

# DL: Сверточные сети

## Приложения

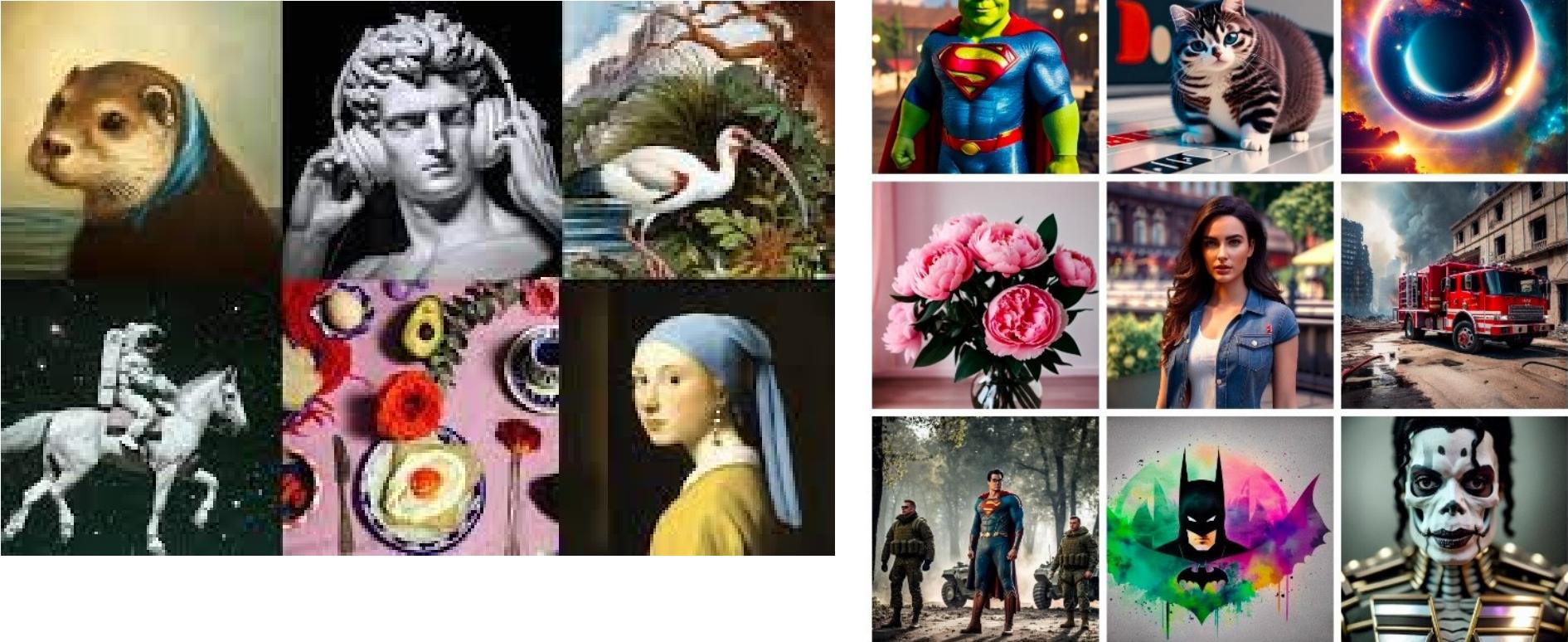
# План

- Автоэнкодер
- GAN
- Диффузионные модели
- Нормализационные потоки

# Генеративные сети: приложения

- Генерация фотографий
- Stable Diffusion (Kandinsky)
- Text-to-Video
- Deepfake
- Deepfake detection

# Генерация фотографий



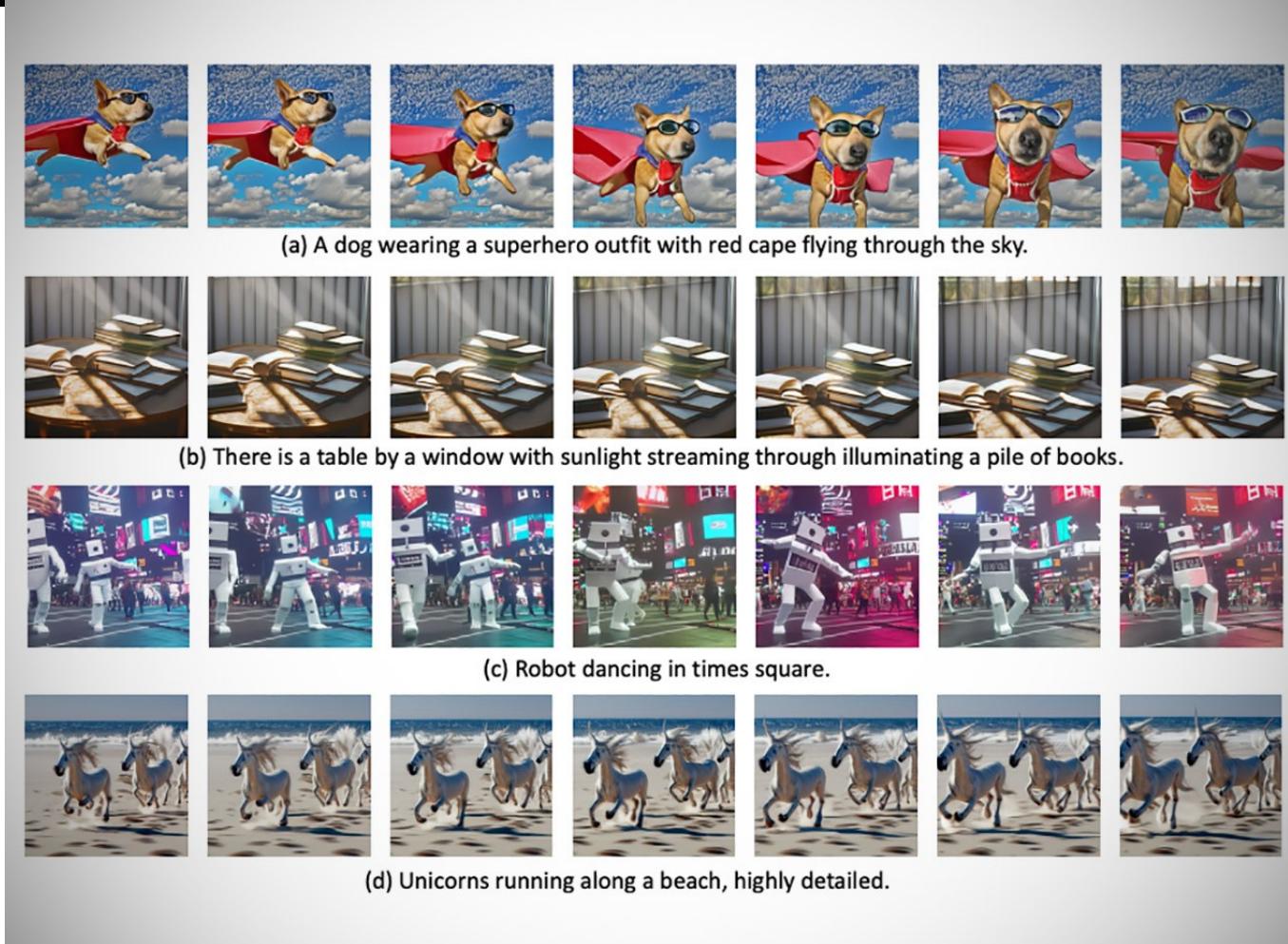
- Stable Diffusion (Kandinsky...)



- Stable Diffusion (Kandinsky...)



- Text-to-Video Meta



# • Deepfake



John Wick 4: Dangerous Cleanup  
7,4 млн просмотров

This is you right now 😂  
1,2 млн просмотров

Katana wins  
18 млн просмотров

Thank you, my followers ❤️  
1,9 млн просмотров

10 minutes of eternity...  
1,9 млн просмотров

Keanu Reeves lives with his girlfriend  
19 млн просмотров



Did you recognize everyone?  
2,1 млн просмотров



When I came back from filming  
1,2 млн просмотров



You spin USB right 'round,  
baby, right 'round... #keanu...  
1,5 млн просмотров



Who is the best dancer? 😊  
#keanu #reeves #dance #fyp  
7,9 млн просмотров



Mirror trick 😊 #keanu  
#reeves #mirror #tenet  
1,1 млн просмотров



Dressing up like a cool guy.  
#reeves #keanu #dressup  
1,1 млн просмотров

- Deepfake



## Unreal Keanu Reeves

@unreal\_keanu · 1,71 млн подписчиков · 69 видео

Welcome to «Unreal Keanu Reeves», the deepfake channel of actor Keanu Reeves, the kind... >

[tiktok.com/@unreal\\_keanu](https://tiktok.com/@unreal_keanu) и ещё 5 ссылок

Подписаться

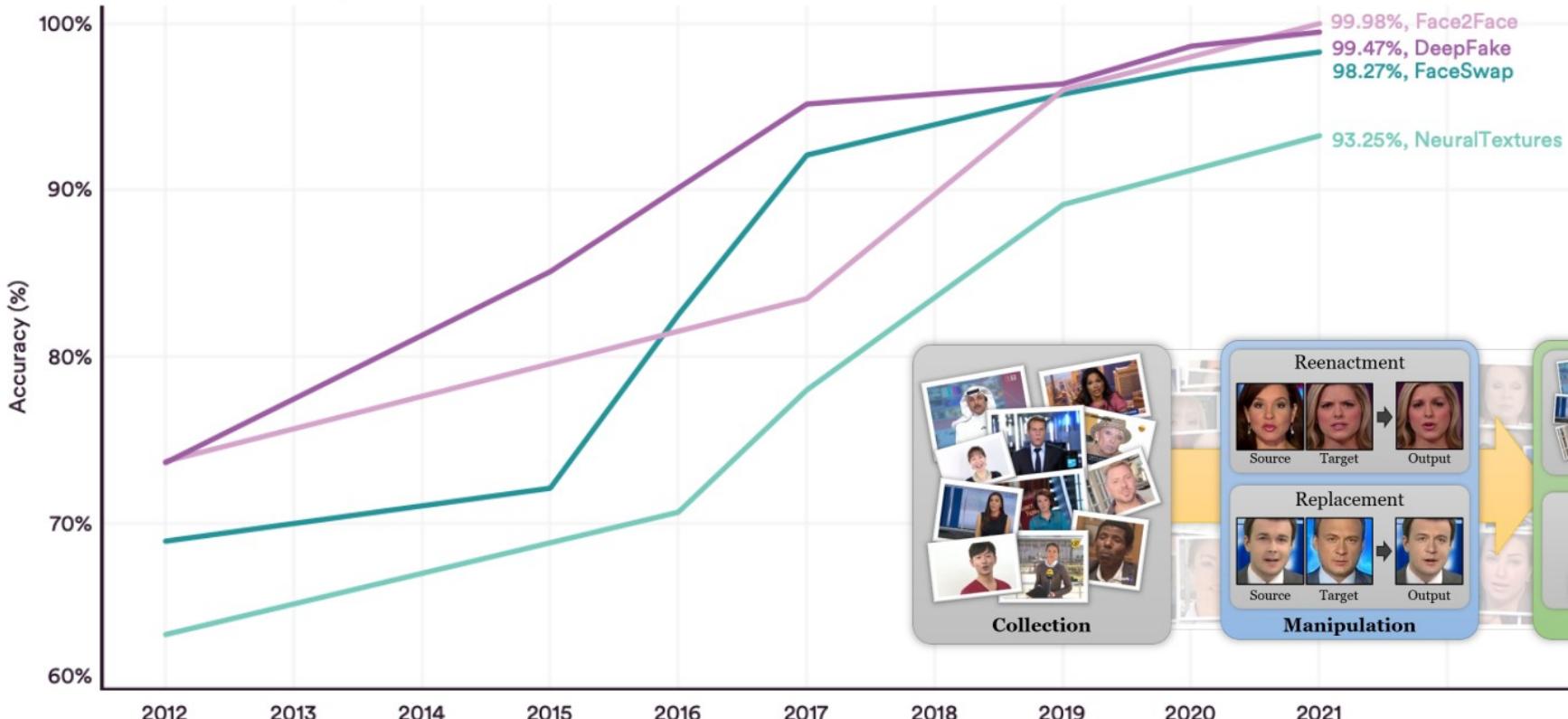


[https://www.youtube.com/@unreal\\_keanu/about](https://www.youtube.com/@unreal_keanu/about)

# • Deepfake detection

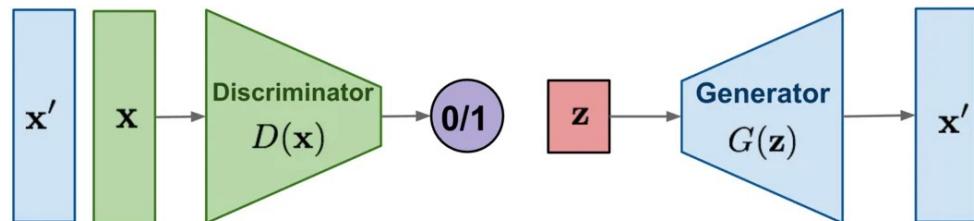
## FACEFORENSICS++: ACCURACY

Source: arXiv, 2021 | Chart: 2022 AI Index Report

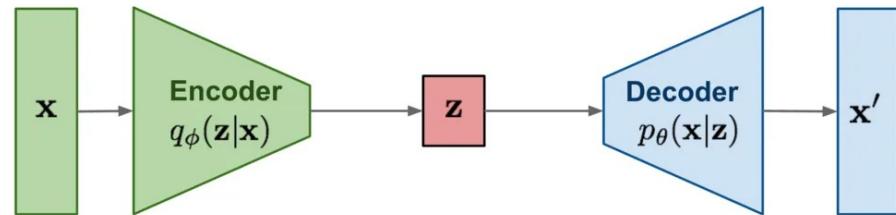


# Генеративные модели

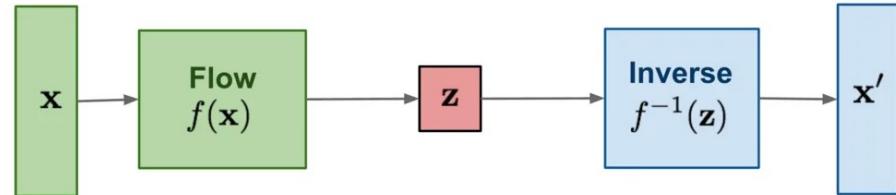
**GAN:** Adversarial training



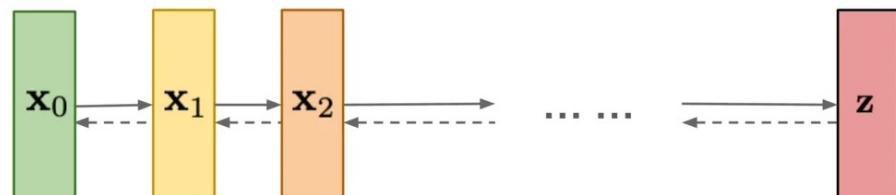
**VAE:** maximize variational lower bound



