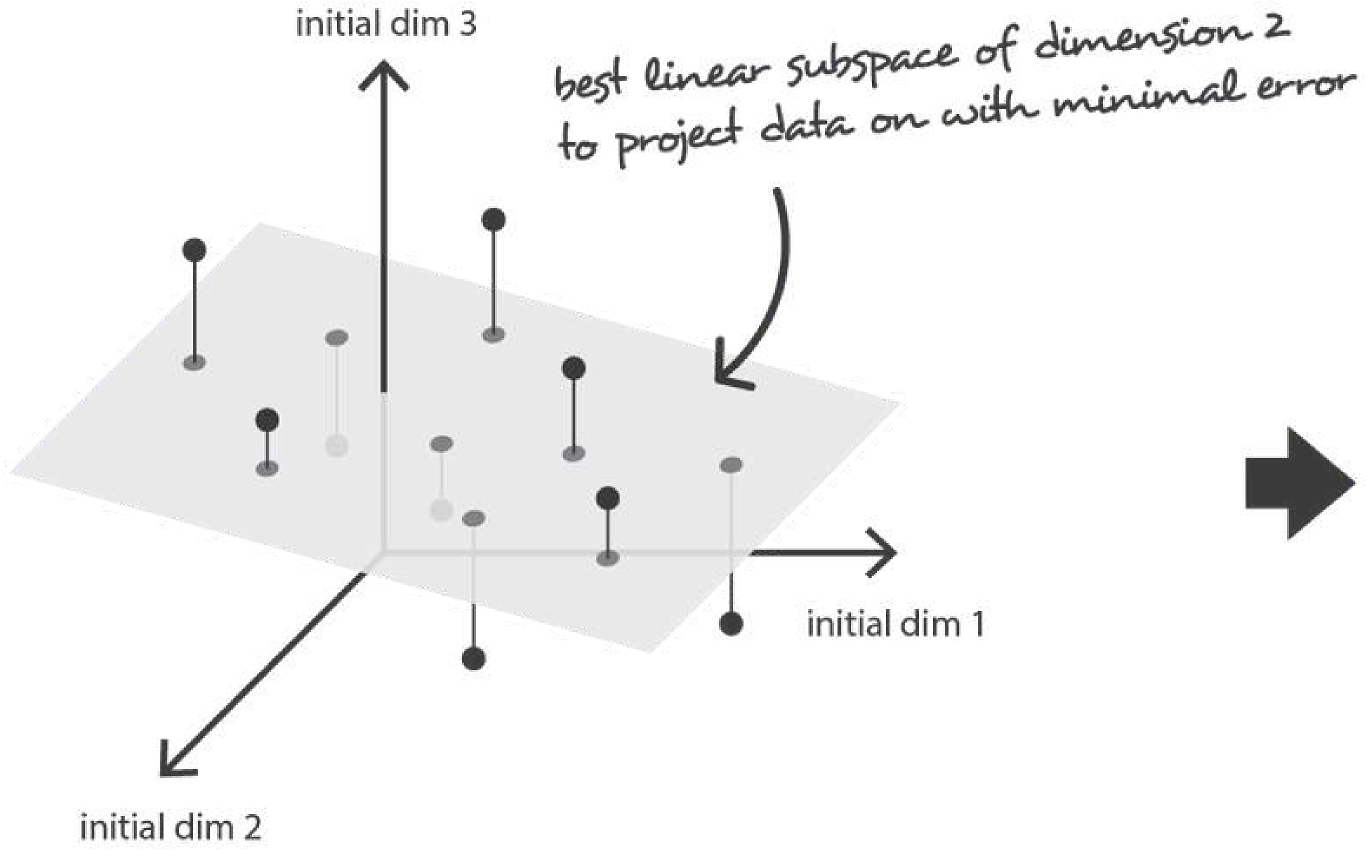
- Generate Examples for Image Datasets
- Generate Photographs of Human Faces
- Generate Realistic Photographs
- Generate Cartoon Characters
- Image-to-Image Translation
- Text-to-Image Translation
- Semantic-Image-to-Photo Translation
- Face Frontal View Generation
- Generate New Human Poses
- Photos to Emojis
- Photograph Editing
- Face Aging
- Photo Blending
- Super Resolution
- Photo Inpainting
- Clothing Translation
- Video Prediction
- 3D Object Generation

Проблемы GAN'ов



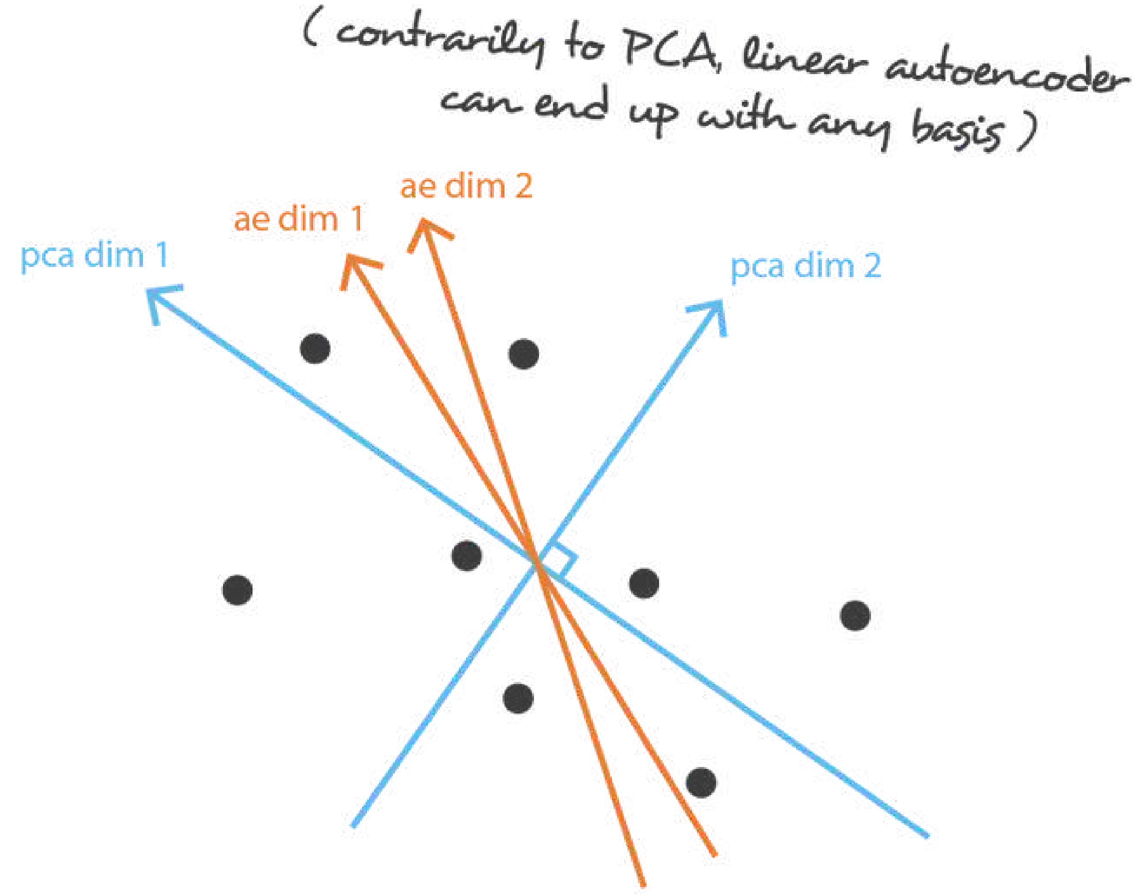
Variational Autoencoders

Задача уменьшения размерности



Data in the full initial space

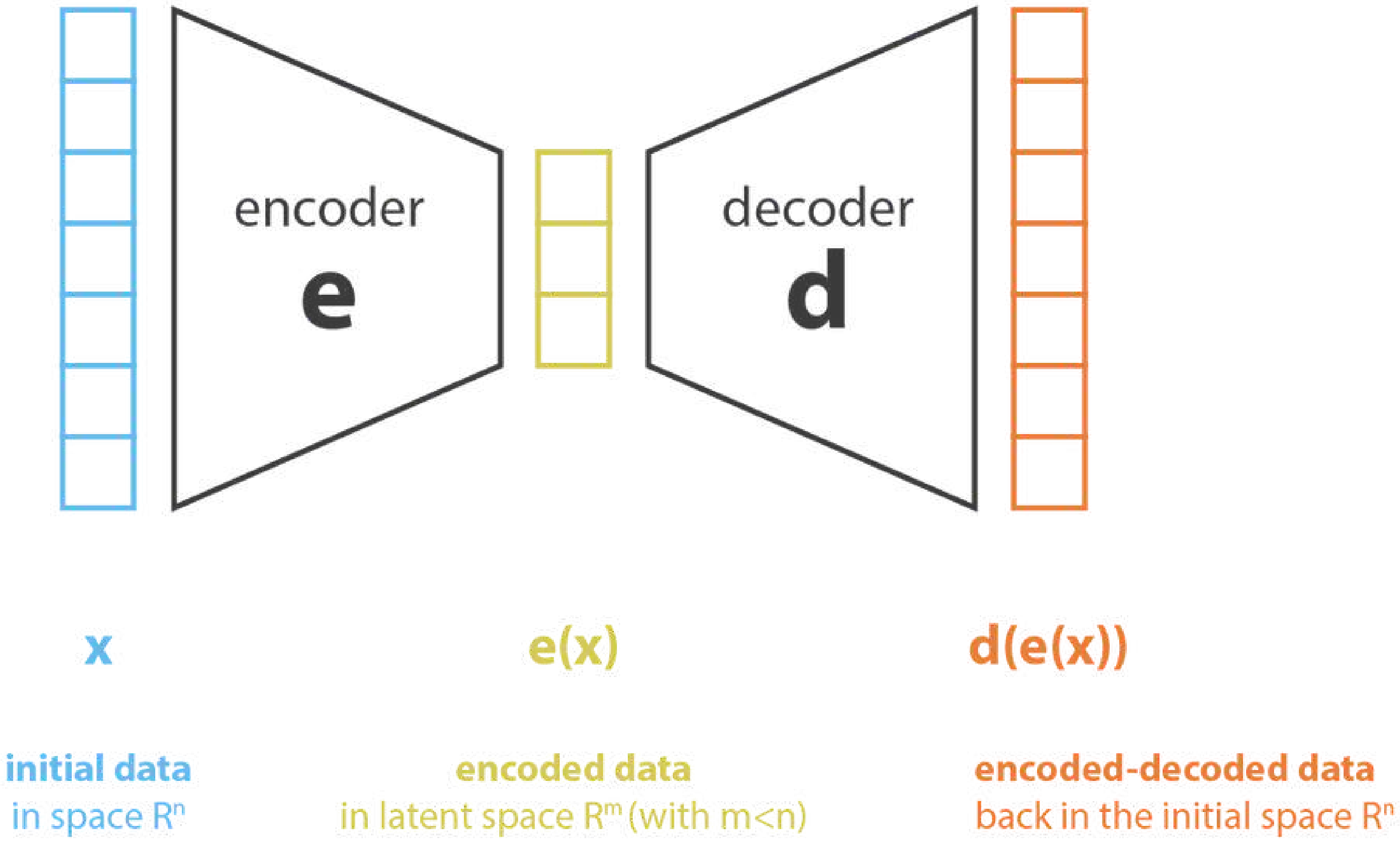
In order to reduce dimensionality, PCA and linear autoencoder target, in theory, the same optimal subspace to project data on...