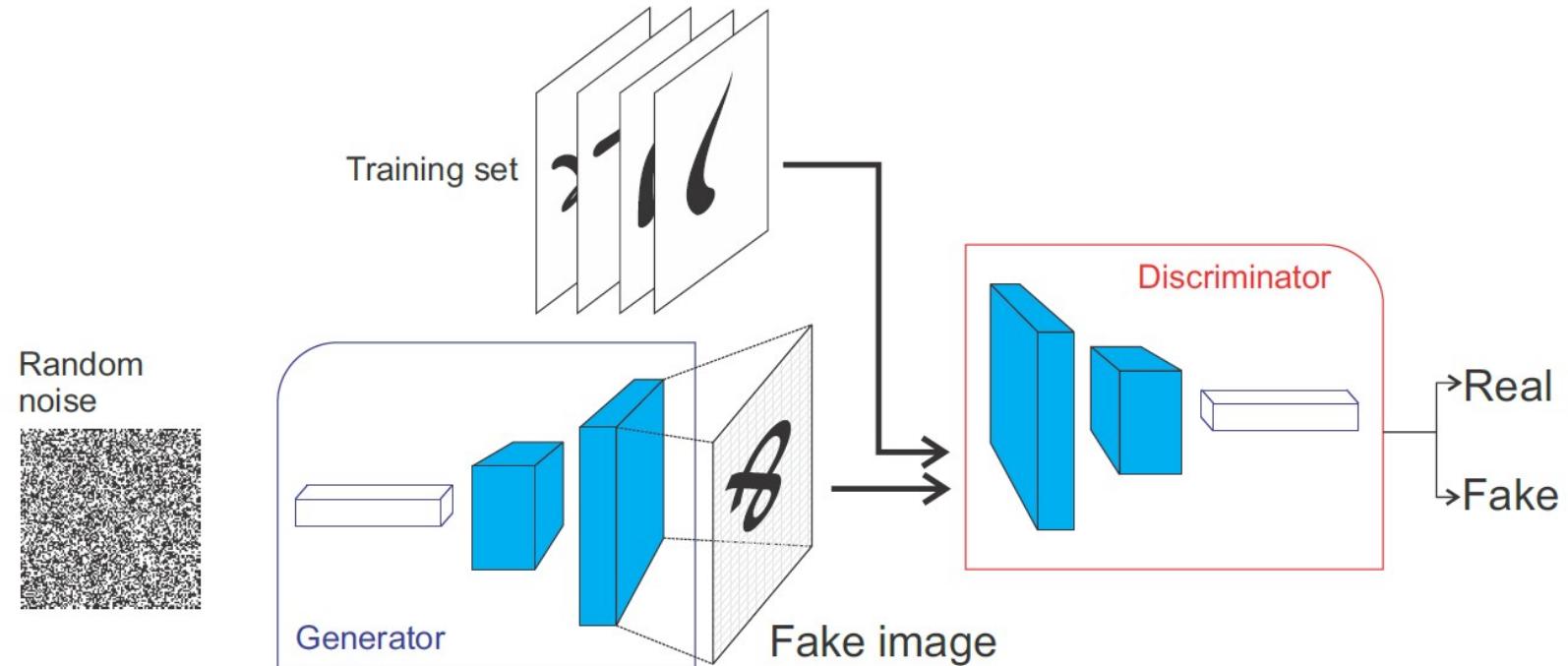
**Flow-based models:**  
Invertible transform of distributions



**Diffusion models:**  
Gradually add Gaussian noise and then reverse

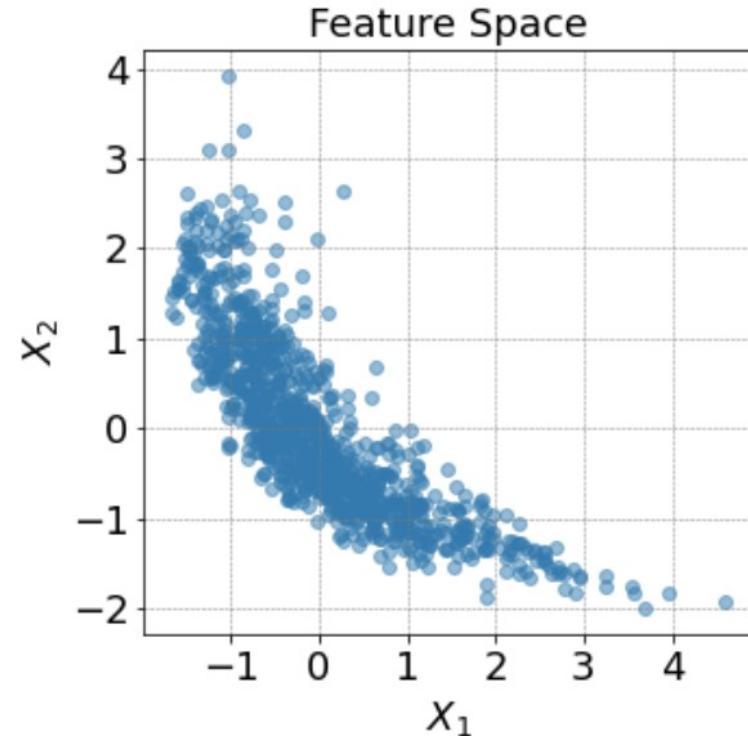


# Генеративно — состязательные сети



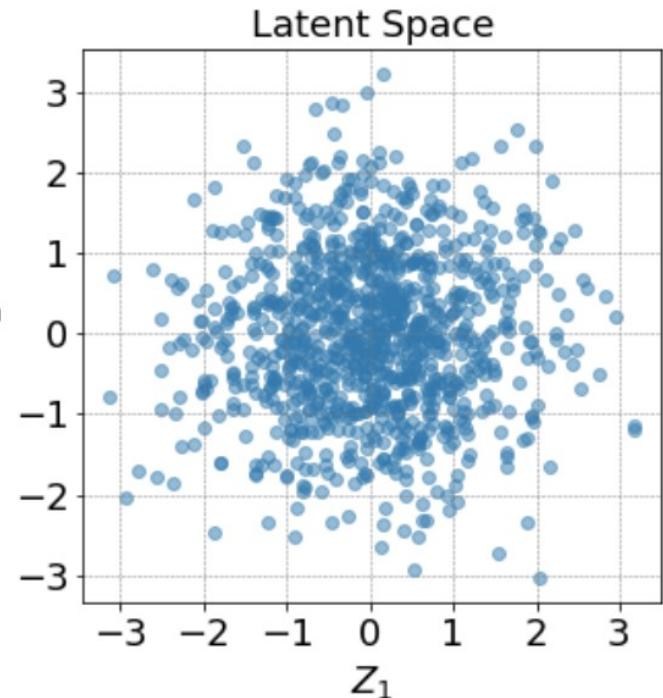
# Постановка задачи

- Дано: выборка объектов  $x_i \sim p_x(x)$ , где распределение  $p_x(x)$  неизвестно
- Задача: сгенерировать новые объекты  $\tilde{x}_j \sim p_x(x)$  с таким же распределением, т. е. похожими на  $x_i$

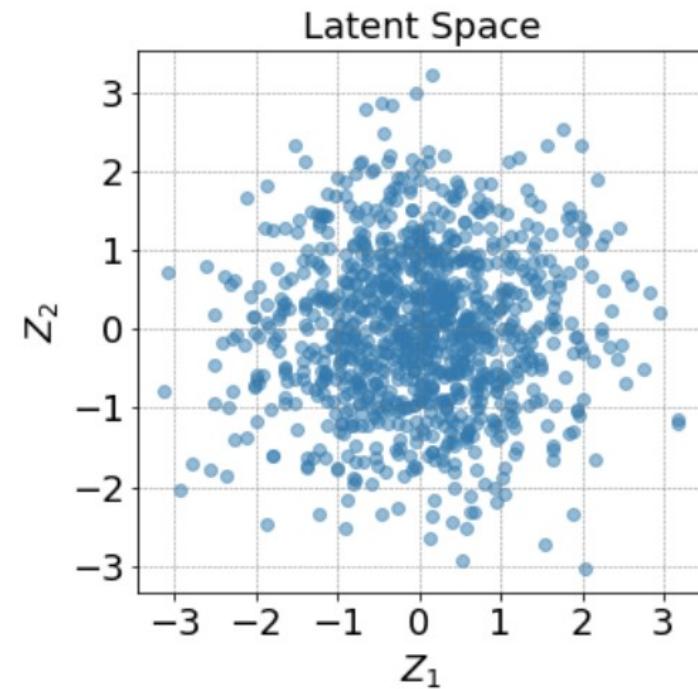


# Известные распределения

- Мы можем (умеем) генерировать объекты из известных распределений
- Генерируем выборку объектов  $z_i \sim p_z(z)$ , где  $p_z(z)$  например нормальное распределение
- Пространство Z называют скрытым

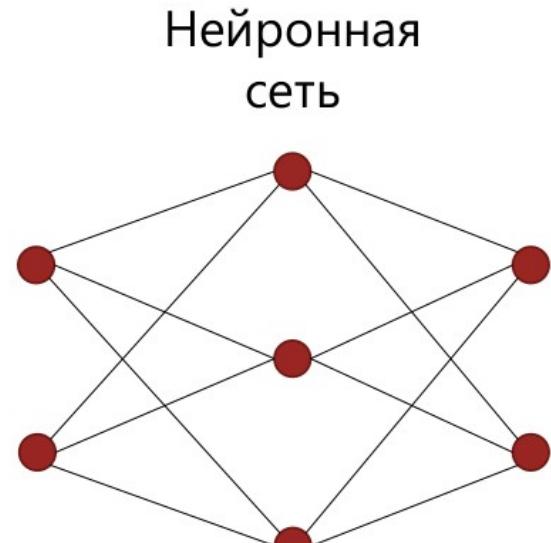


# Генератор



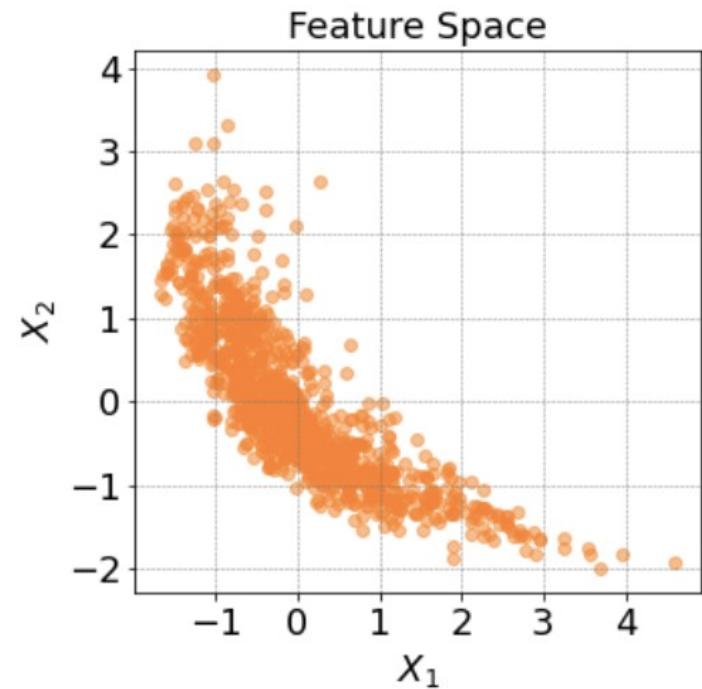
$Z$

Случайный шум



$G(Z)$

Генератор



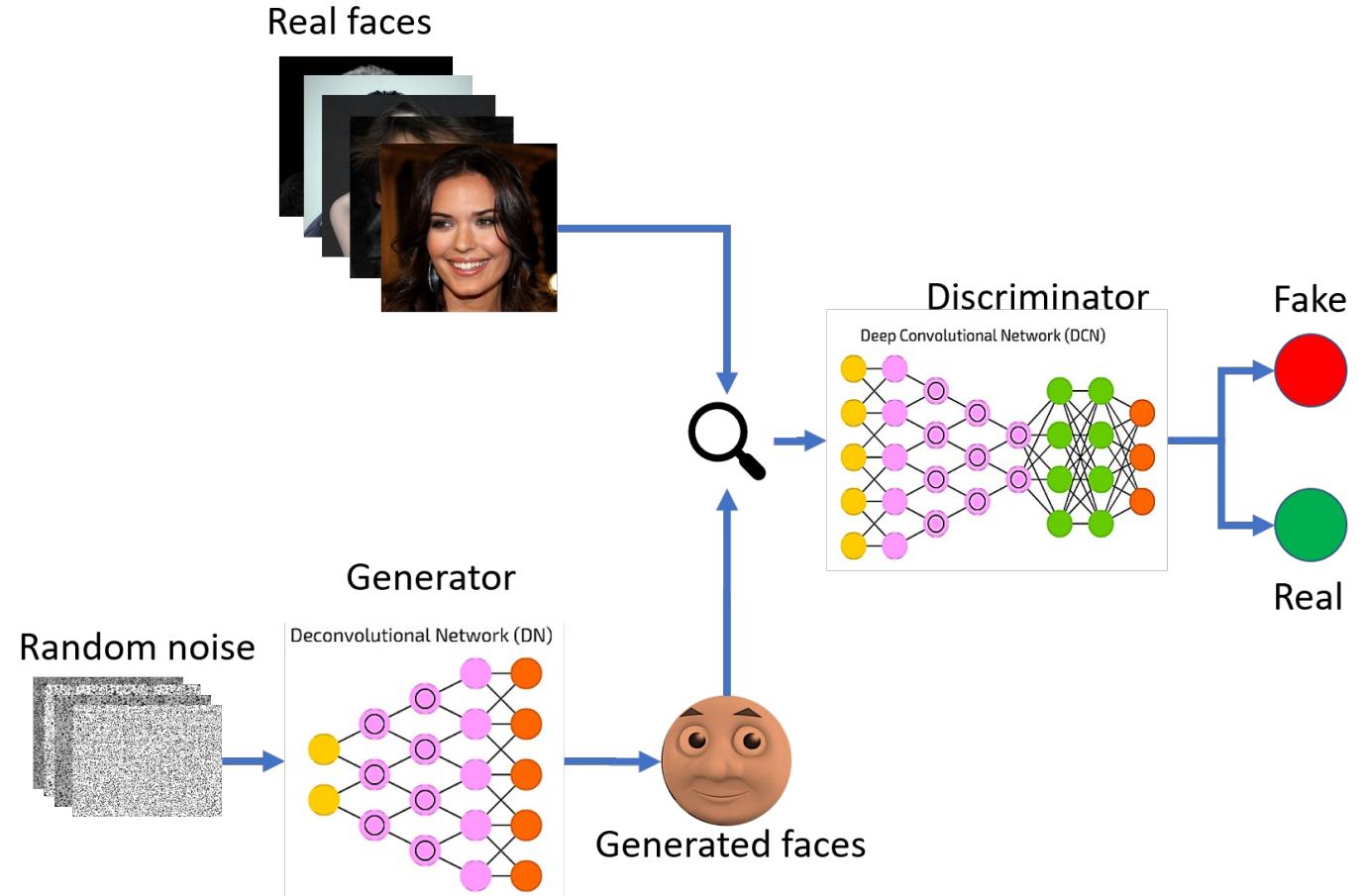
$\hat{X} = G(Z)$

Сгенерированные  
объекты

# Генератор

- $X$  — реальные объекты
- $\hat{X} = G(Z)$  - сгенерированные объекты
- Хотим, чтобы объекты  $\hat{X}$  были максимально похожи на  $X$
- Как измерить расстояние между объектами?
- Как обучить генератор?

# GAN



# Дискриминатор

- Рассмотрим задачу бинарной классификации. Пусть:
  - $X$  - класс 1: реальный объект
  - $\hat{X}$  - класс 0: сгенерированный объект
- Задача классификатора — разделить 2 класса. В ГАНах этот классификатор называют дискриминатором

# Дискриминатор: функция потерь

- Имеем 2 класса объектов (реальные и сгенерированные).
- $X$  - класс 1: реальный объект
- $\hat{X}$  - класс 0: сгенерированный объект
- Задача классификатора — разделить 2 класса. В ГАНах этот классификатор называют дискриминатором

# Дискриминатор: функция потерь

- Бинарная функция классификации:

$$L = -\frac{1}{N} \sum y_i \ln D(x_i) + (1 - y_i) \ln(1 - D(x_i))$$

- $D(x)$  — выход дискриминатора,  $y_i$  — метка класса

$$L = -\frac{1}{N} \sum y_i \ln D(x_i) - \frac{1}{N} \sum (1 - y_i) \ln(1 - D(x_i))$$

-

# Дискриминатор: функция потерь

- Исходно, мы вводили:
- $X$  — класс 1, реальные объекты,  $\hat{X} = G(Z)$  - класс 0, сгенерированные объекты
- В итоге, функция потерь:

$$L(D, L) = -\frac{1}{N} \sum y_i \ln D(x_i) - \frac{1}{N} \sum (1 - y_i) \ln (1 - D(G(z_i)))$$

•

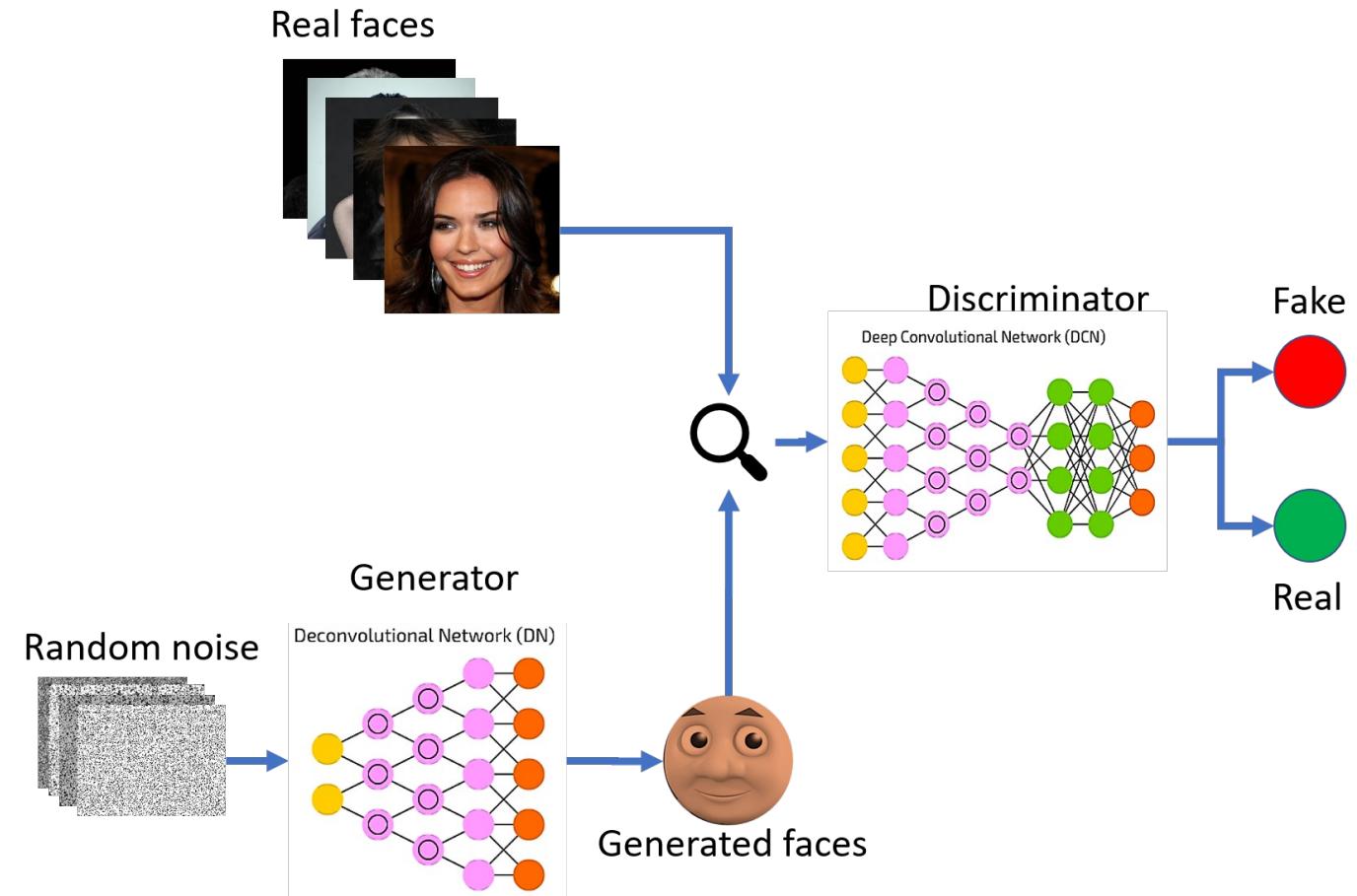
# Обучение дискриминатора и генератора

$$L(D, G) = -\frac{1}{N} \sum y_i \ln D(x_i) - \frac{1}{N} \sum (1 - y_i) \ln (1 - D(G(z_i)))$$