Data projected on the best linear subspace

... but not necessarily with the same basis due to different constraints (in PCA the first component is the one that explains the maximum of variance and components are orthogonal)

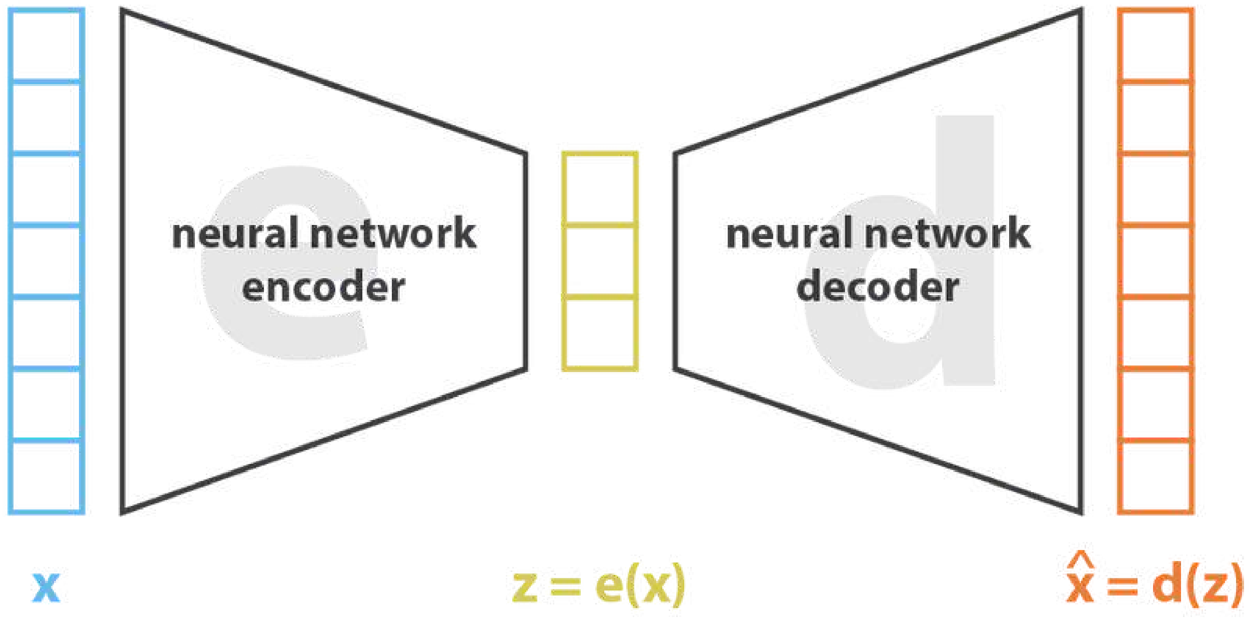
Автоэнкодер



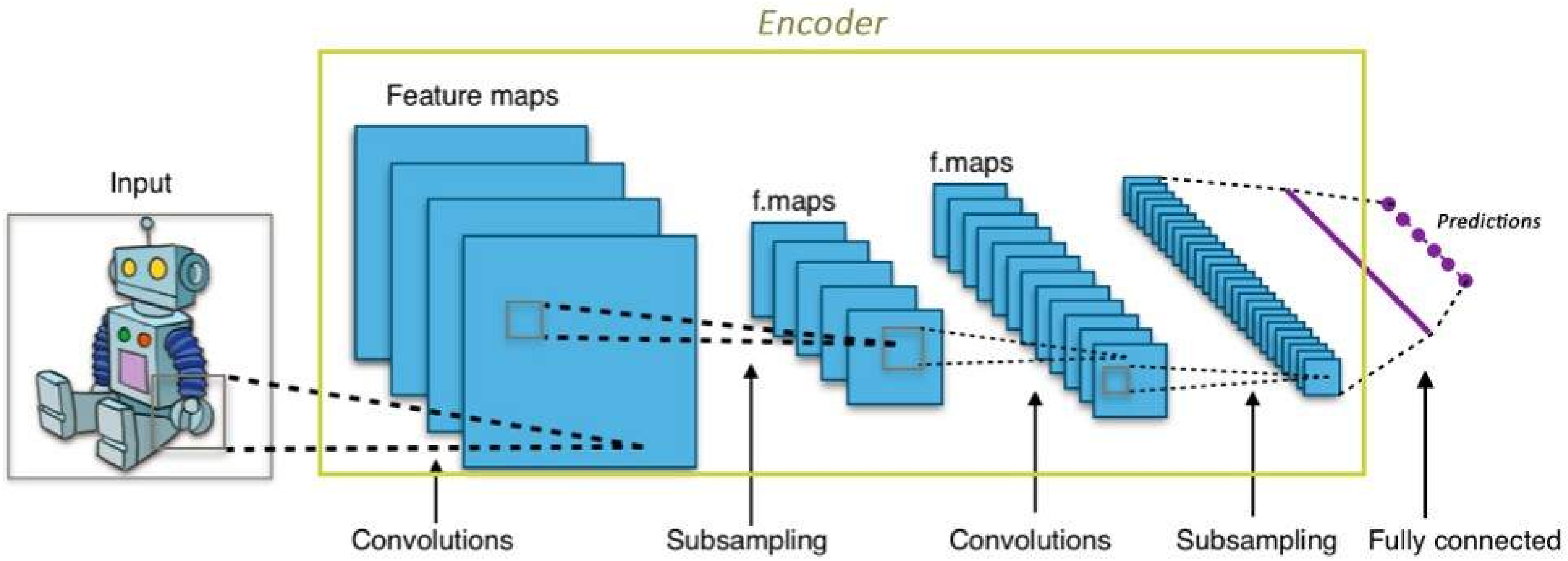
$x = d(e(x))$ ➡ **lossless encoding**
no information is lost when reducing the number of dimensions

$x \neq d(e(x))$ ➡ **lossy encoding**
some information is lost when reducing the number of dimensions and can't be recovered later

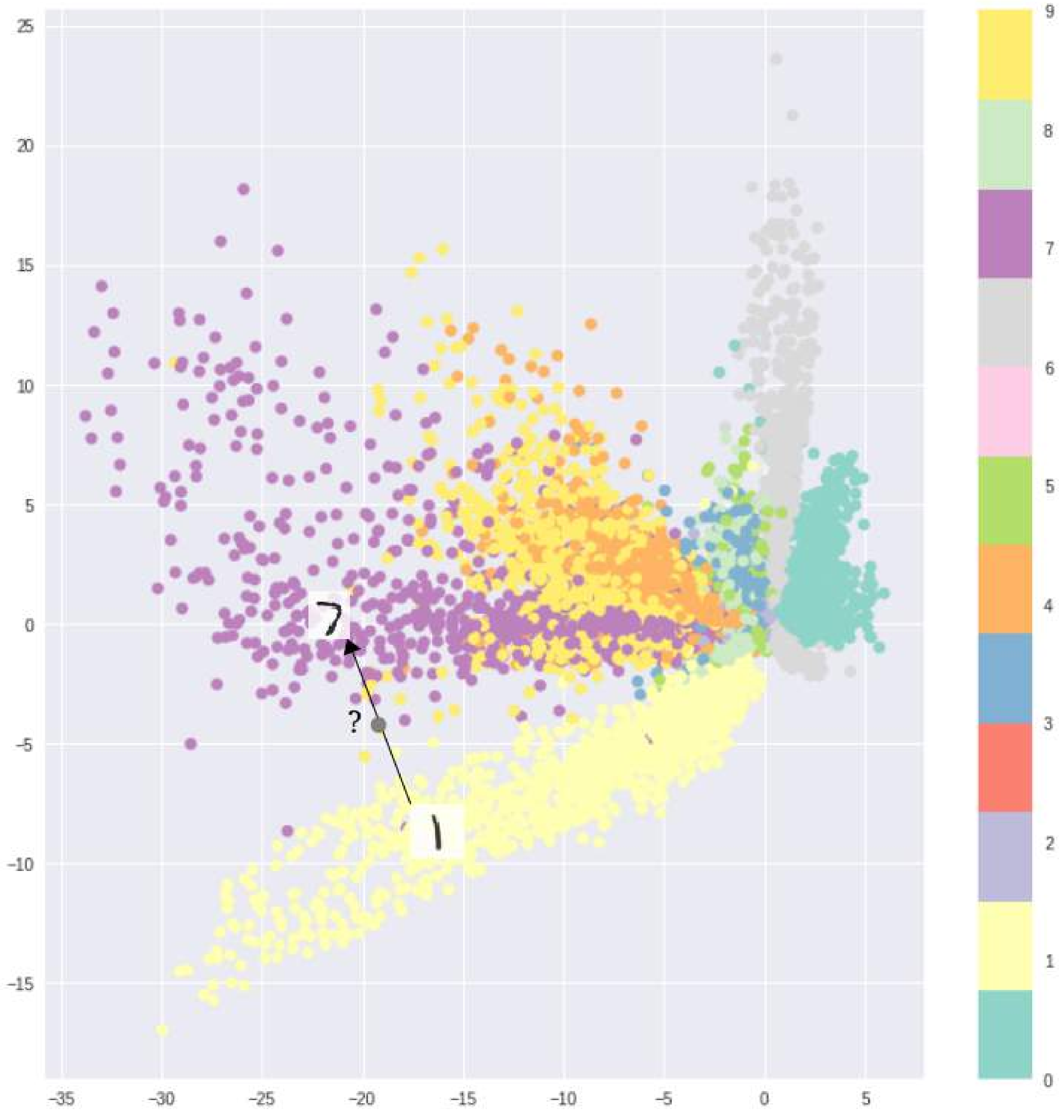
Автоэнкодер на основе нейронной сети



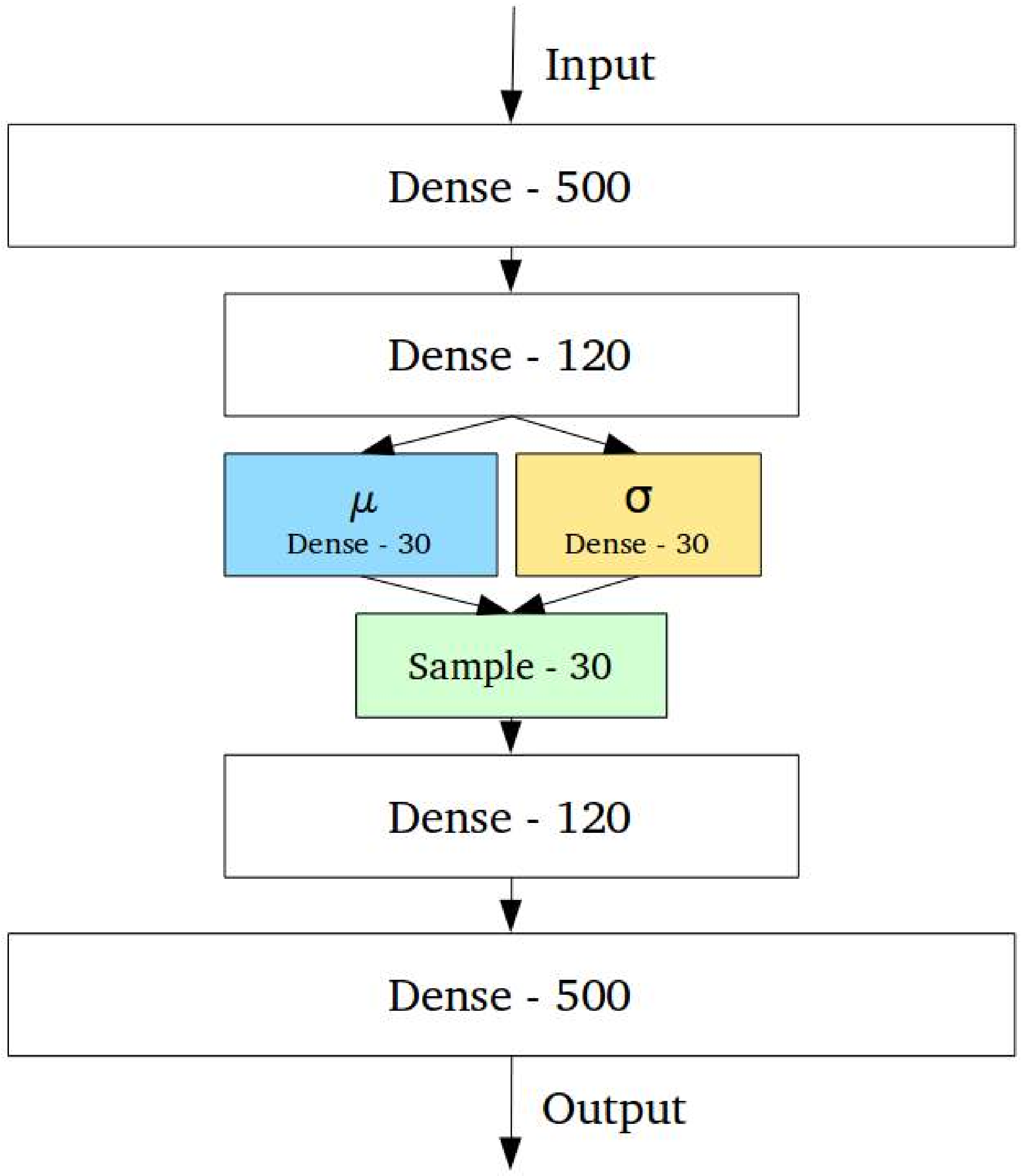
$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

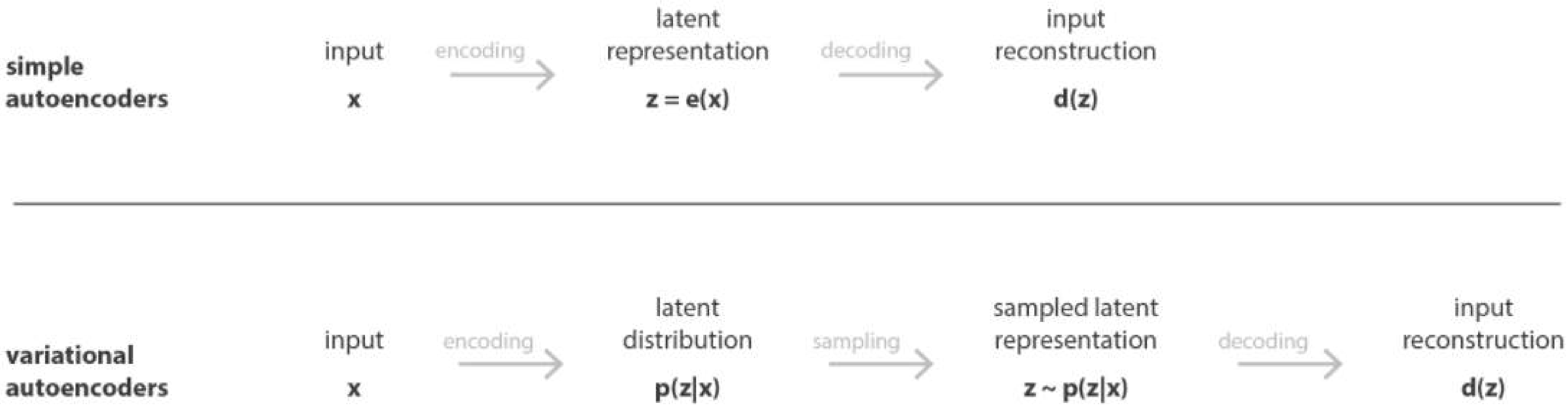


Визуализация двумерного скрытого пространства

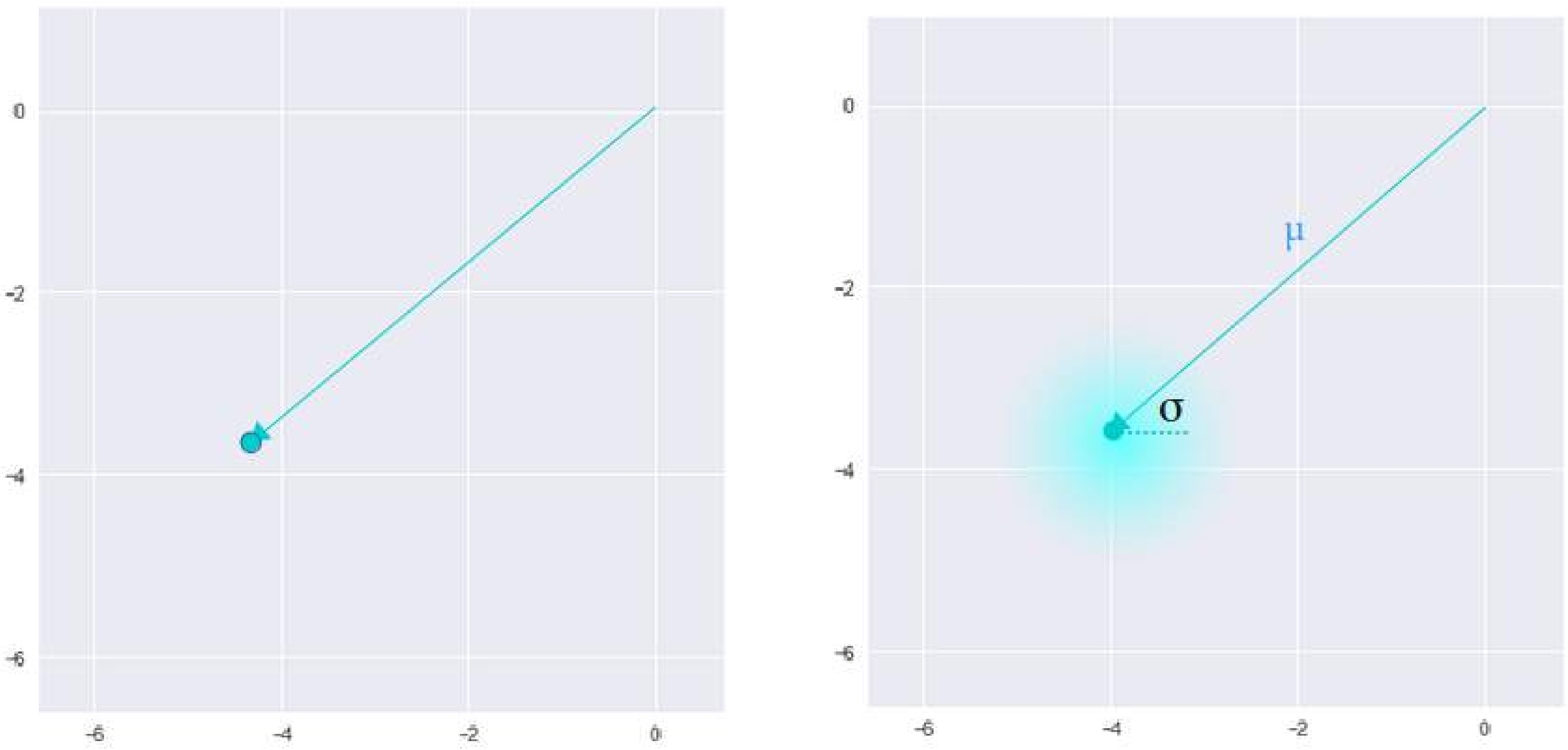


Вариационный автоэнкодер





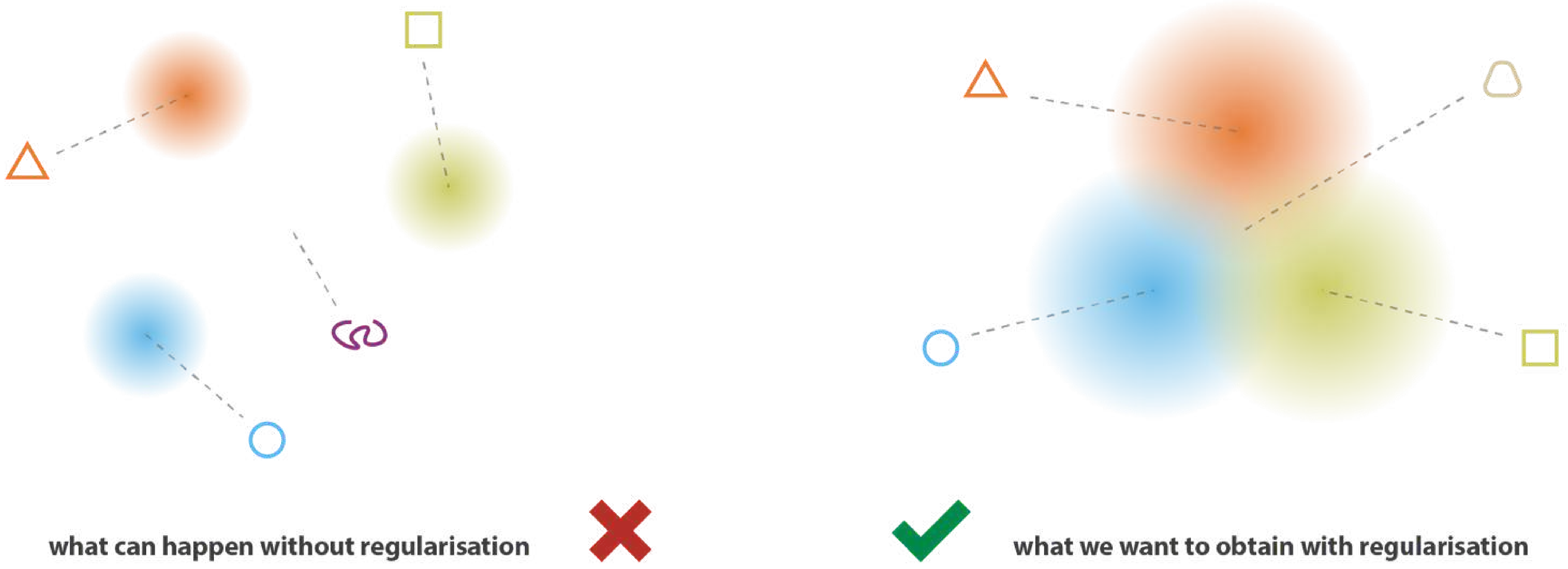
Преимущество VAE



Standard Autoencoder
(direct encoding coordinates)

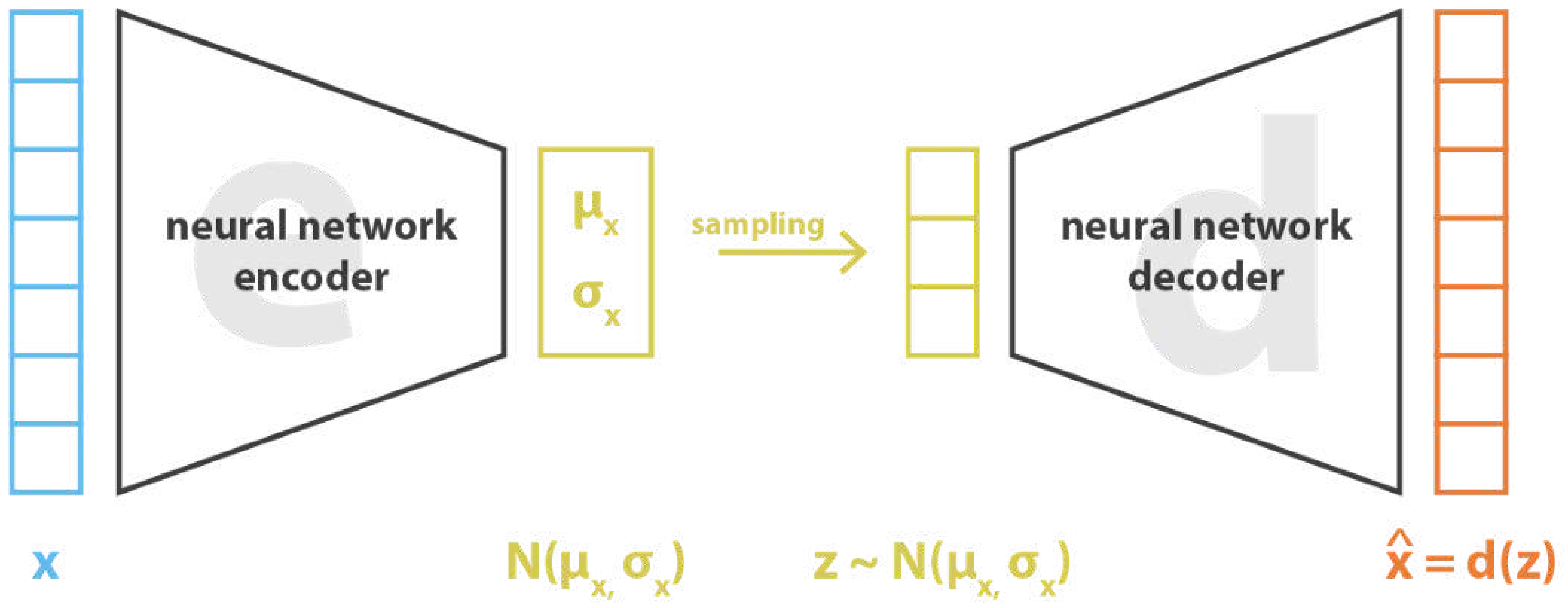
Variational Autoencoder
(μ and σ initialize a probability distribution)

На проблему напали...