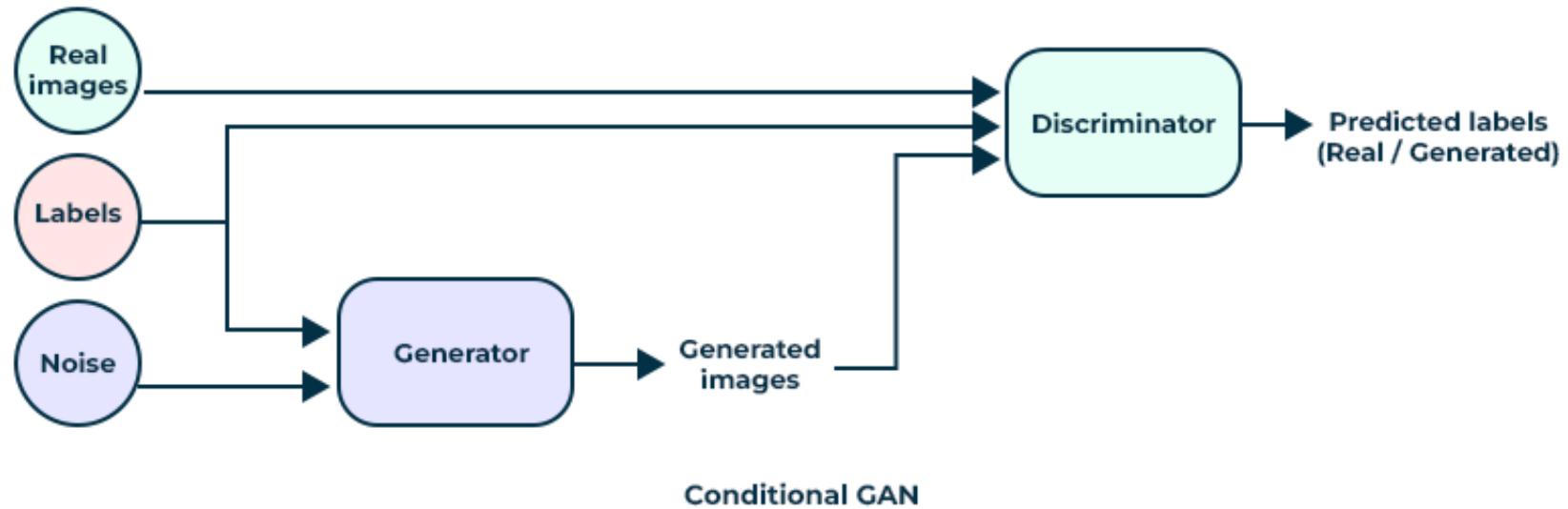
•

$$\max_G \min_D L(D, G)$$

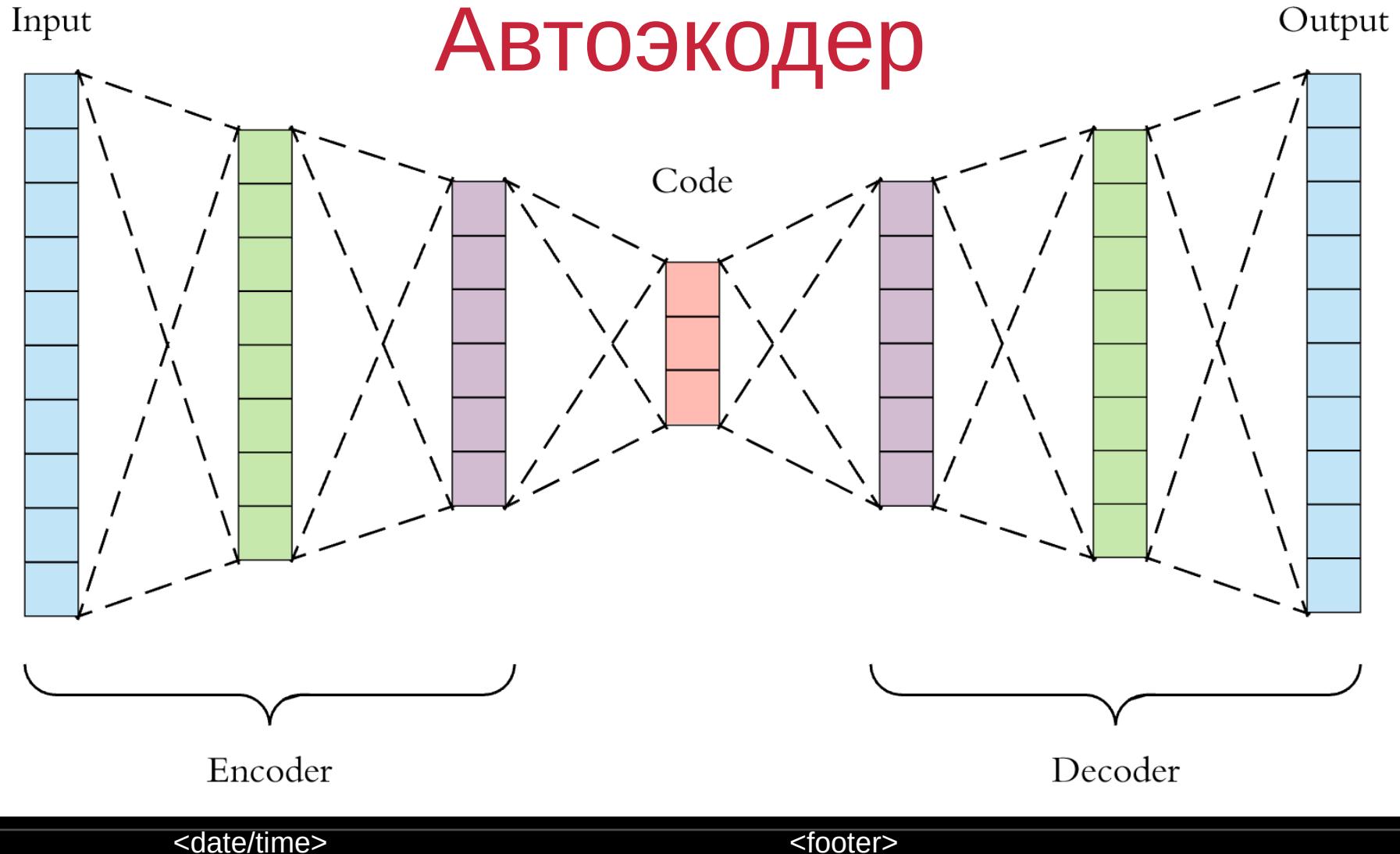
# Алгоритм



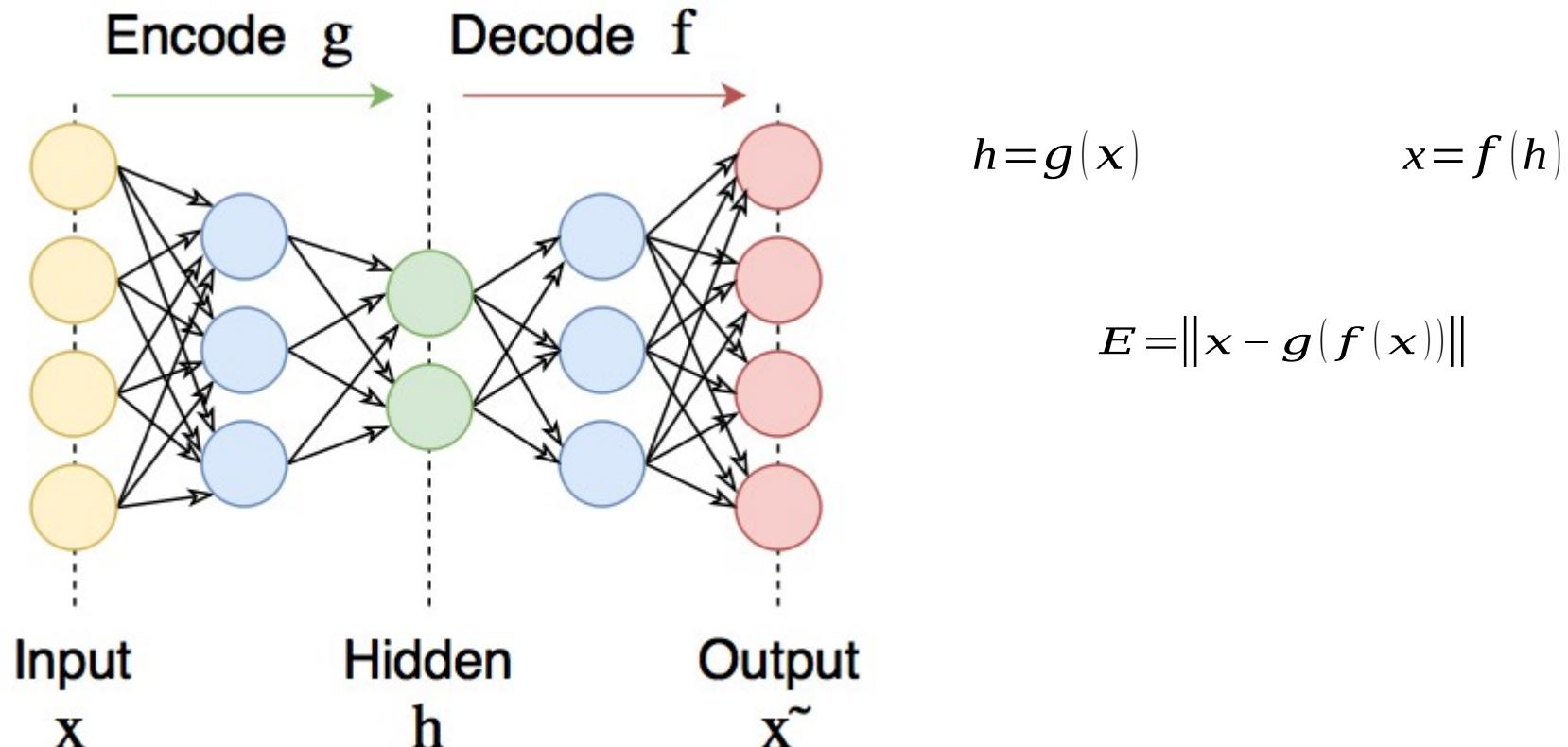
# Условный (conditional) GAN



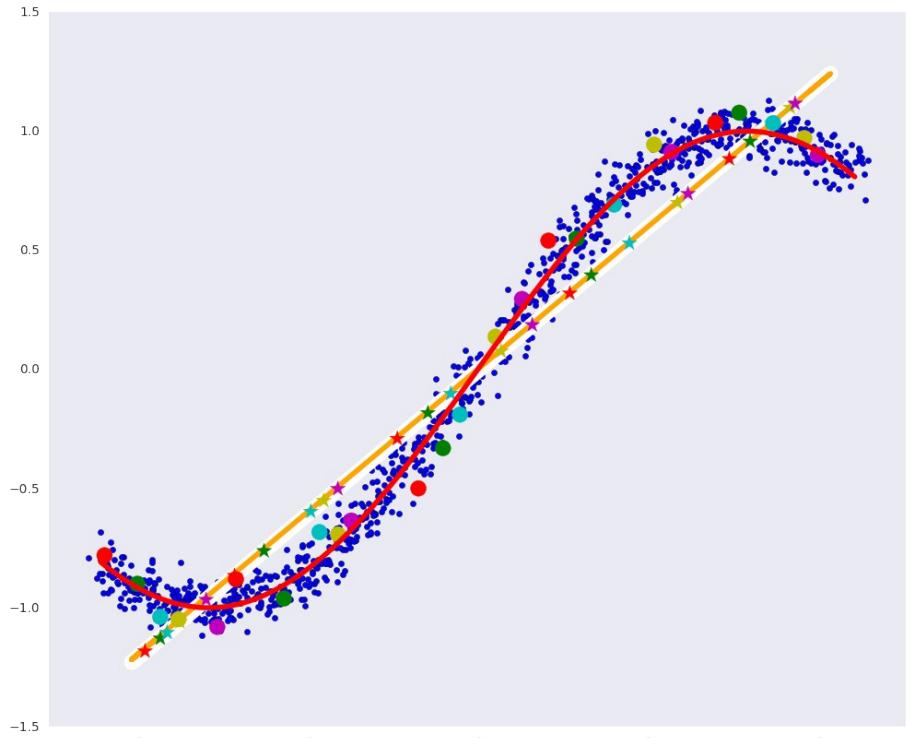
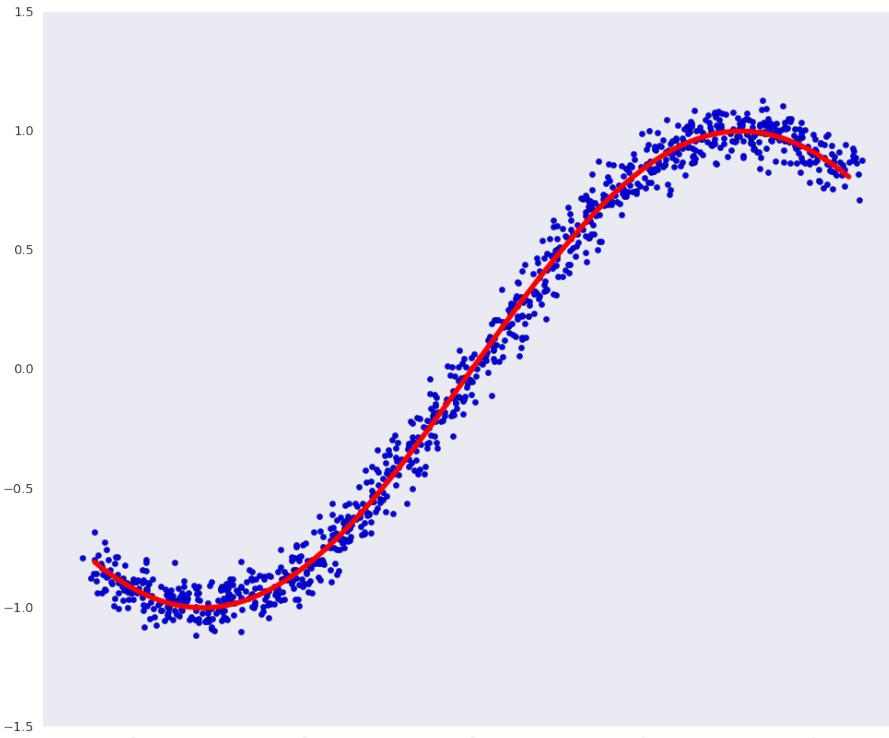
# АВТОЭКОДЕР



# АВТОЭКОДЕР

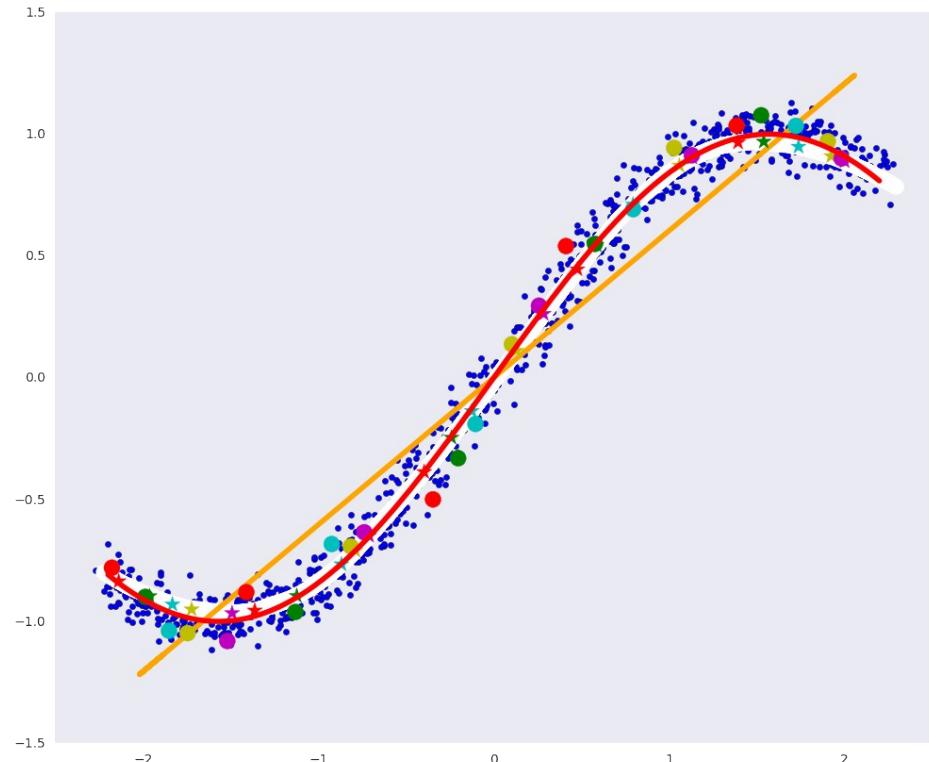
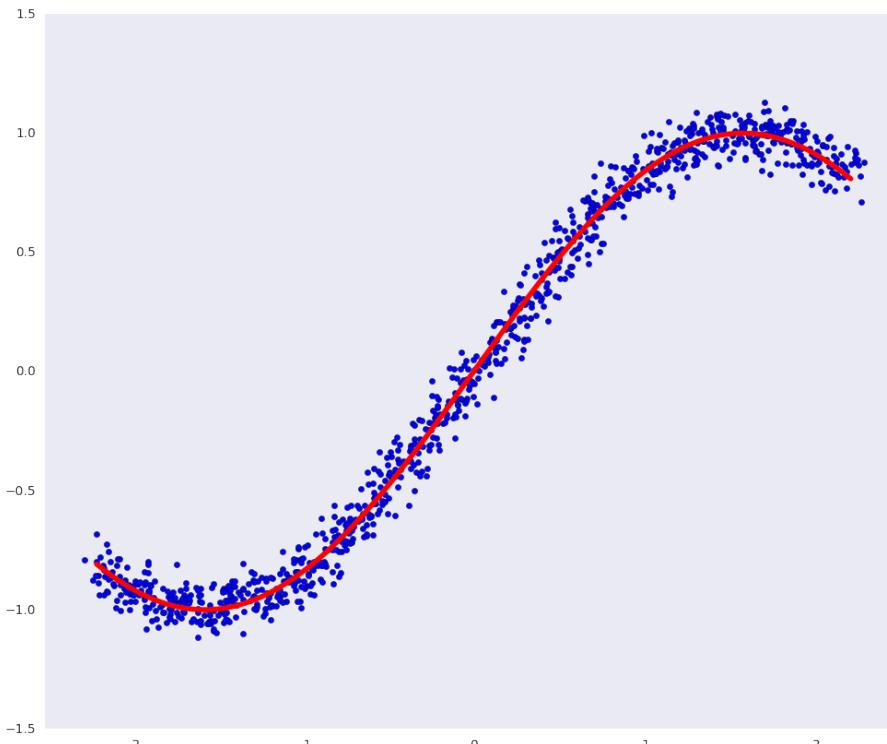


# Автоэнкодер: manifold learning



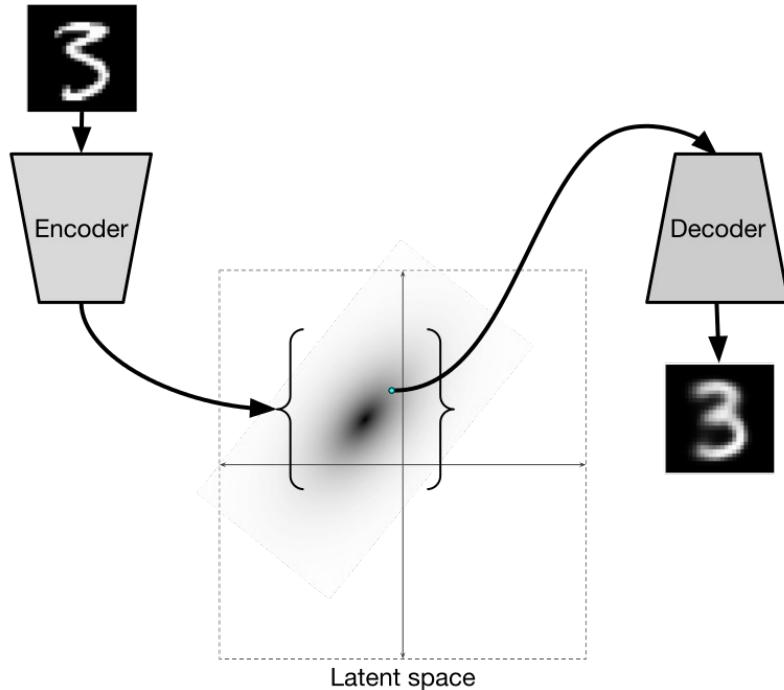
<https://habr.com/ru/post/331382/>

# Автоэнкодер: manifold learning

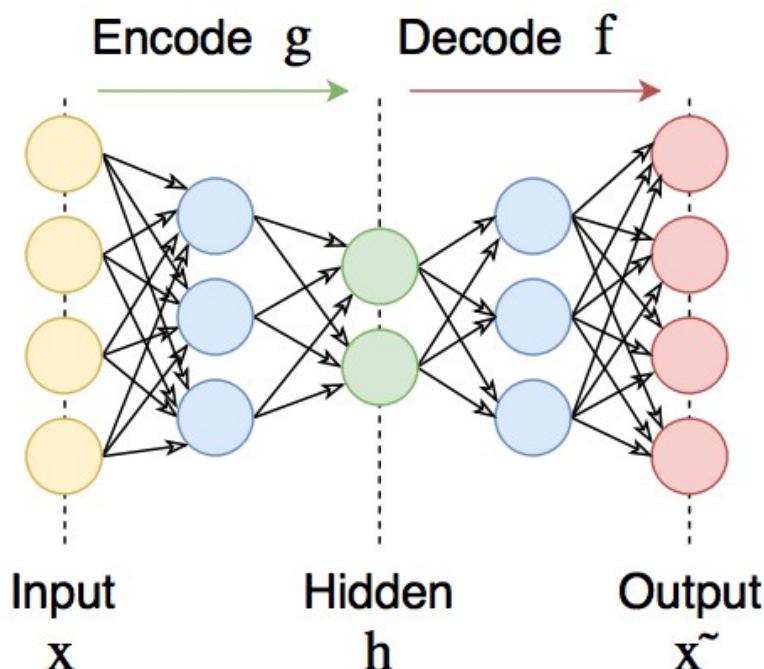


<https://habr.com/ru/post/331382/>

# VAE: Вариационный автоэнкодер



# VAE: Вариационный автоэнкодер



$$P(X) = \int_H P(X|h) P(h) dh \quad P(X|h) = f(h) + \varepsilon$$

$$P(X; \theta) = \int_H P(X|h; \theta) P(h) dh \quad P(X|h; \theta) = f(h; \theta) + \varepsilon$$

$$P(X|h; \theta) = N$$

# VAE: Вариационный автоэнкодер

$$P(X) = \int_H P(X|h) P(h) dh \quad P(X|h; \theta) = N$$

Выберем подмножество из которого мы получаем множество  $X$

Введем распределение которое даст нам те , которые привели к  $X$

$$KL[Q(H|X)\|P(H,X)] = E_{H \sim Q}[\log Q(H|X) - \log P(H|X)]$$

$$KL[Q(H|X)\|P(H,X)] = E_{H \sim Q}[\log Q(H|X) - \log P(X|H) - \log P(H)] + \log P(X)$$

$$KL[Q(H|X)\|P(H,X)] = KL[Q(H|X)\|P(H)] - E_{H \sim Q}[\log P(X|H)] + \log P(X)$$

$$\log P(X) - KL[Q(H|X)\|P(H,X)] = E_{H \sim Q}[\log P(X|H)] - KL[Q(H|X)\|P(H)]$$