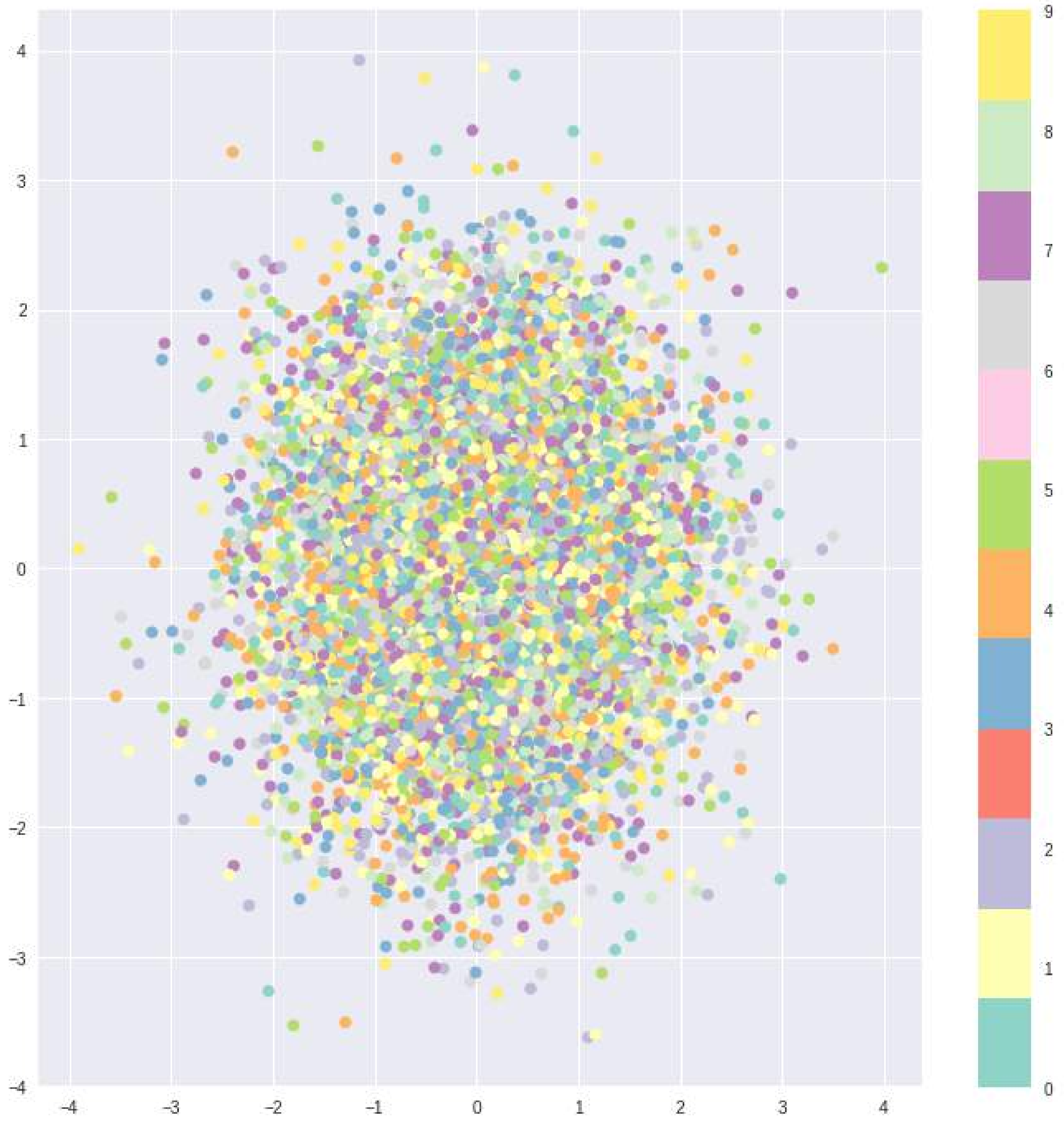
KL-дивергенция

$$\sum_{i=1}^{n} \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1$$

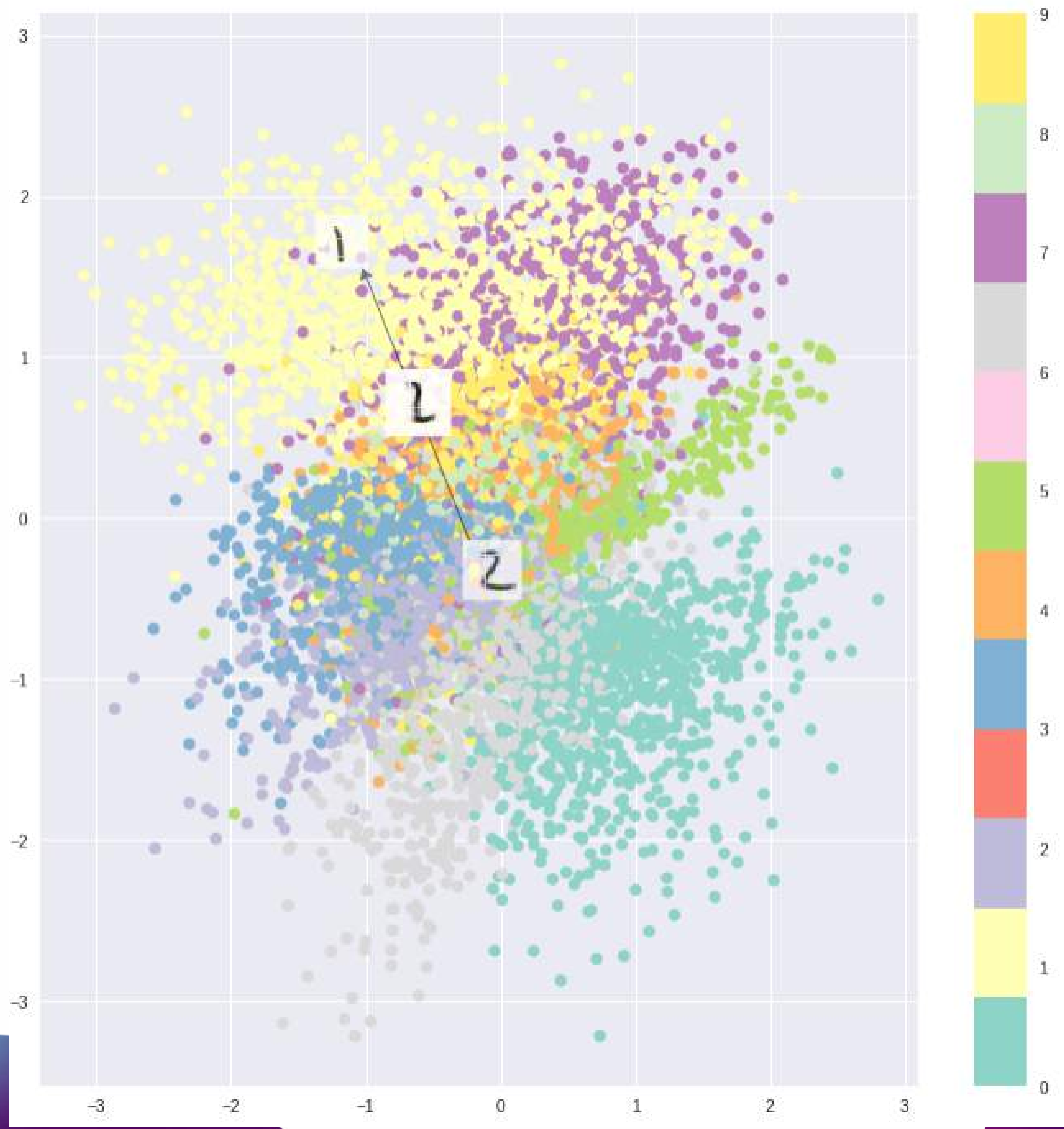


$$\text{loss} = \| x - \hat{x} \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,] = \| x - d(z) \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,]$$

Итог



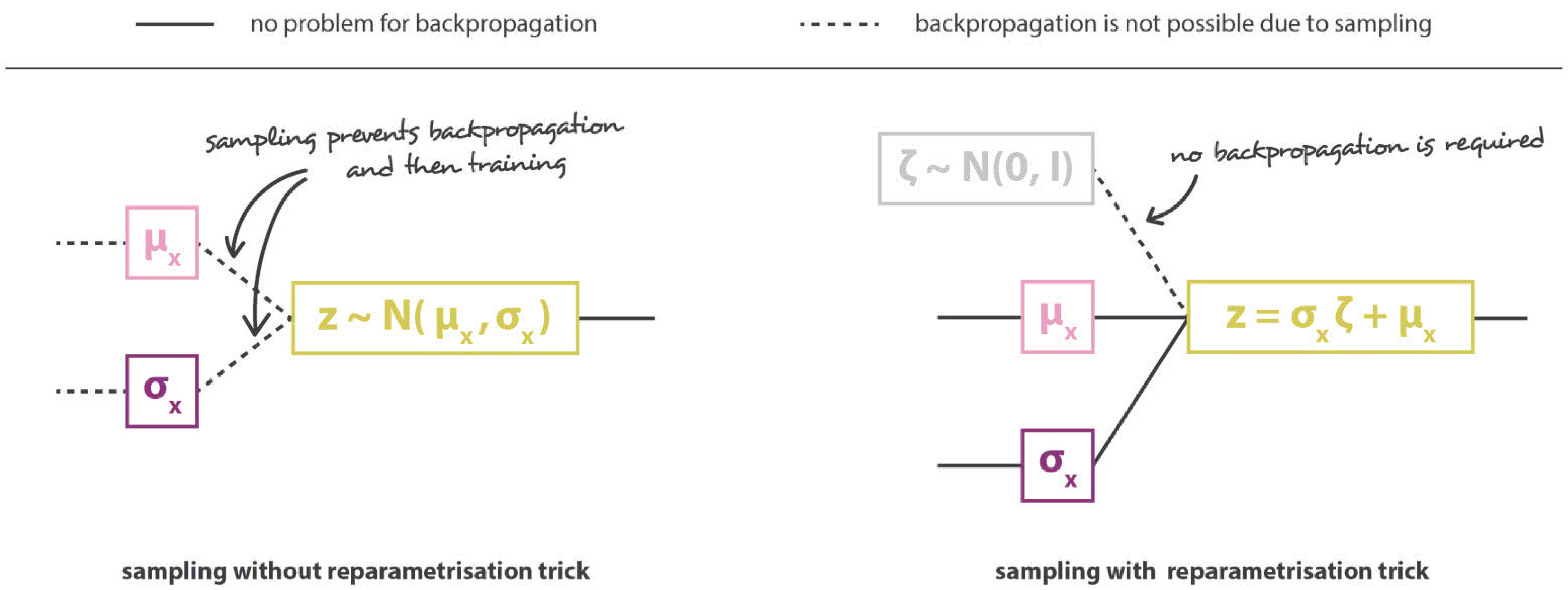
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4
9
8
7
6
5
4
3
2
1
0



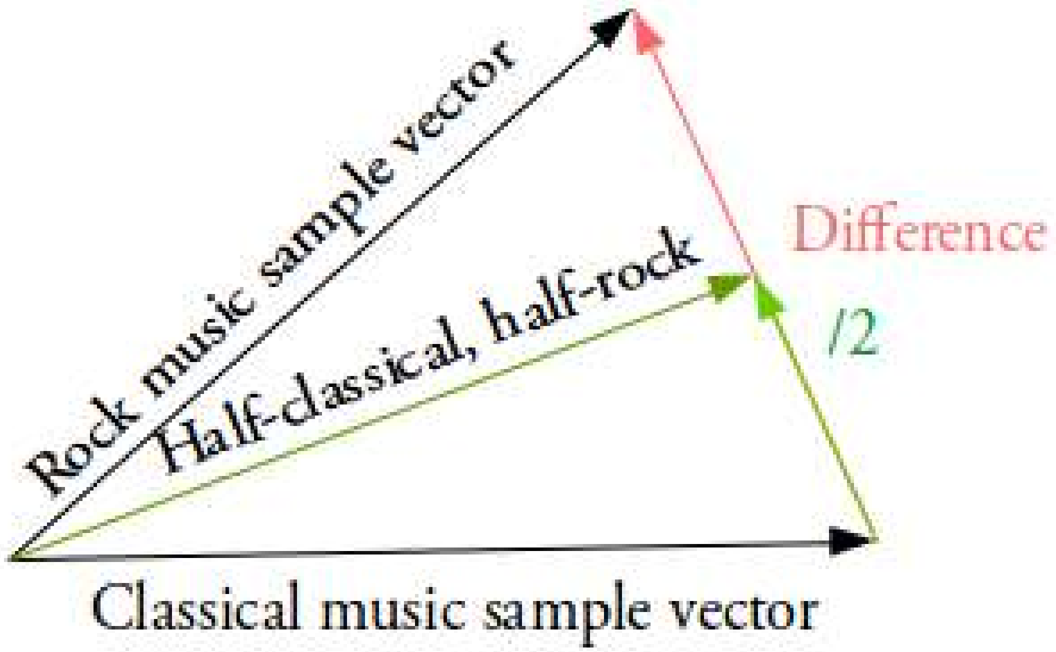
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3
1
2
2
9
8
7
6
5
4
3
2
1
0