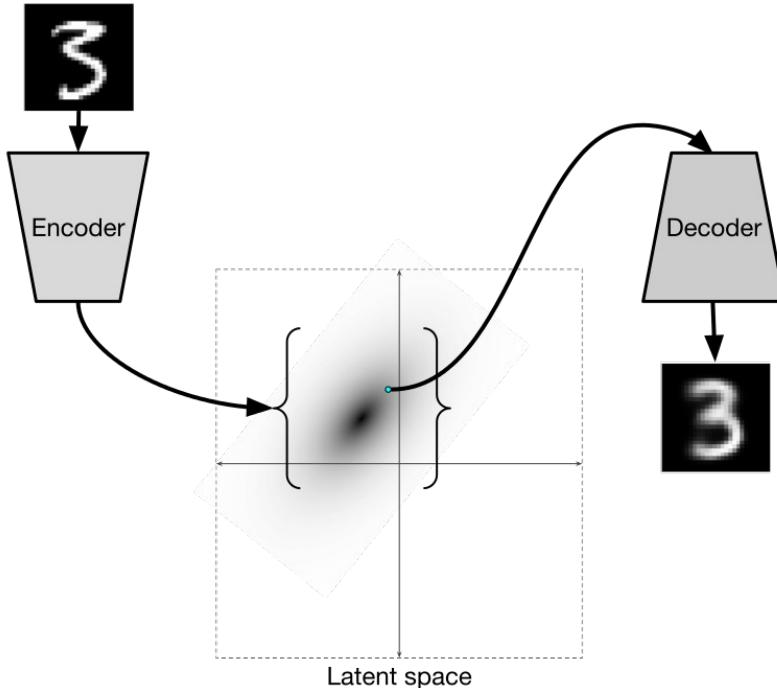
# VAE: Вариационный автоэнкодер

$$P(X) = \int_H P(X|h) P(h) dh$$

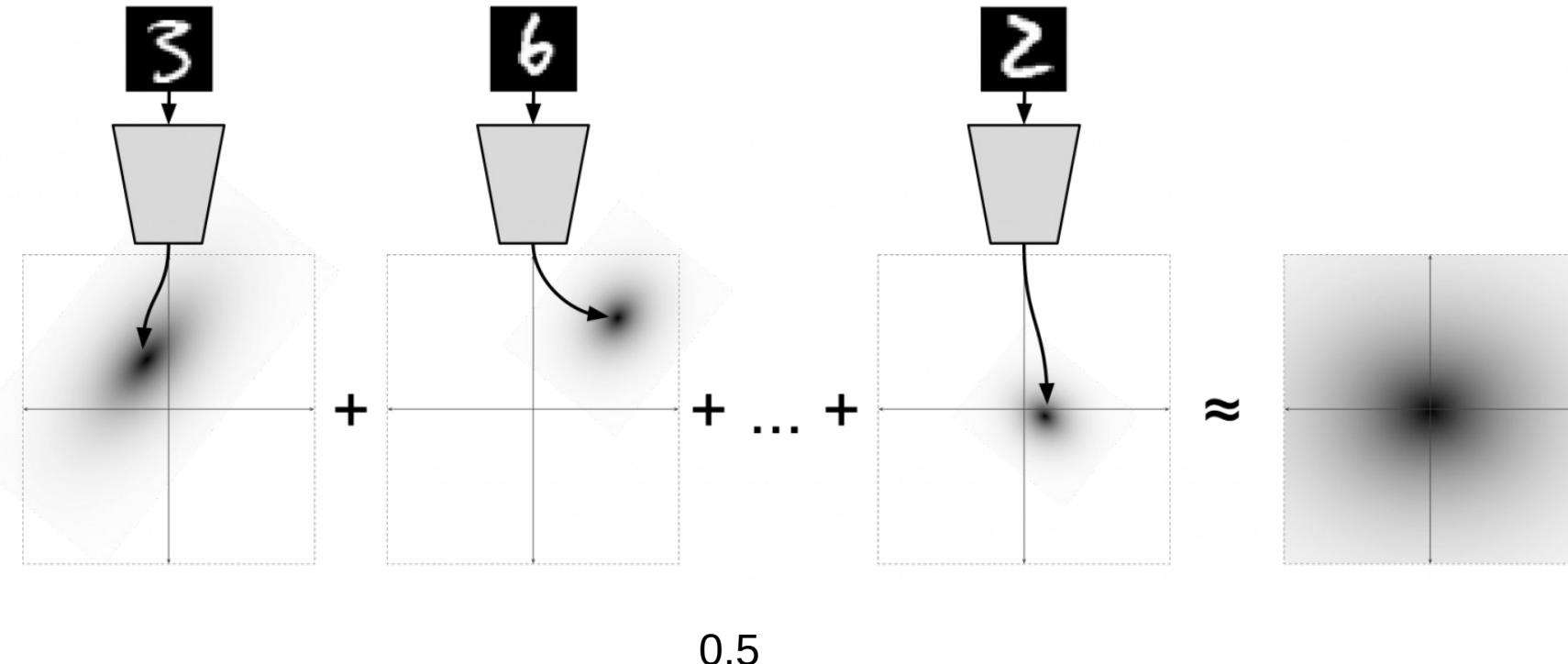
$$\log P(X) - KL[Q(H|X) \sim P(H, X)] = E_{H \sim Q}[\log P(X|H)] - KL[Q(H|X) \| P(H)]$$

$$\log P(X|\theta_2) - KL[Q(H|X; \theta_1) \| P(H, X; \theta_2)] = E_{H \sim Q}[\log P(X|H; \theta_2)] - KL[Q(H|X; \theta_1) \| N(0, I)]$$

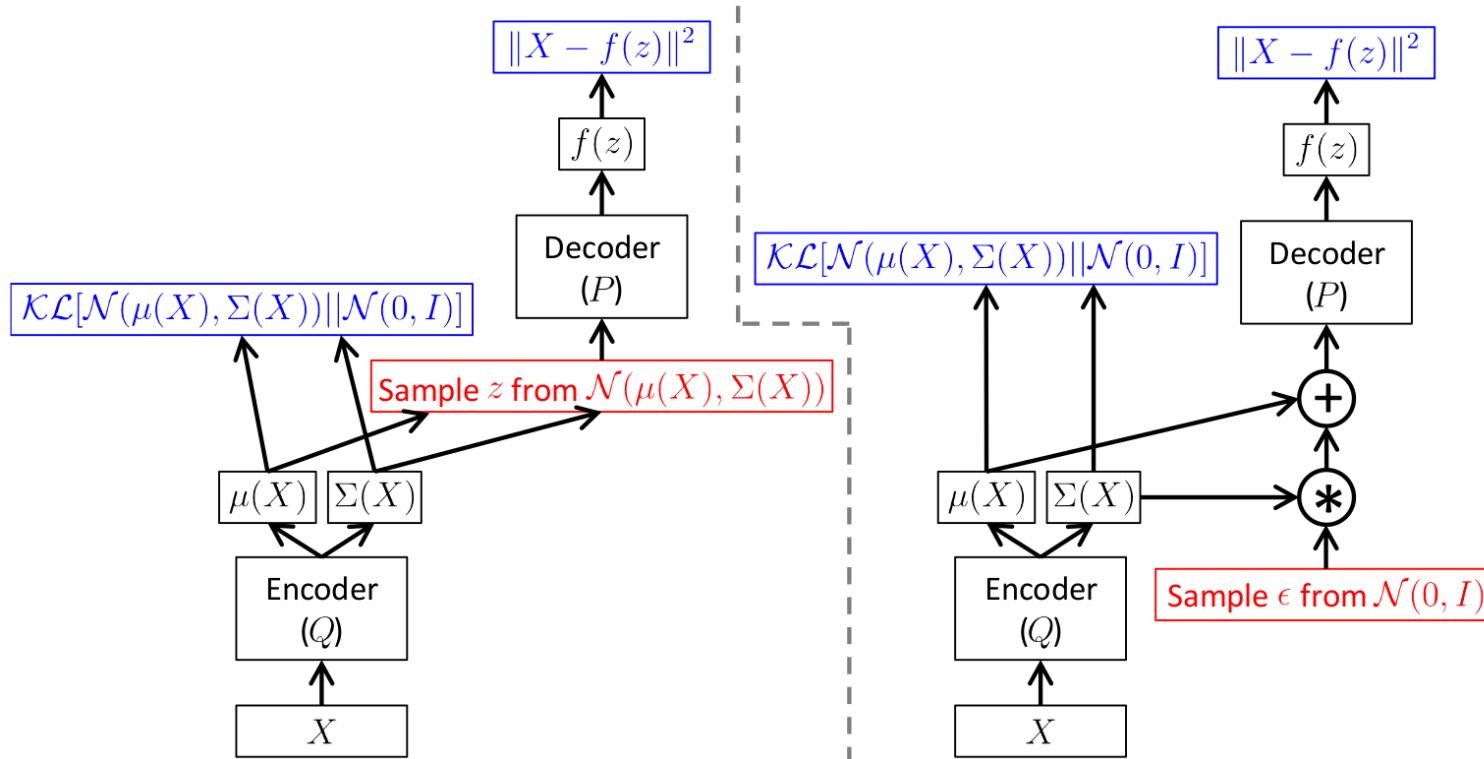
# VAE: Вариационный автоэнкодер



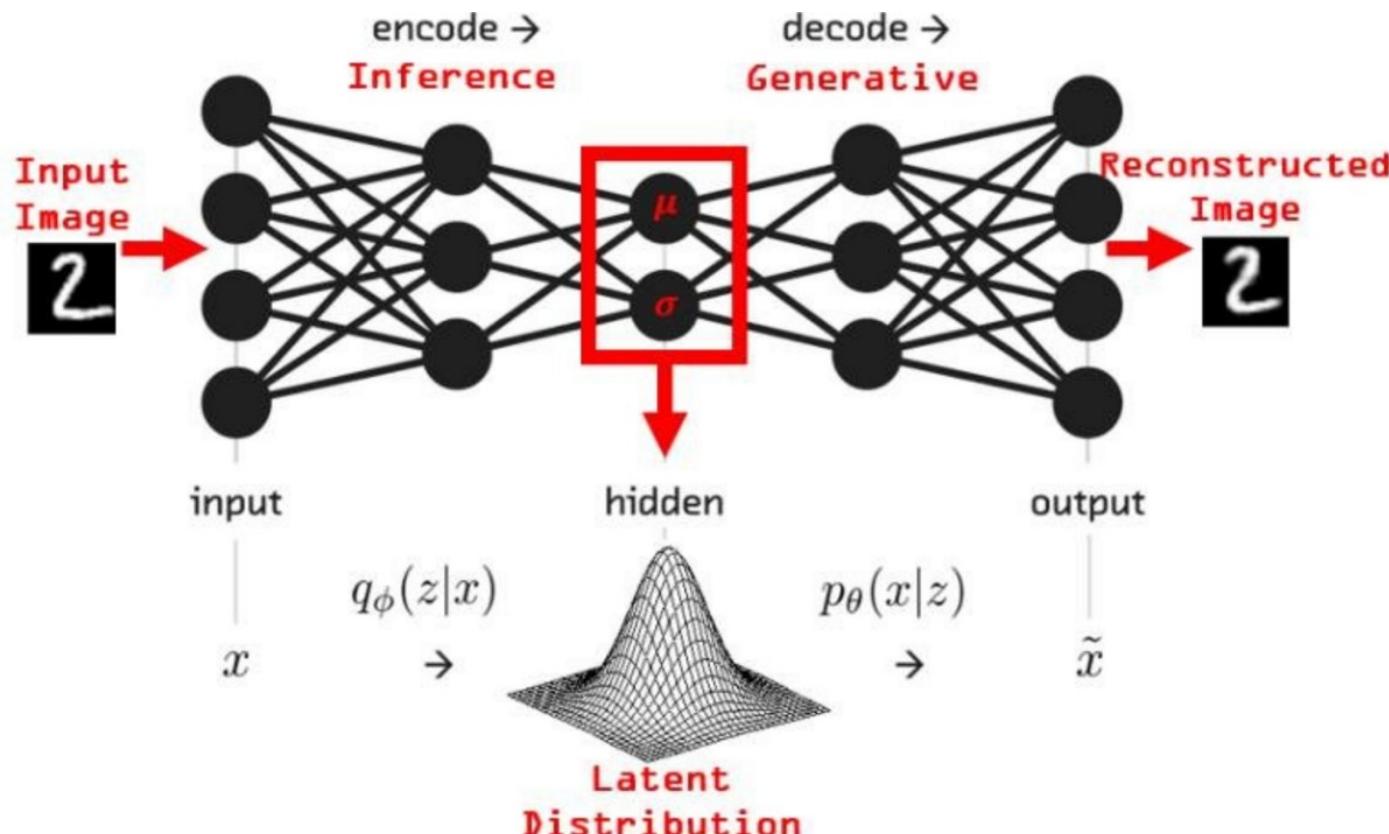
# VAE: Вариационный автоэнкодер



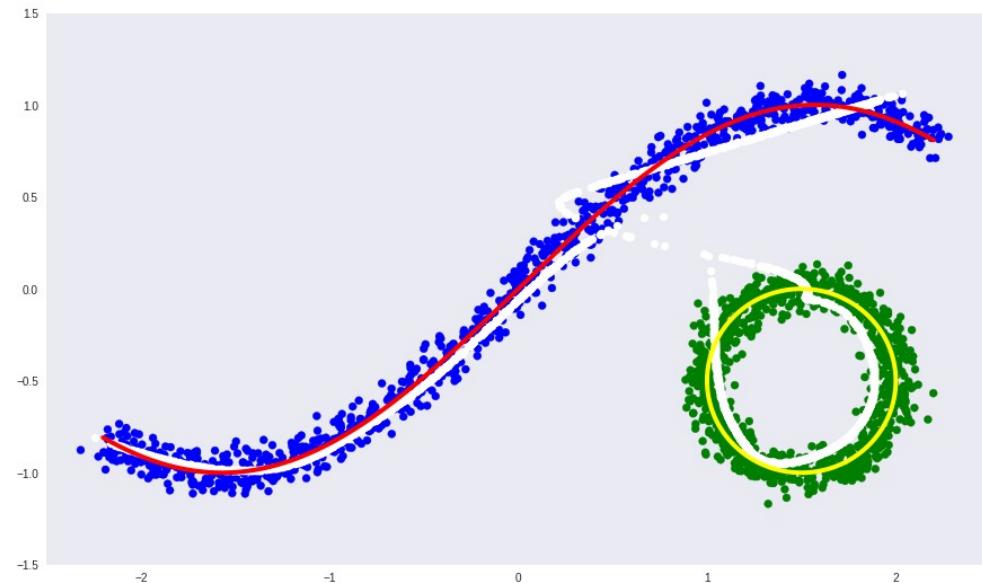
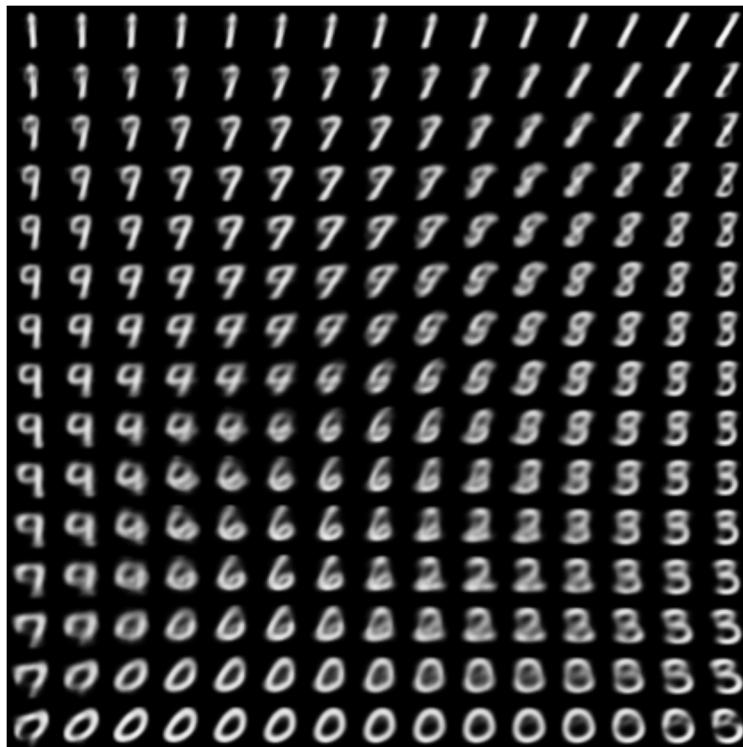
# VAE: Вариационный автоэнкодер