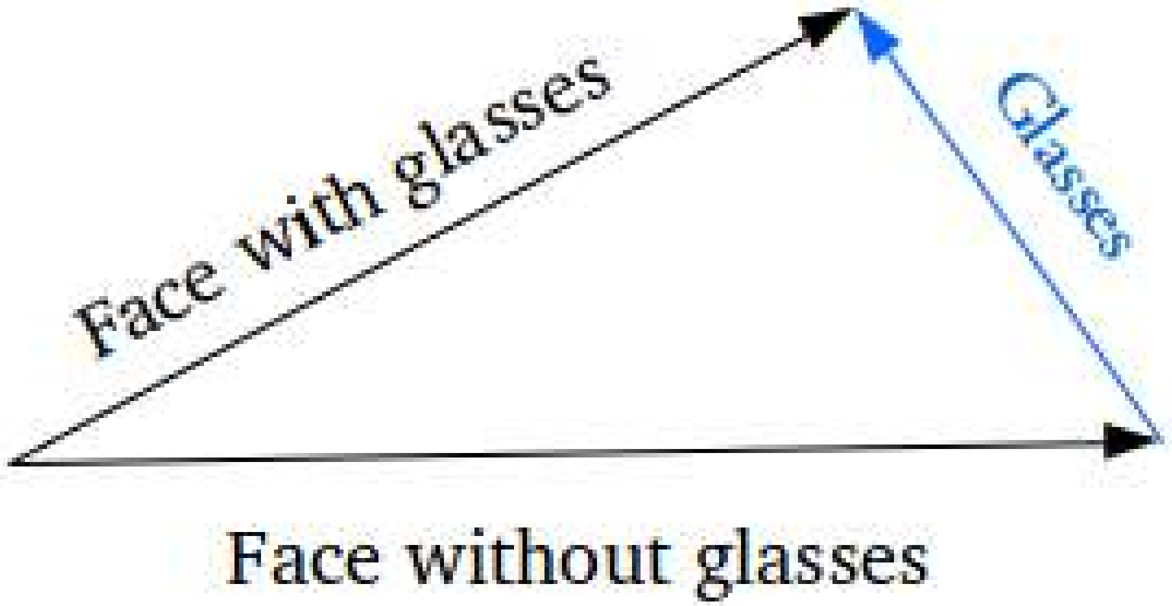
Секреты реализации

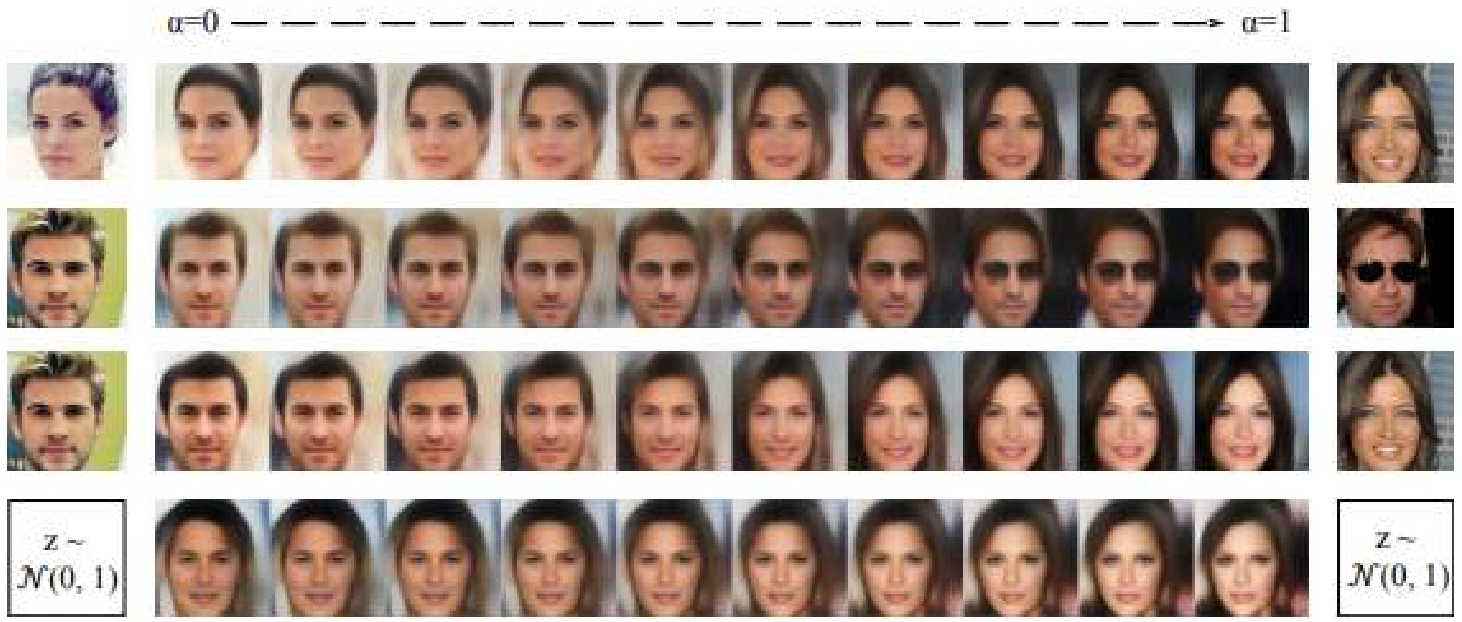
$$z = h(x)\zeta + g(x) \qquad \zeta \sim \mathcal{N}(0, I)$$

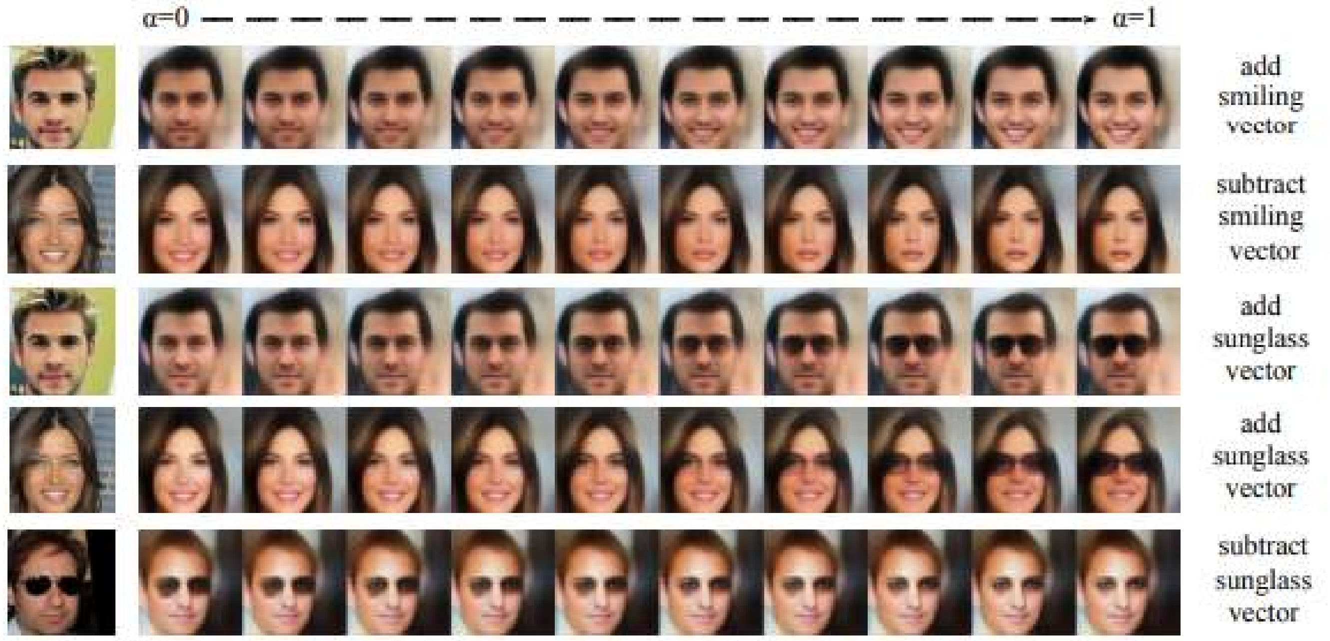


Векторная алгебра

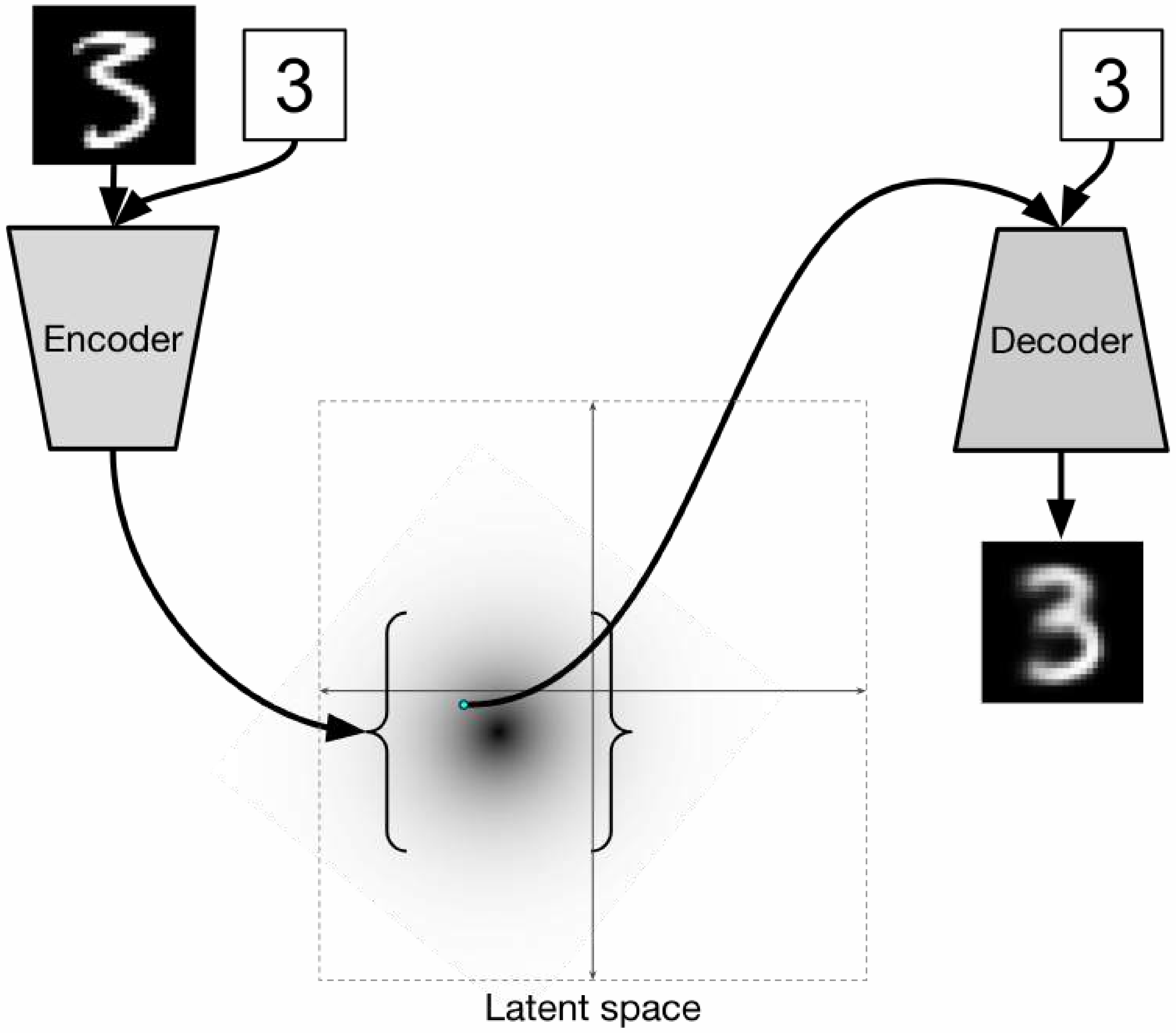


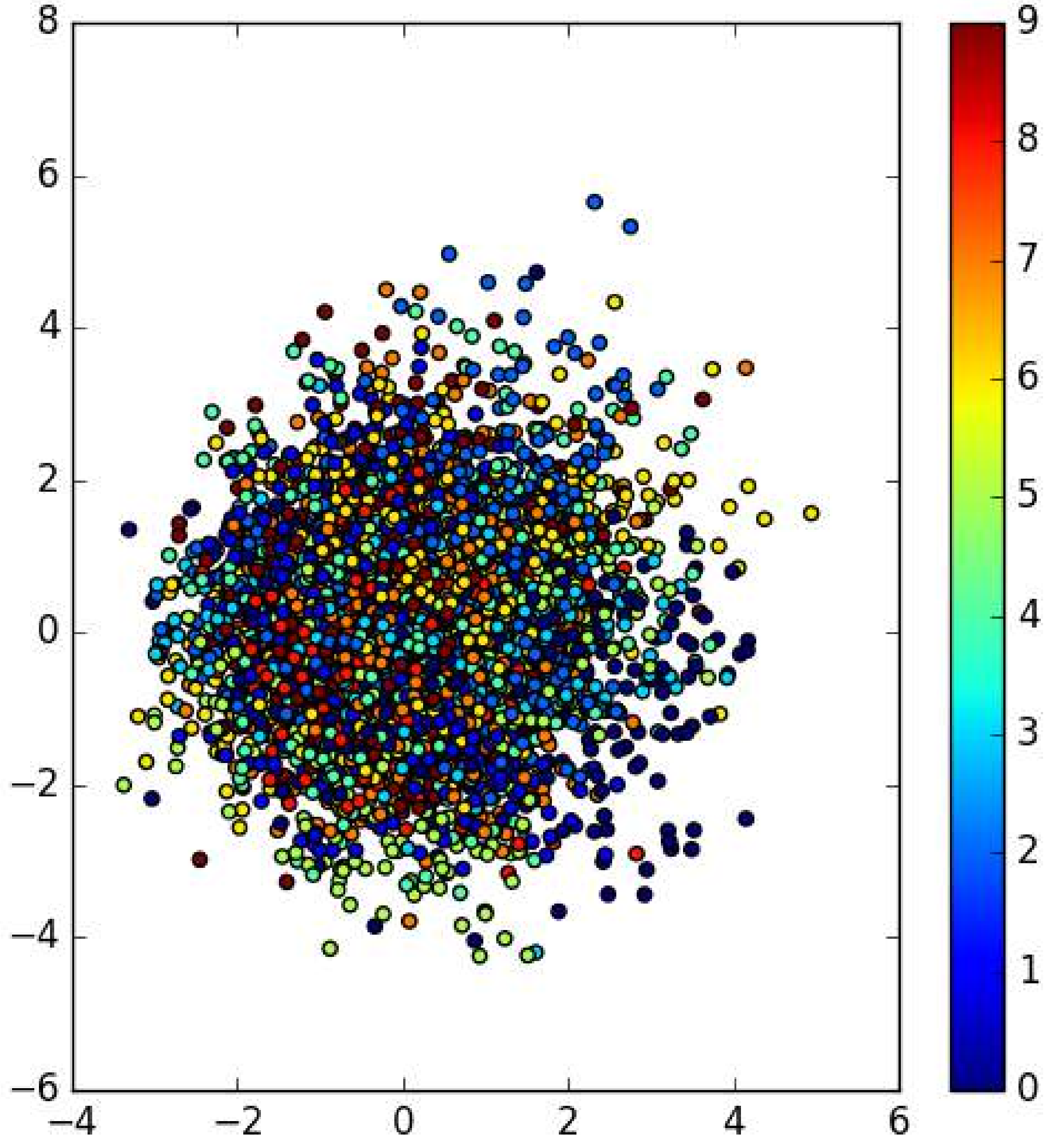




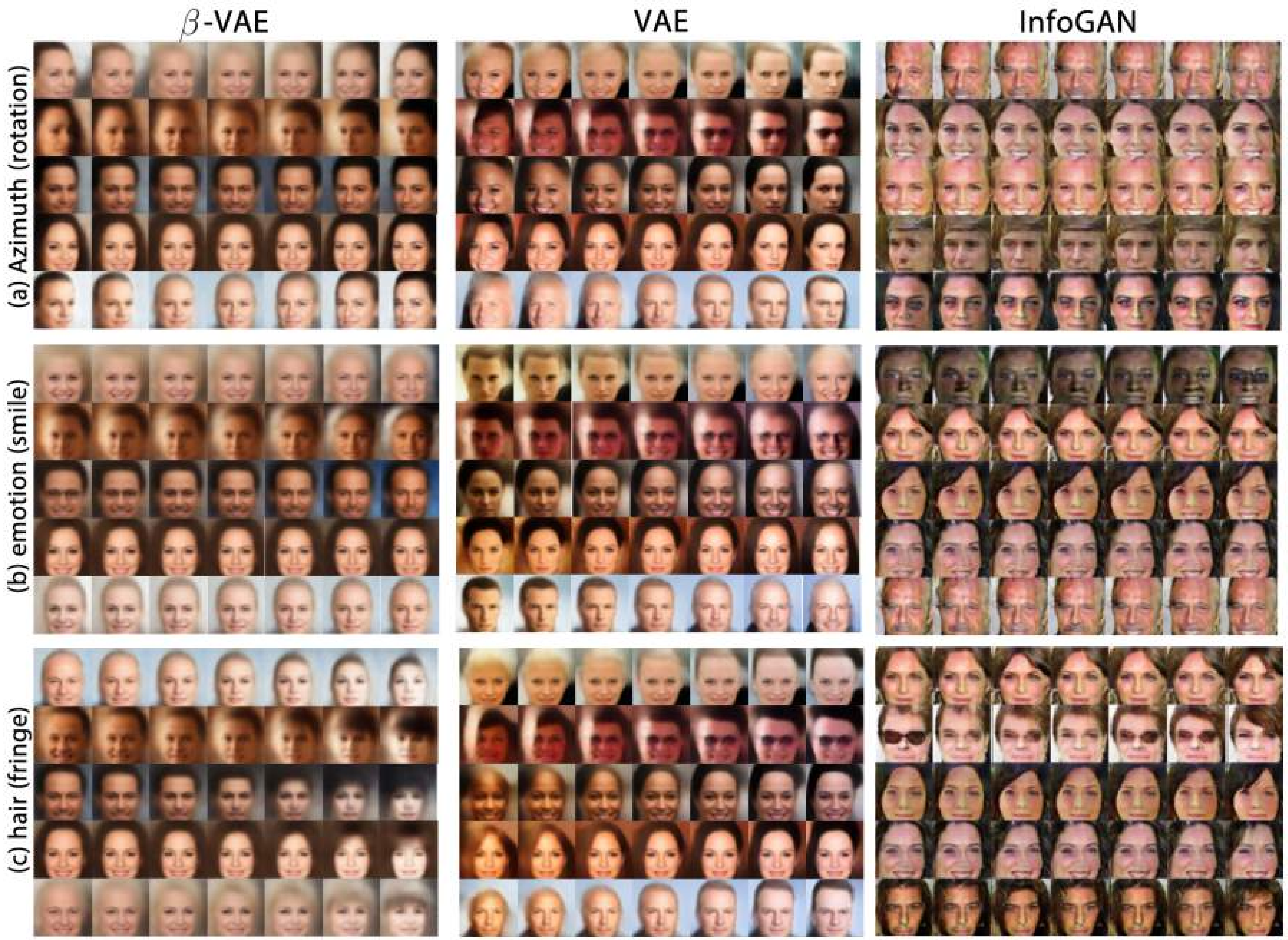


Conditional VAE





β-VAE и GAN



β-VAE
VAE
InfoGAN
(a) Azimuth (rotation)
(b) emotion (smile)
(c) hair (fringe)

Сравнение

VAE

- Более высокая скорость обучения
- Вариативность полученных изображений
- Отсутствие mode collapce

GAN

- Более высокая резкость изображений
- Отсутствие блюра

Normalizing Flows

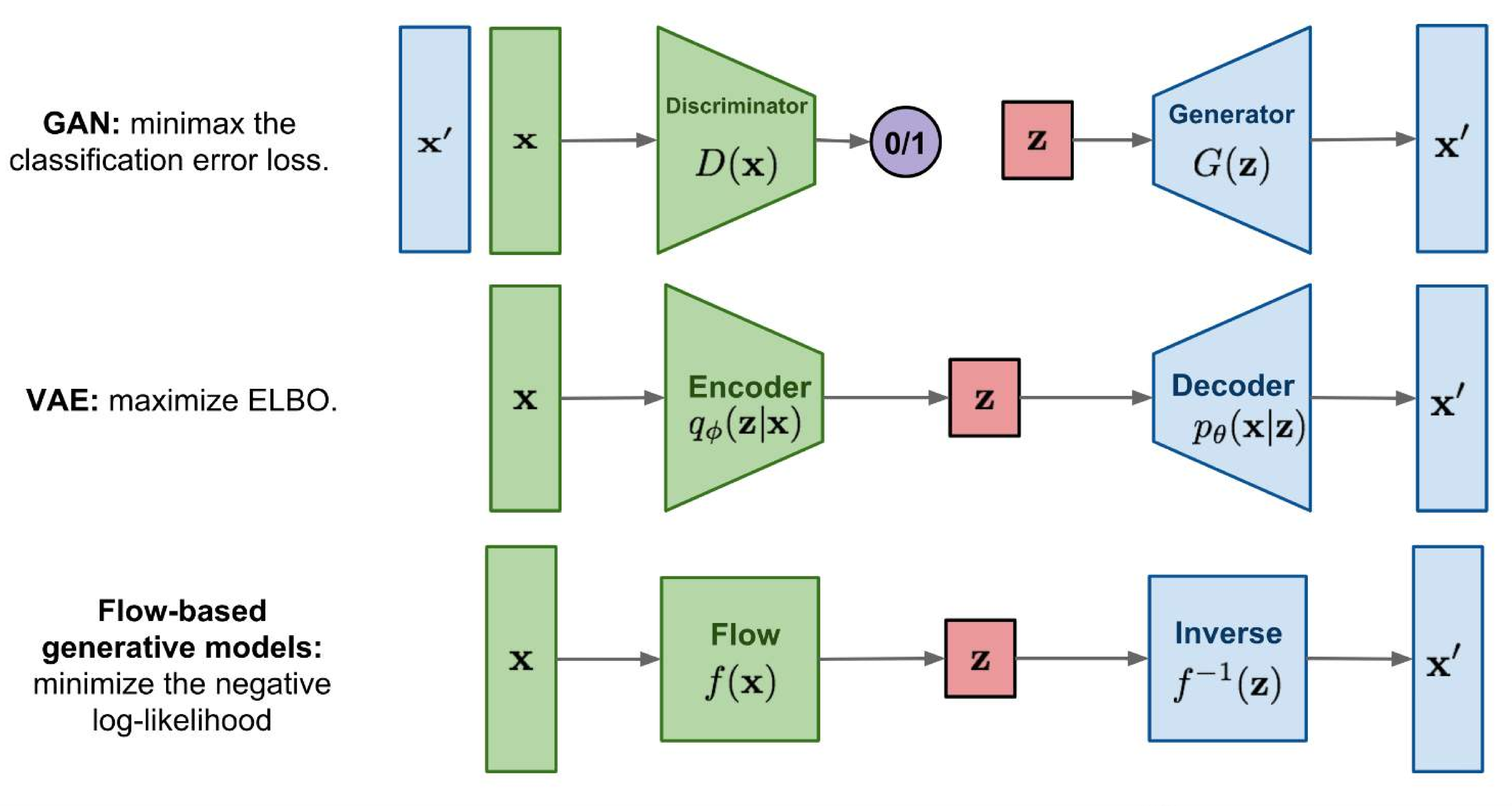


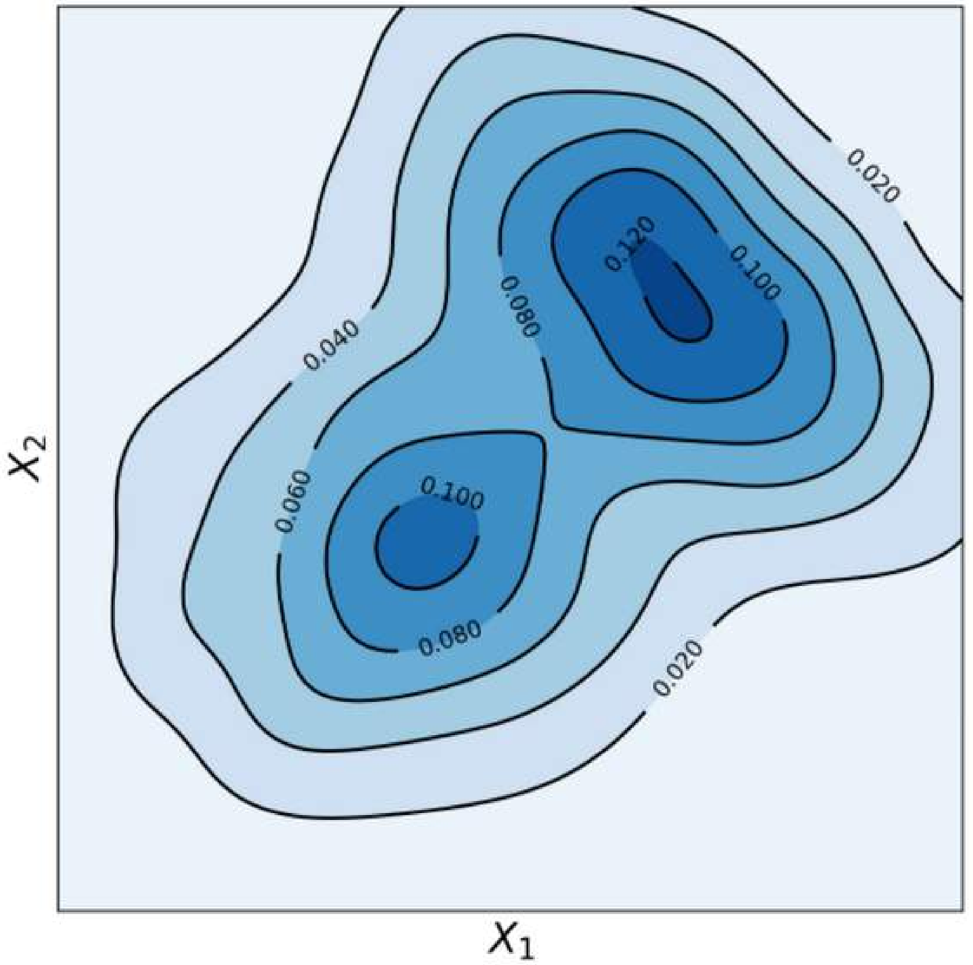
Fig. 1. Comparison of three categories of generative models.

VAE и GAN преобразуют Z в X необратимо.