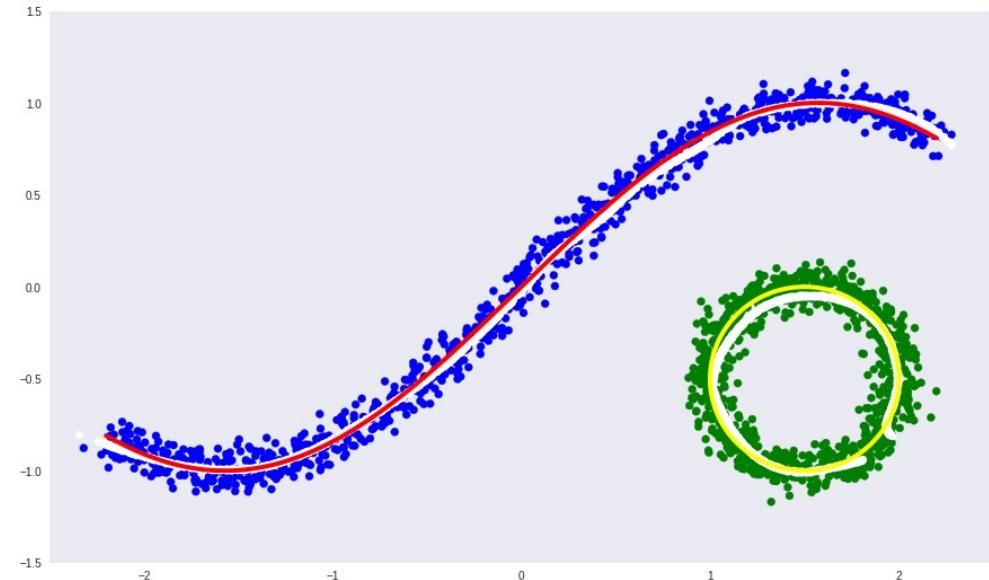
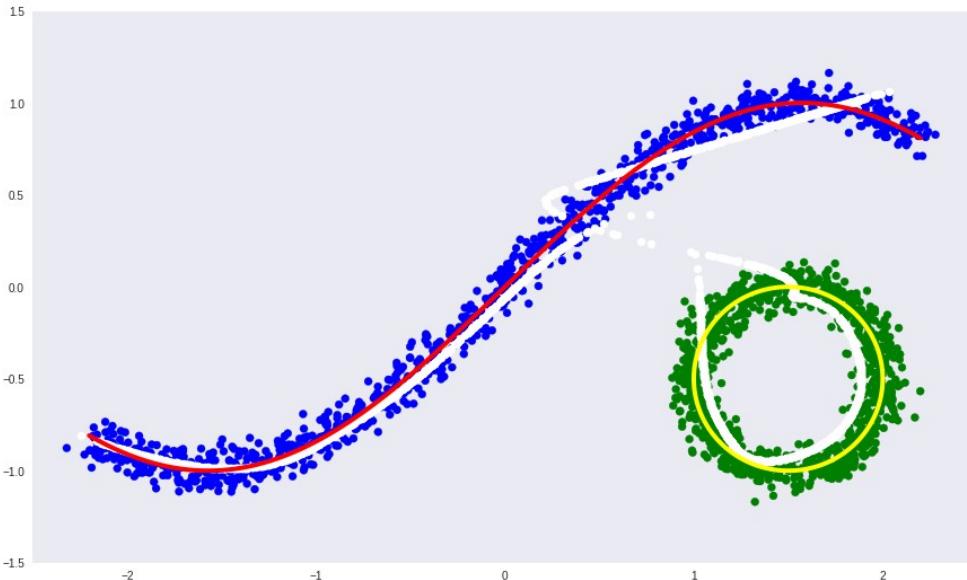
# VAE: Вариационный автоэнкодер



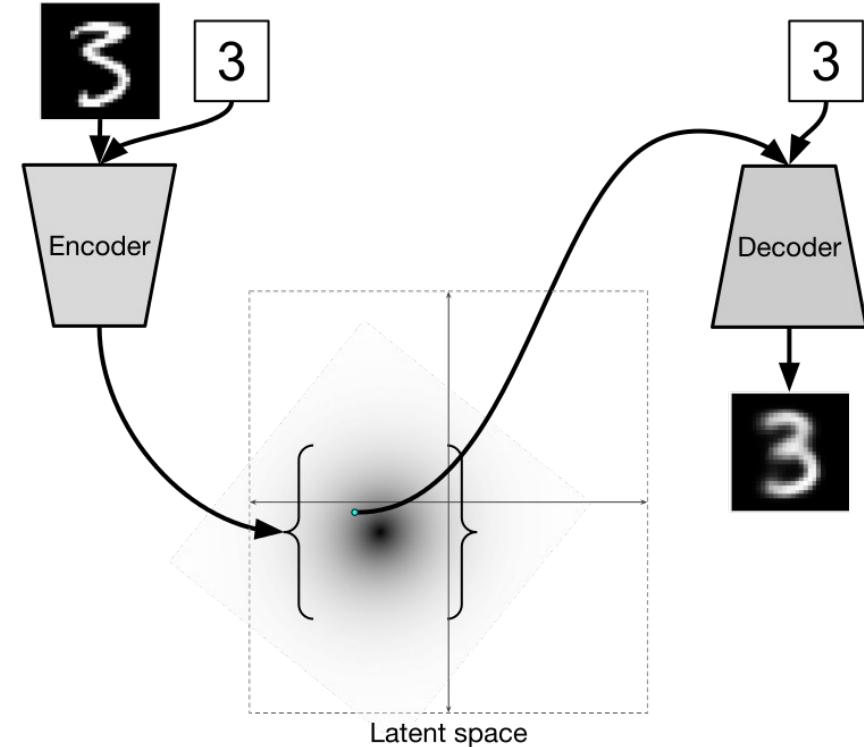
# VAE: недостатки



# Conditional VAE



# Conditional VAE



$$\log P(X|Y; \theta_2) - KL[Q(H|X, Y; \theta_1) \| P(H|X, Y; \theta_2)] = E_{H \sim Q}[\log P(X|H, Y; \theta_2)] - KL[Q(H|X, Y; \theta_1) \| N(0, I)]$$

# Conditional VAE: перенос стиля

- Обучаем CVAE на картинках с метками
- Кодируем стиль заданной картинки в  $H$
- Меняем метки  $Y$ , создаем из закодированного  $H$  новые картинки

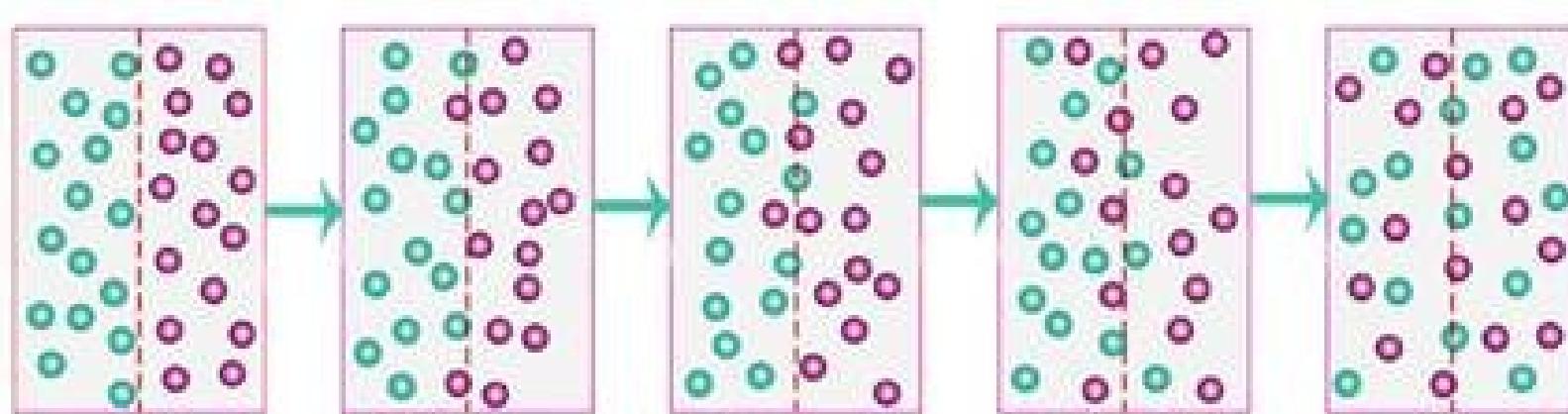


# GAN+VAE

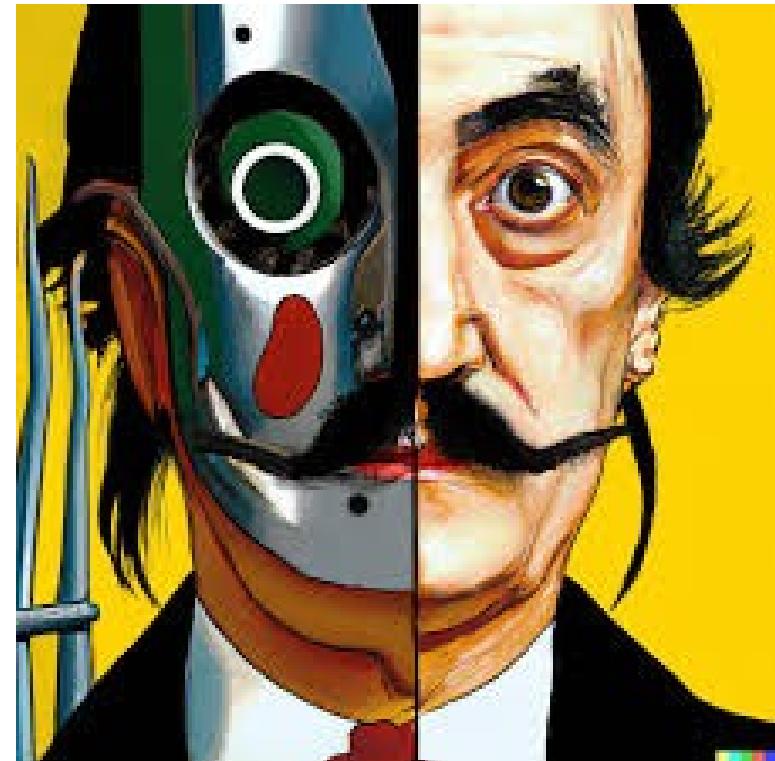
<https://github.com/csinva/gan-vae-pretrained-pytorch>

<https://github.com/rishabh786/VAE-GAN-PYTORCH>

# Диффузионные модели



# DALL – E2 (OpenAI)



# Imagen (Google)



*"A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat."*

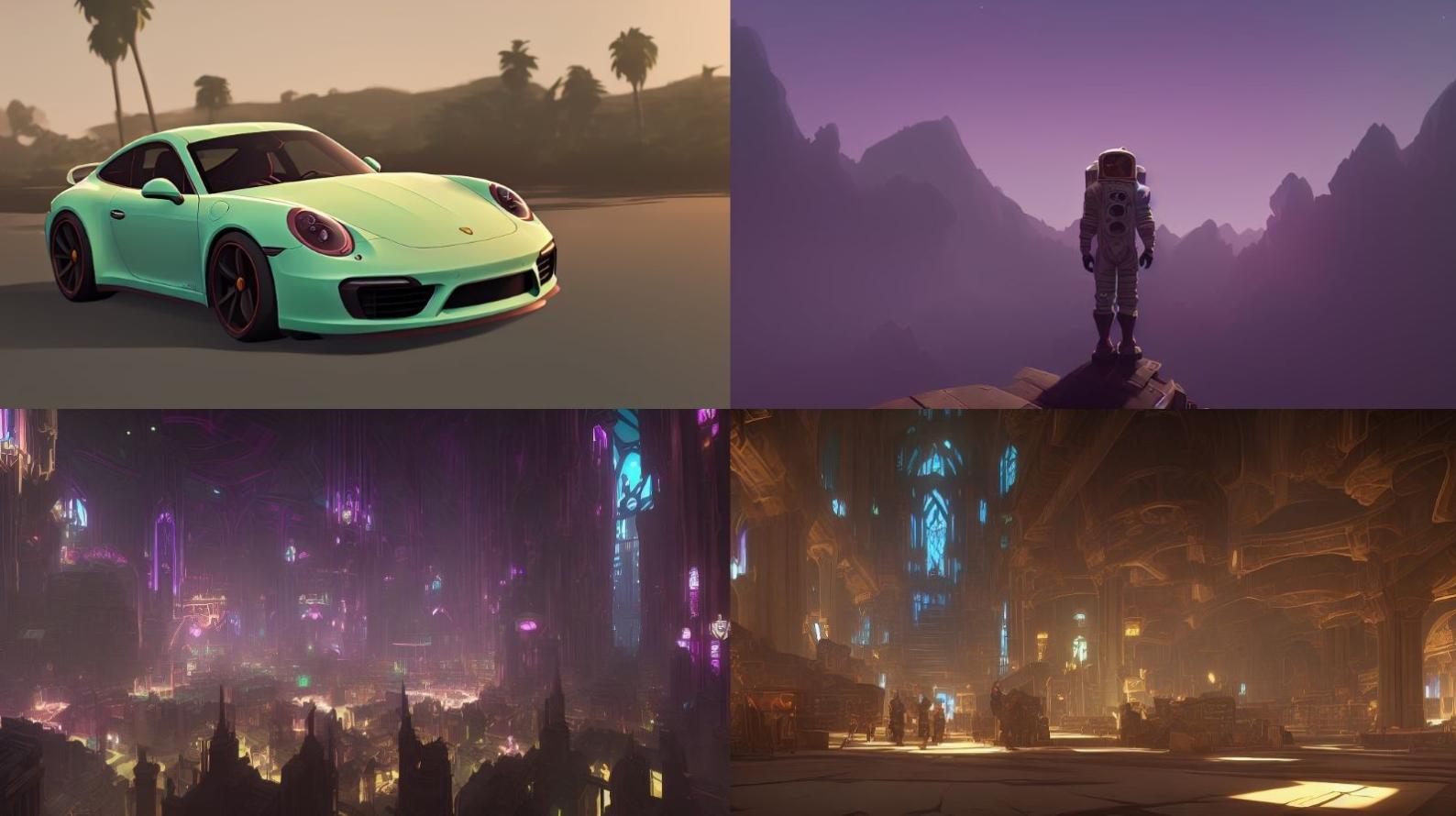


*"A blue jay standing on a large basket of rainbow macarons"*



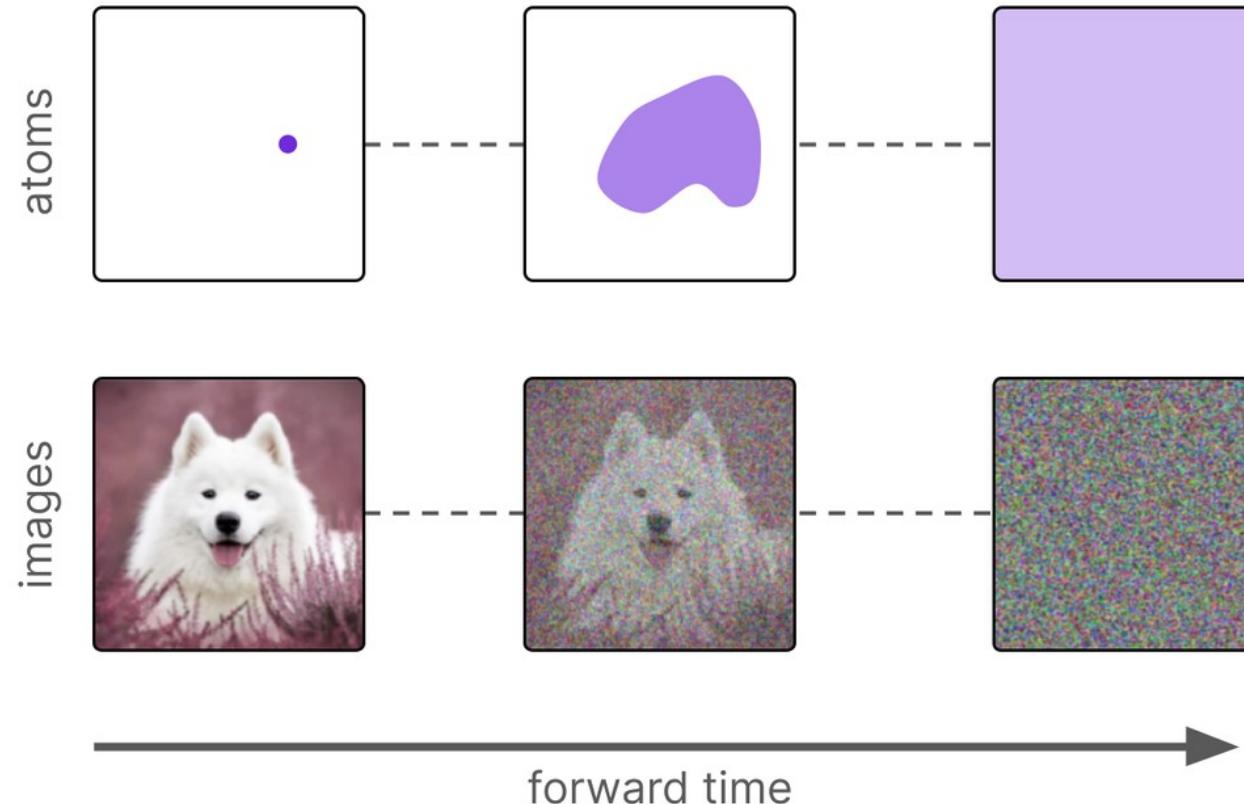
*"A brain riding a rocketship heading towards the moon"*