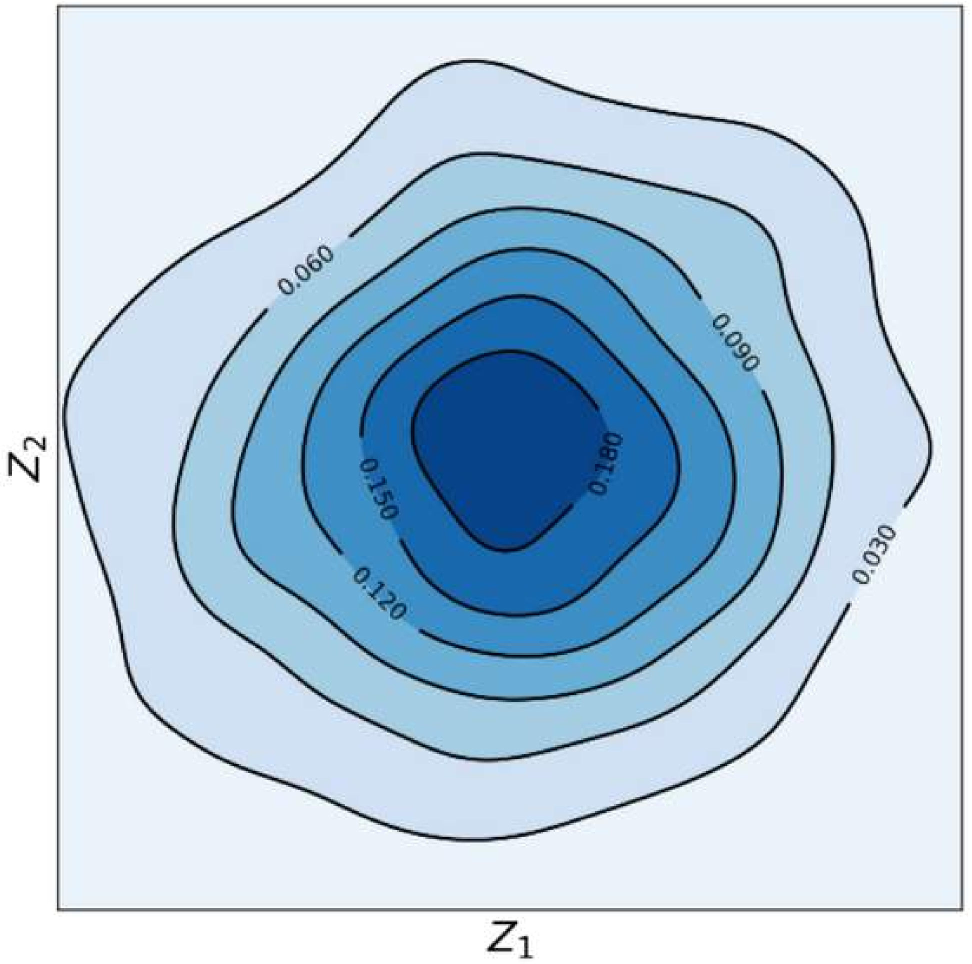
Для преобразования X в Z требуется обучать отдельную сеть **Encoder**.

Нормализационные потоки выучивают **обратимое** преобразование Z в X.

Нормализационные потоки позволяют посчитать $p_x(x)$.



$$z = f(x)$$



$x_i \sim p_x(x)$

$p_x(x)$ - ?

$z_i \sim p_z(z)$

$p_z(z)$ - известно

Теорема о замене переменной

Пусть даны $\boldsymbol{p_z}(z)$ и z $= \boldsymbol{f}(x)$, тогда $p_x(x)$ находим так:

Отношение объема ∂z к новому объему ∂x

$$p_x(x_i) = \boldsymbol{p_z}(\boldsymbol{f}(x_i)) \left| \det \frac{\partial \boldsymbol{f}(x_i)}{\partial x_i} \right|,$$

где матрица первых производных определяется так:

$$\frac{\partial \boldsymbol{f}(x_i)}{\partial x_i} = \begin{pmatrix} \frac{\partial \boldsymbol{f}(x_i)_1}{\partial x_{i1}} & \cdots & \frac{\partial \boldsymbol{f}(x_i)_1}{\partial x_{in}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{f}(x_i)_m}{\partial x_{i1}} & \cdots & \frac{\partial \boldsymbol{f}(x_i)_m}{\partial x_{in}} \end{pmatrix}.$$

Обратная замена

Пусть даны $\boldsymbol{p_z}(z)$ и z $= \boldsymbol{f}(x)$, тогда $p_x(x)$ находим так:

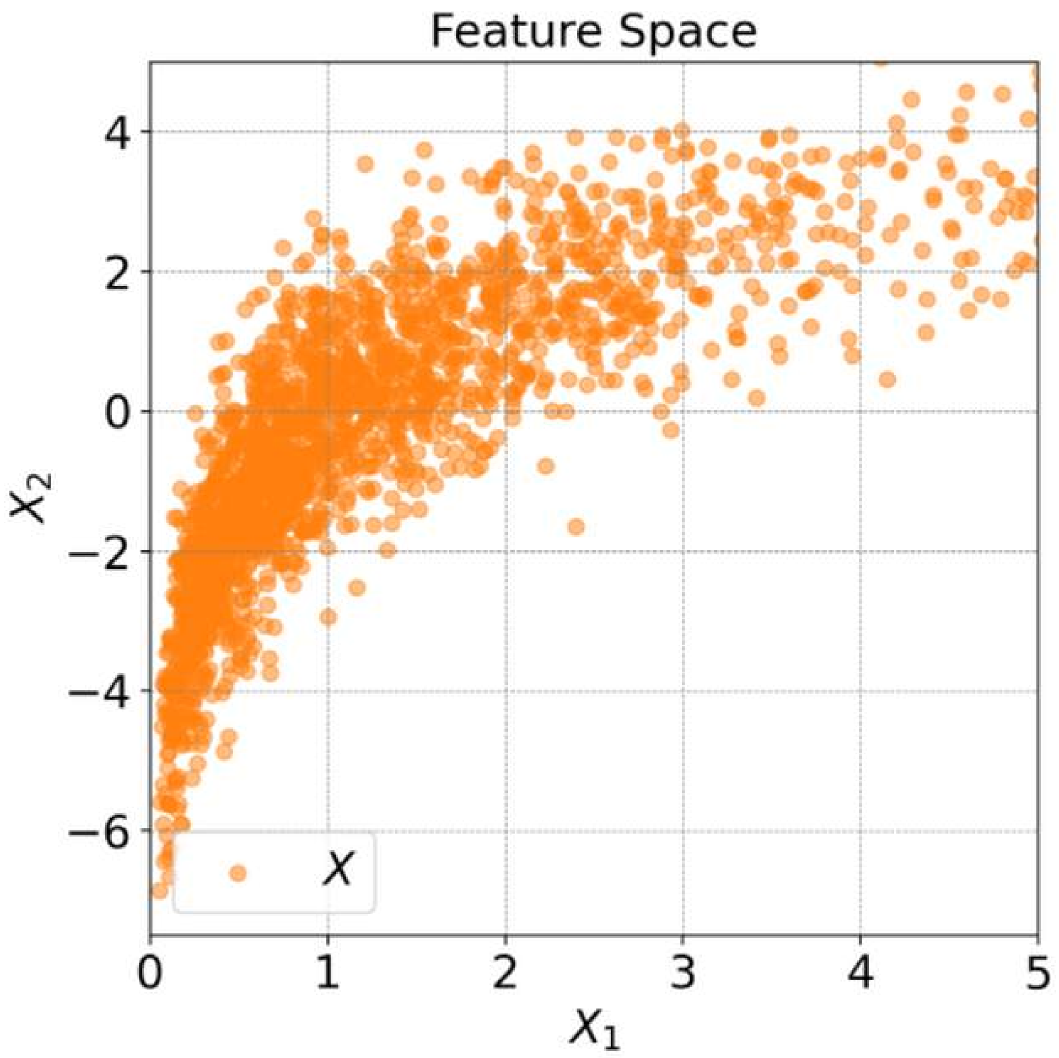
Отношение объема∂z к новому объему∂x

$$p_x(x_i) = \boldsymbol{p_z}\left(\boldsymbol{f}(x_i)\right)\left|\det\frac{\partial \boldsymbol{f}(x_i)}{\partial x_i}\right|,$$

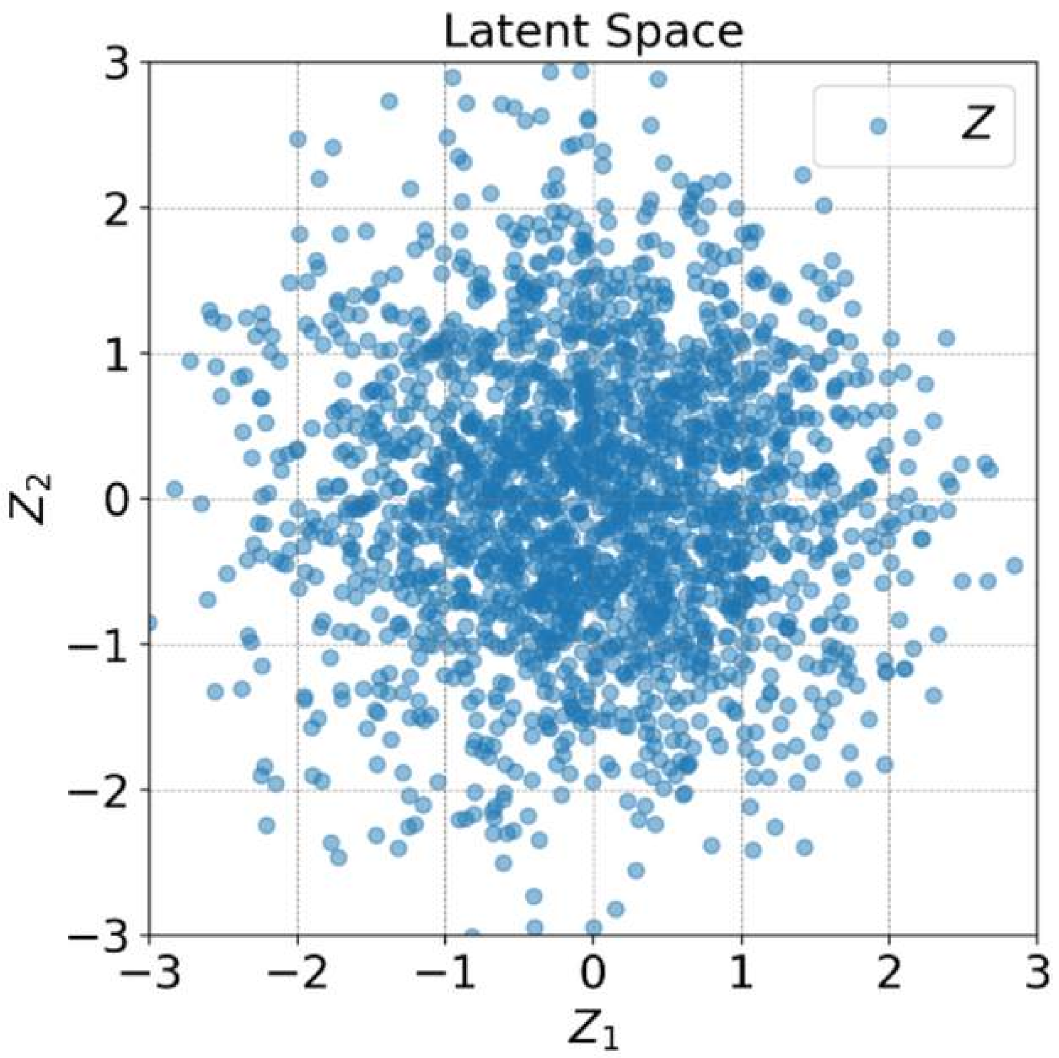
Обратная замена переменных:

$$\boldsymbol{p_z}(z_i) = p_x\left(\boldsymbol{f}^{-1}(z_i)\right)\left|\det\frac{\partial \boldsymbol{f}^{-1}(z_i)}{\partial z_i}\right|$$

Постановка задачи



$$z = f(x) - ?$$



$$x_i \sim p_x(x)$$

$$p_x(x) - ?$$

$$z_i \sim p_z(z)$$

$p_z(z)$ - известно

- **Дано:** матрица реальных объектов X
- **Задача:** найти такую $z_i = \boldsymbol{f}(x_i)$, чтобы $z_i \sim \boldsymbol{p_z}(z)$. При этом, $\boldsymbol{p_z}(z)$ известно и задано.

  - Как будем находить $z_i = \boldsymbol{f}(x_i)$?
  - **Ответ:** методом градиентного спуска!