# Stable Diffusion (StabilityAI)



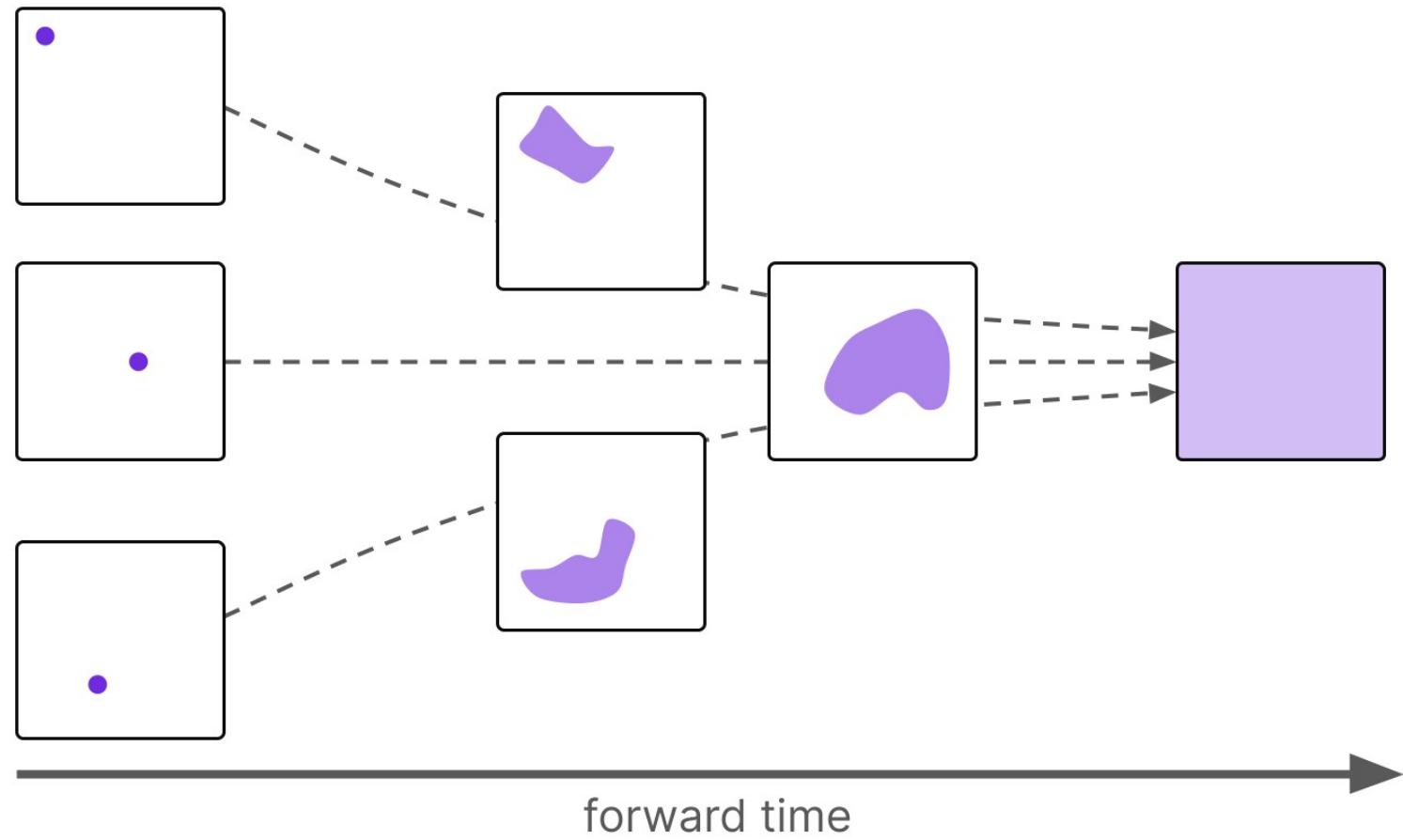
<https://learnopencv.com/image-generation-using-diffusion-models/>

# Диффузионные модели: идея

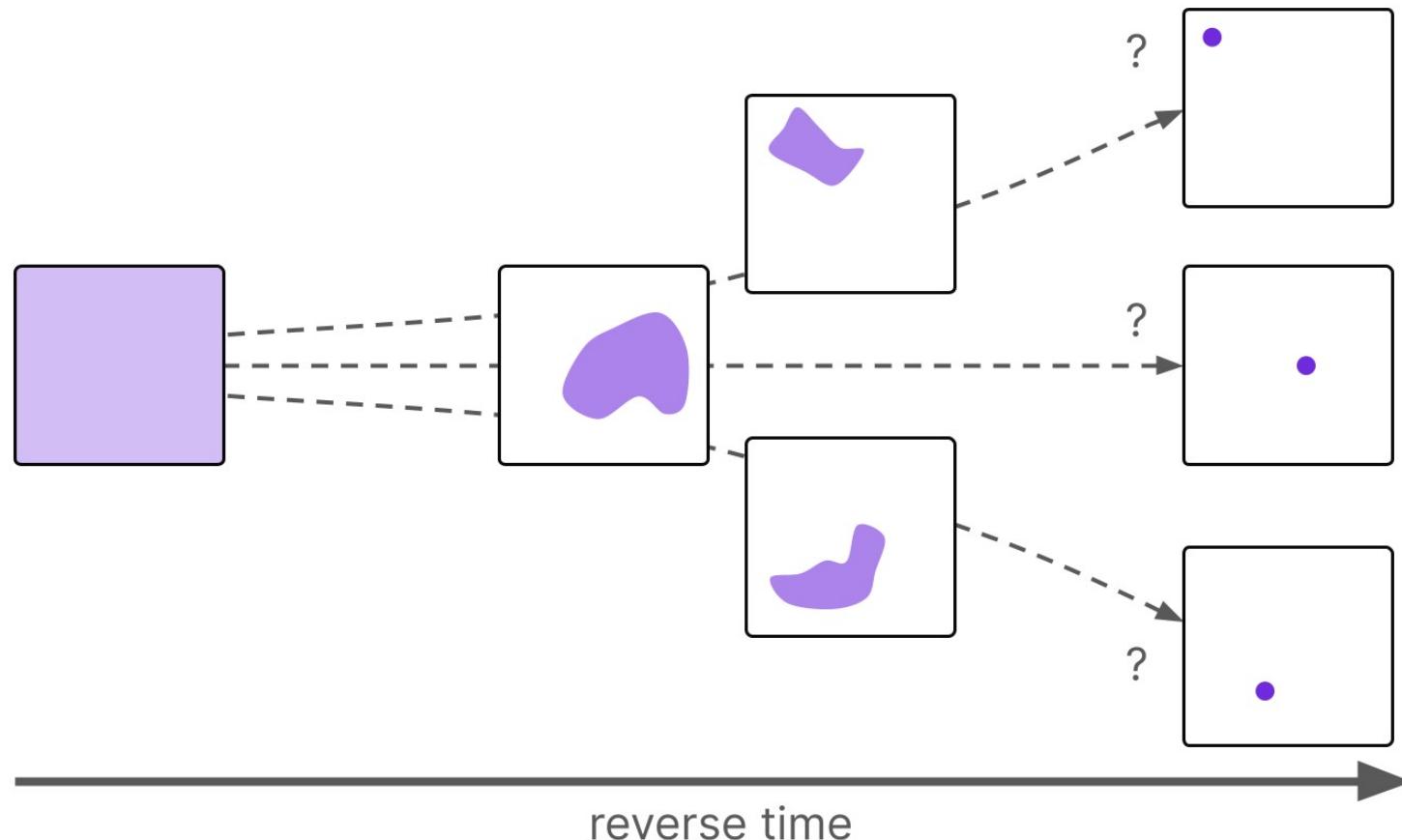


<https://www.assemblyai.com/blog/how-physics-advanced-generative-ai/#generative-ai-with-thermodynamics>

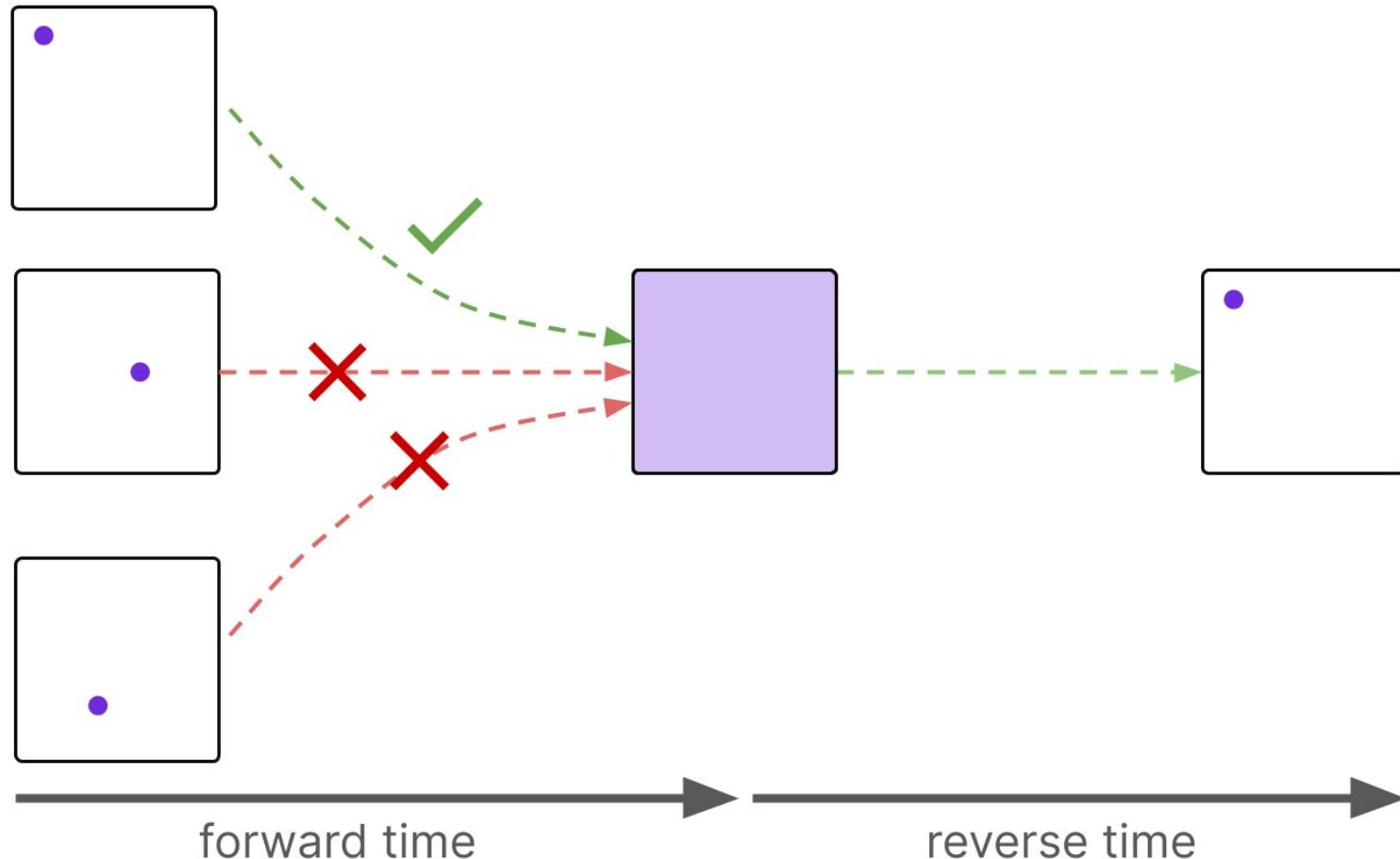
# Диффузионные модели: идея



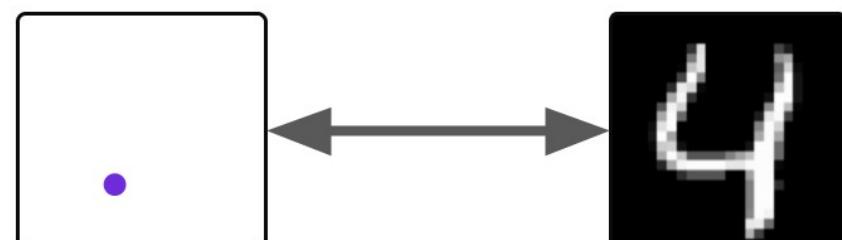
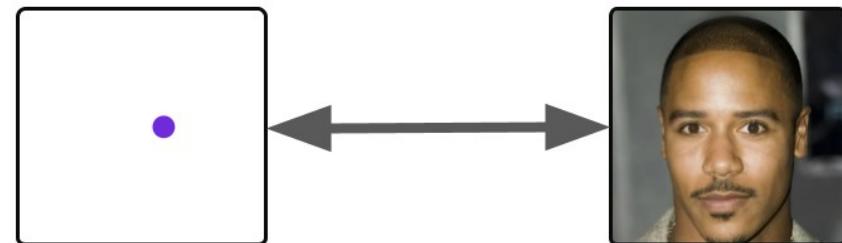
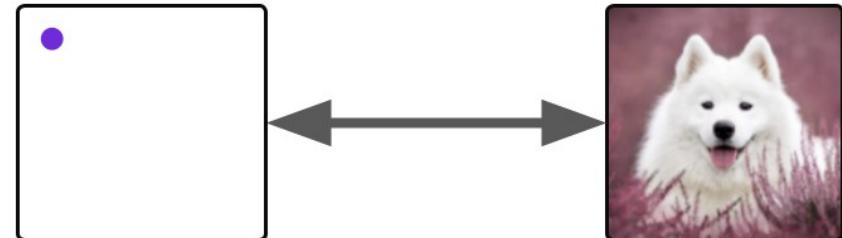
# Диффузионные модели: идея