  - Какую функцию потерь будем оптимизировать?
  - **Ответ:** логарифм правдоподобия:

$$L = -\frac{1}{n}\sum_{i=1}^{n} \log p_x(x_i)$$

Функция потерь

Функция потерь:

$$L = -\frac{1}{n}\sum_{i=1}^{n} \log p_x(x_i)$$

Замена переменных:

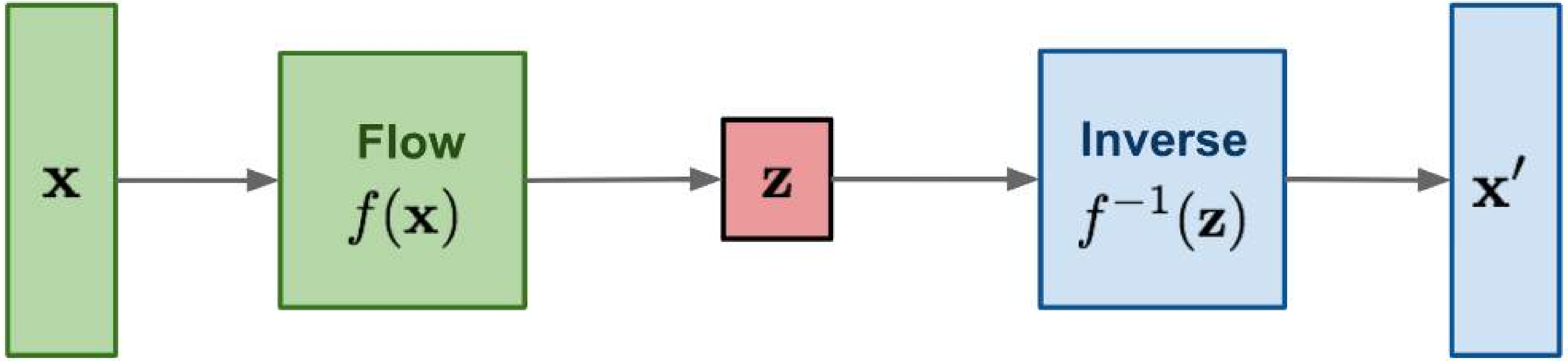
$$p_x(x_i) = \boldsymbol{p_z}(\boldsymbol{f}(x_i)) \left|\det \frac{\partial \boldsymbol{f}(x_i)}{\partial x_i}\right|$$

Подставим в функцию потерь:

$$L = -\frac{1}{n}\sum_{i=1}^{n} \left(\log \boldsymbol{p_z}(\boldsymbol{f}(x_i)) + \log \left|\det \frac{\partial \boldsymbol{f}(x_i)}{\partial x_i}\right|\right)$$

Алгоритм

Flow-based generative models:
minimize the negative log-likelihood



Алгоритм обучения

for number of training iterations **do:**

- Sample m of real objects $\{x_1, x_2, \dots, x_m\}$.
- Calculate loss function:

$$L = -\frac{1}{m}\sum_{i=1}^{m}\left(\log \boldsymbol{p_z}(\boldsymbol{f}(x_i)) + \log\left|\det\frac{\partial \boldsymbol{f}(x_i)}{\partial x_i}\right|\right)$$

- Update parameters of the function $z_i = \boldsymbol{f}(x_i)$:

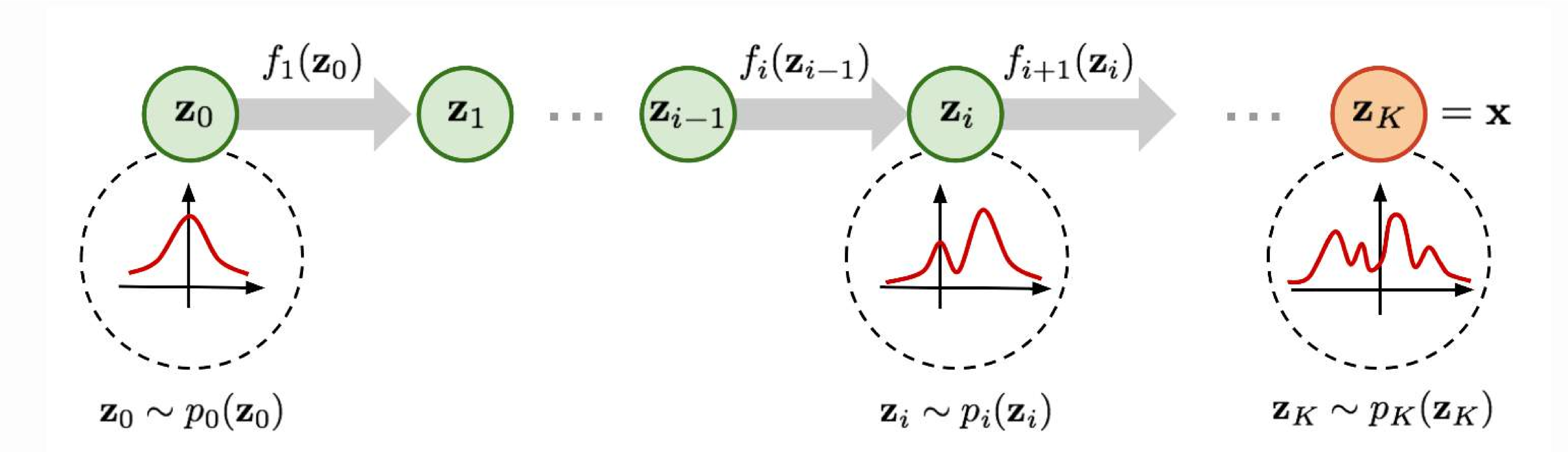
$$\theta_f = \theta_f - \nabla_{\theta_f} L$$

Алгоритм генерации

- Sample m of noise objects $\{z_1, z_2, \dots, z_m\}$.
- Generate new objects using the learned function:

$$x_i = f^{-1}(z_i)$$

Добавление слоёв



Пример с двумя слоями

Пусть $z_i = \boldsymbol{f_2}(y_i), y_i = \boldsymbol{f_1}(x_i)$.

Тогда:

$$p_x(x_i) = \boldsymbol{p_y}\left(\boldsymbol{f_1}(x_i)\right)\left|\det\frac{\partial \boldsymbol{f_1}(x_i)}{\partial x_i}\right|$$

$$p_y(y_i) = \boldsymbol{p_z}\left(\boldsymbol{f_2}(y_i)\right)\left|\det\frac{\partial \boldsymbol{f_2}(y_i)}{\partial y_i}\right|$$

В итоге получим:

$$p_x(x_i) = \boldsymbol{p_z}\left(\boldsymbol{f_2}\left(\boldsymbol{f_1}(x_i)\right)\right)\left|\det\frac{\partial \boldsymbol{f_2}(y_i)}{\partial y_i}\right|\left|\det\frac{\partial \boldsymbol{f_1}(x_i)}{\partial x_i}\right|$$

Критерии функции f

Как выбрать такую функцию $z_i = f(x_i)$, чтобы она:

- была дифференцируемой,
- была обратимой?

Примеры:

Real-NVP

$$z = f(x) = \begin{cases} z_{1:d} = x_{1:d} \\ z_{d+1:D} = x_{d+1:D} \odot \exp\left(s(x_{1:d})\right) + t(x_{1:d}) \end{cases}$$

где:

- $z_{1:d}$ - первые d компонент вектора z;
- $s(x_{1:d})$ и $t(x_{1:d})$ – **нейронные сети** с d входами и $D - d$ выходами;
- $\odot$ - поэлементное умножение.

Real-NVP

- Матрица первых производных:

$$\frac{\partial \boldsymbol{f}(x)}{\partial x} = \begin{pmatrix} \mathbb{I}_d & 0 \\ \frac{\partial z_{1:d}}{\partial x_{1:d}} & diag\left(\exp(\boldsymbol{s}(x_{1:d}))\right) \end{pmatrix}$$

- Значение Якобиана:

$$\left|\det\frac{\partial \boldsymbol{f}(x)}{\partial x}\right| = \exp(\sum_{j=d+1}^{D} \boldsymbol{s}(x_{1:d})_j)$$

$$x = \boldsymbol{f}^{-1}(z) = \begin{cases} x_{1:d} = z_{1:d} \\ x_{d+1:D} = (z_{d+1:D} - \boldsymbol{t}(x_{1:d})) \odot \exp\left(-\boldsymbol{s}(x_{1:d})\right) \end{cases}$$

Рис.: https://github.com/laurent-dinh/laurent-dinh.github.io/blob/master/img/real_nvp_fig/celeba_samples.png

Masked Autoregressive Flow

$$z = \boldsymbol{f}(x) = \begin{cases} z_1 = (x_1 - \mu_1)\exp(-s_1) \\ z_d = (x_d - \boldsymbol{\mu_d}(x_{1:d-1})) \odot \exp(-\boldsymbol{s_d}(x_{1:d-1})) \end{cases}$$

где:

- $z_{1:d}$ - первые d компонент вектора z;
- $\boldsymbol{\mu_d}(x_{1:d-1})$ и $\boldsymbol{s_d}(x_{1:d-1})$ – **нейронные сети** с $d-1$ входами и 1 выходом;
- $\odot$ - поэлементное умножение.

- Матрица первых производных **нижнетреугольная**:

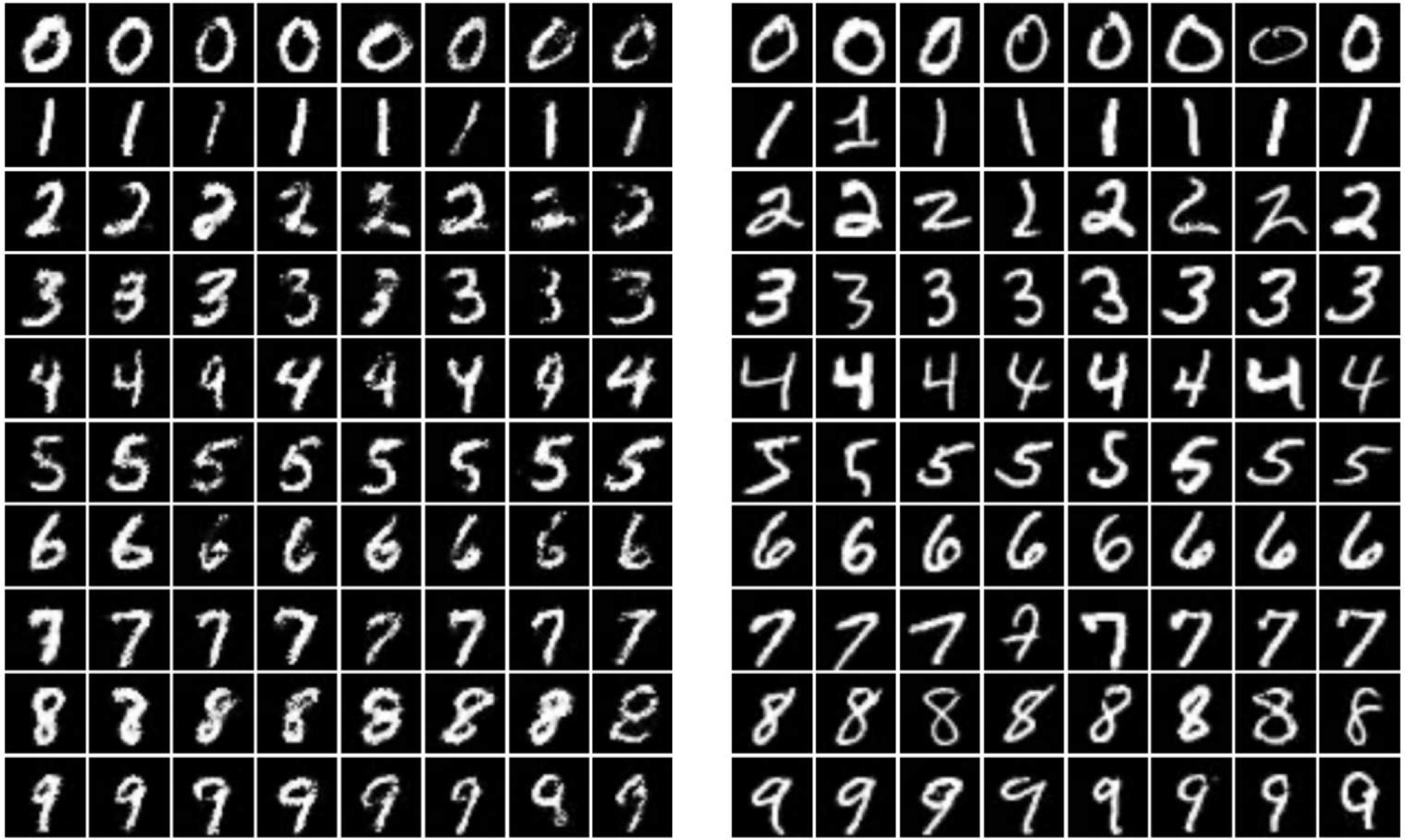
$$\frac{\partial \boldsymbol{f}(x)}{\partial x} = \begin{pmatrix} \exp(-\boldsymbol{s_1}) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \dfrac{\partial z_D}{\partial x_1} & \cdots & \exp(-\boldsymbol{s_D}(x_{1:D-1})) \end{pmatrix}$$

Masked Autoregressive Flow

- Значение Якобиана:

$$\left|\det \frac{\partial \boldsymbol{f}(x)}{\partial x}\right| = \exp(-\sum_{j=1}^{D} \boldsymbol{s_d}(x_{1:d-1}))$$

$$x = \boldsymbol{f}^{-1}(z) = \begin{cases} x_1 = z_1 \exp(s_1) + \mu_1 \\ x_d = z_d \exp(\boldsymbol{s_d}(x_{1:d-1})) + \boldsymbol{\mu_d}(x_{1:d-1}) \end{cases}$$



(a) Generated images (b) Real images

Inverse Autoregressive Flow (IAF)

$$z = \boldsymbol{f}(x) = \begin{cases} z_1 = (x_1 - \mu_1)\exp(-s_1) \\ z_d = (x_d - \boldsymbol{\mu_d}(z_{1:d-1}))\exp(-\boldsymbol{s_d}(z_{1:d-1})) \end{cases}$$

$$x = \boldsymbol{f}^{-1}(z) = \begin{cases} x_1 = z_1 \exp(s_1) + \mu_1 \\ x_d = z_d \exp(\boldsymbol{s_d}(z_{1:d-1})) + \boldsymbol{\mu_d}(z_{1:d-1}) \end{cases}$$

Спасибо за внимание