# Диффузионные модели: идея



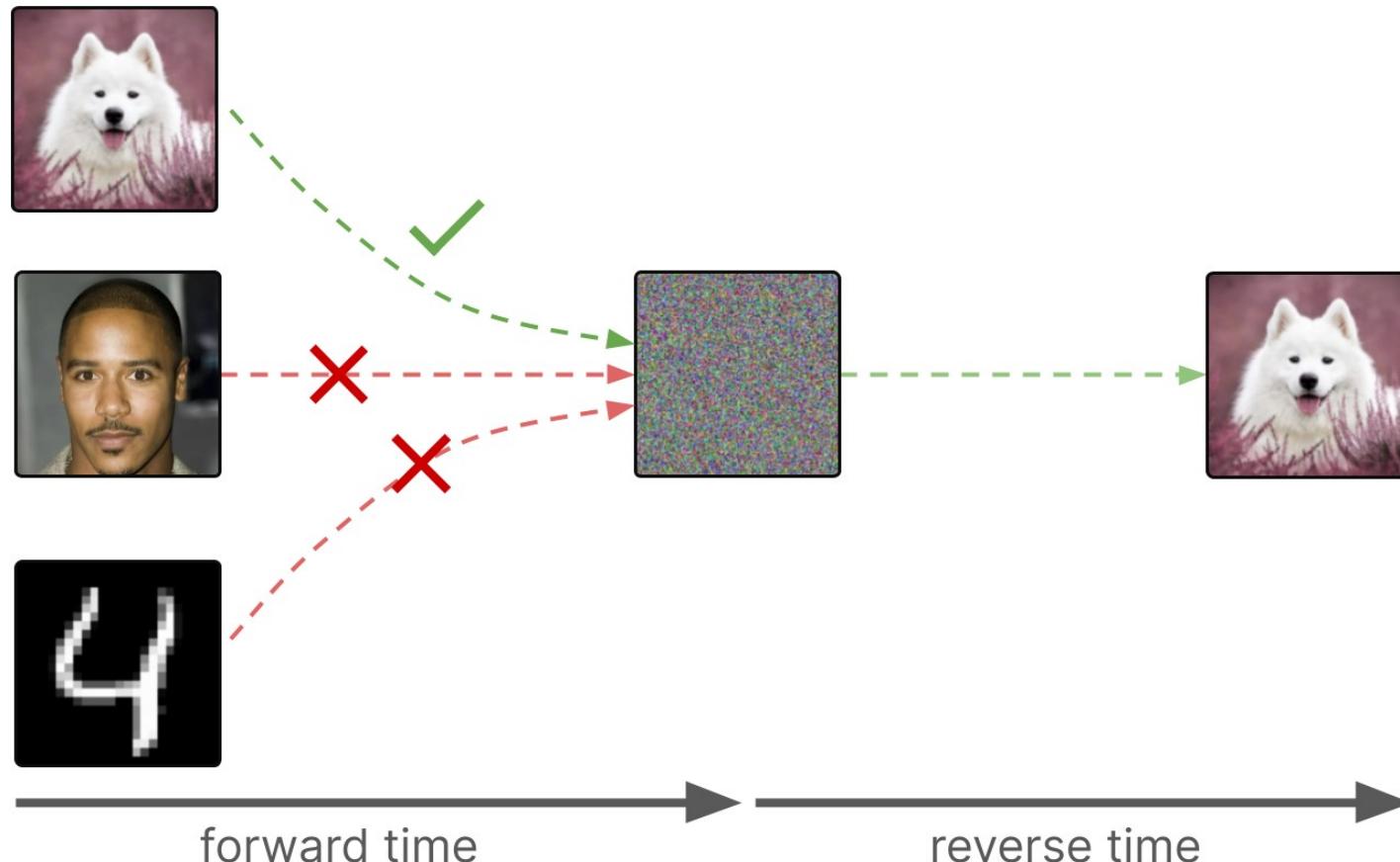
# Диффузионные модели: идея



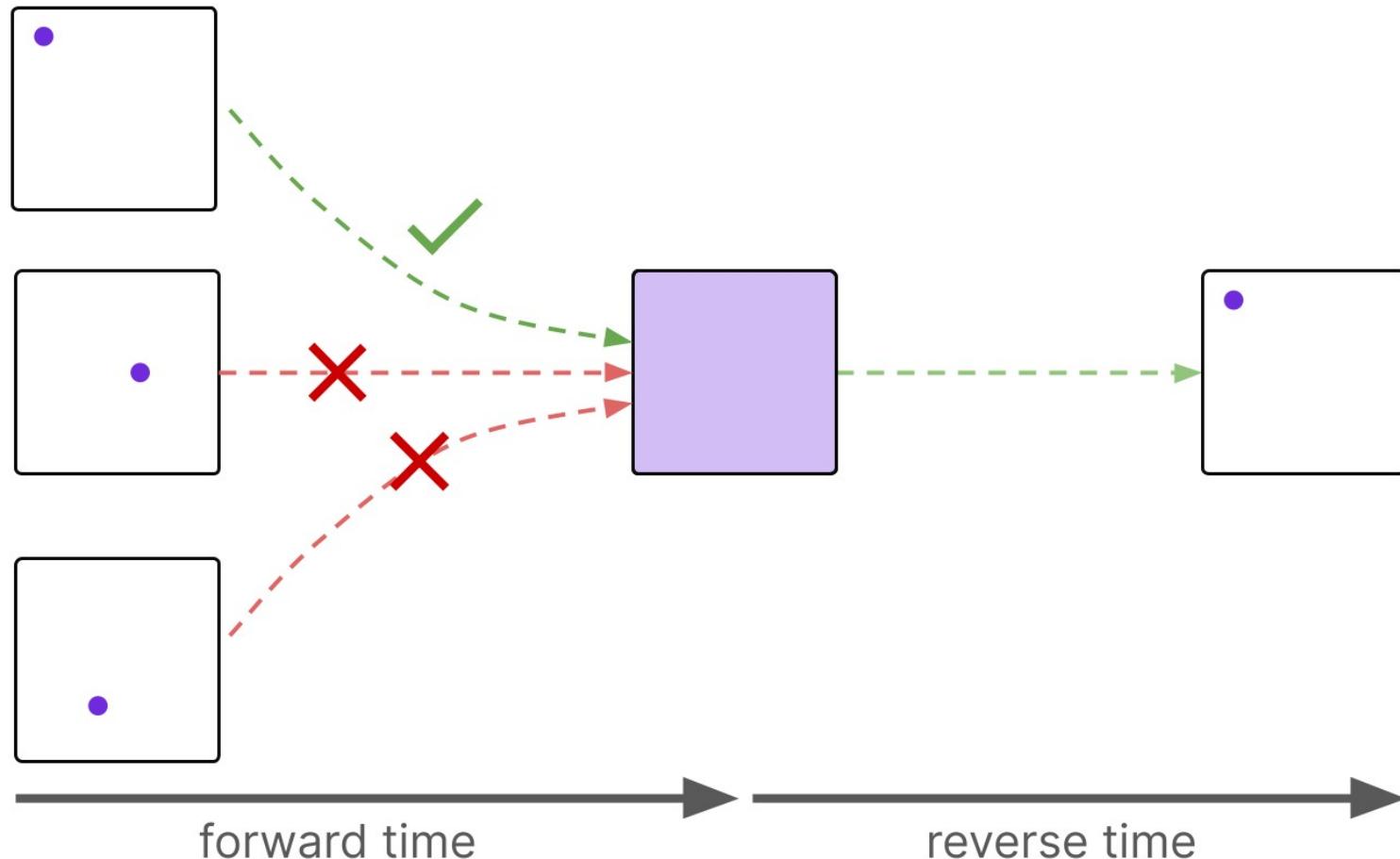
drops

image equivalents

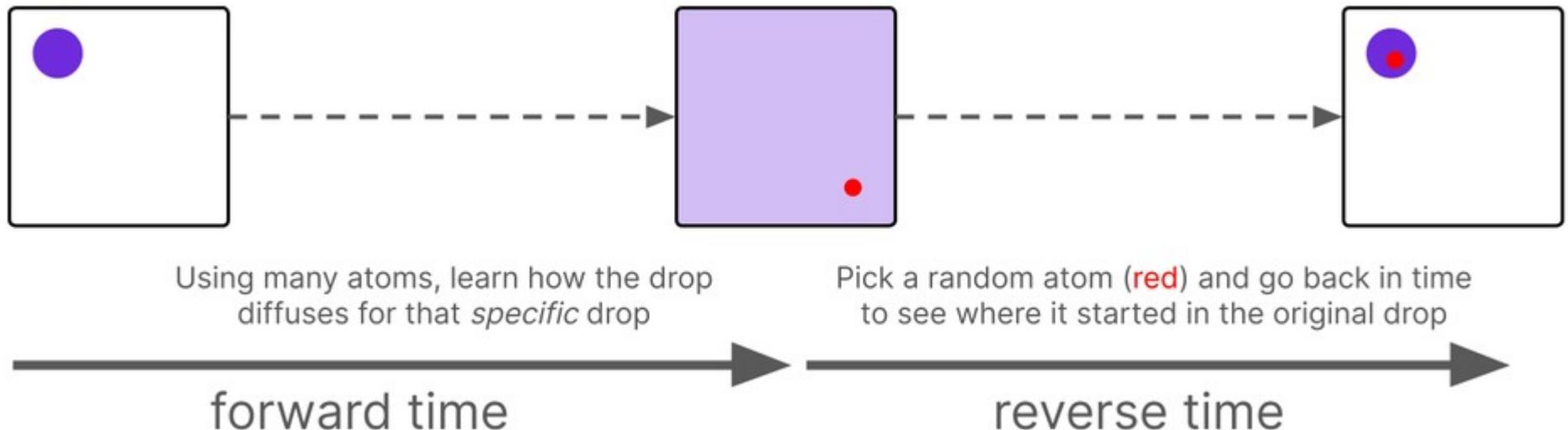
# Диффузионные модели: идея

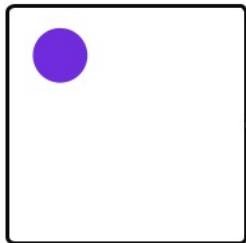


# Диффузионные модели: идея

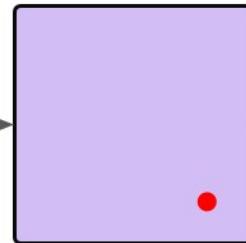


# Диффузионные модели: идея

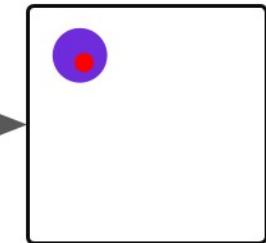




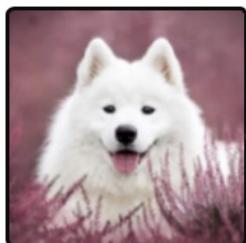
forward time



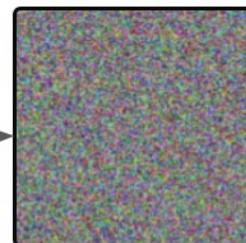
reverse time



reverse time



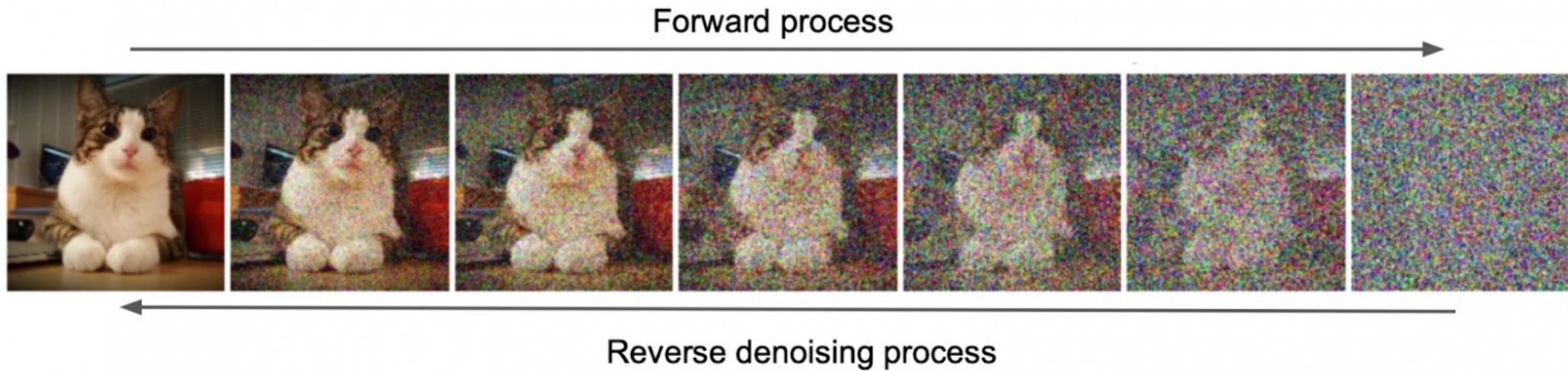
Using many images, learn how they diffuse for that *specific* type of image



Pick a random image of TV static and go back in time to see what image led to it



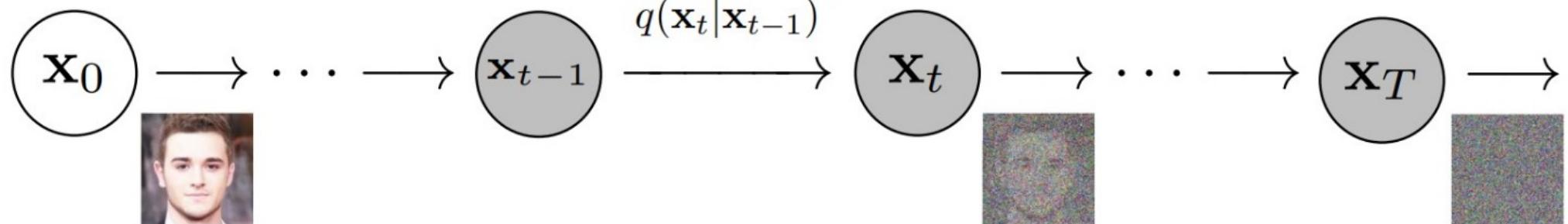
# Диффузионные модели: идея



<https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/>

# Диффузионные модели: зашумление

Мы сами задаем как будем добавлять шум

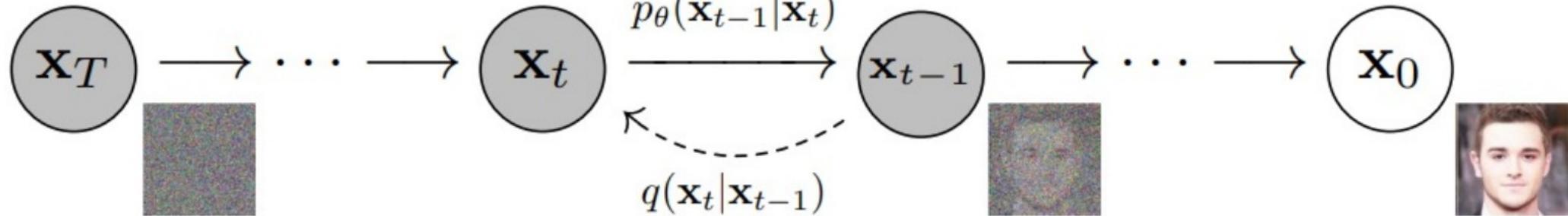


$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \boldsymbol{\Sigma}_t = \beta_t \mathbf{I})$$

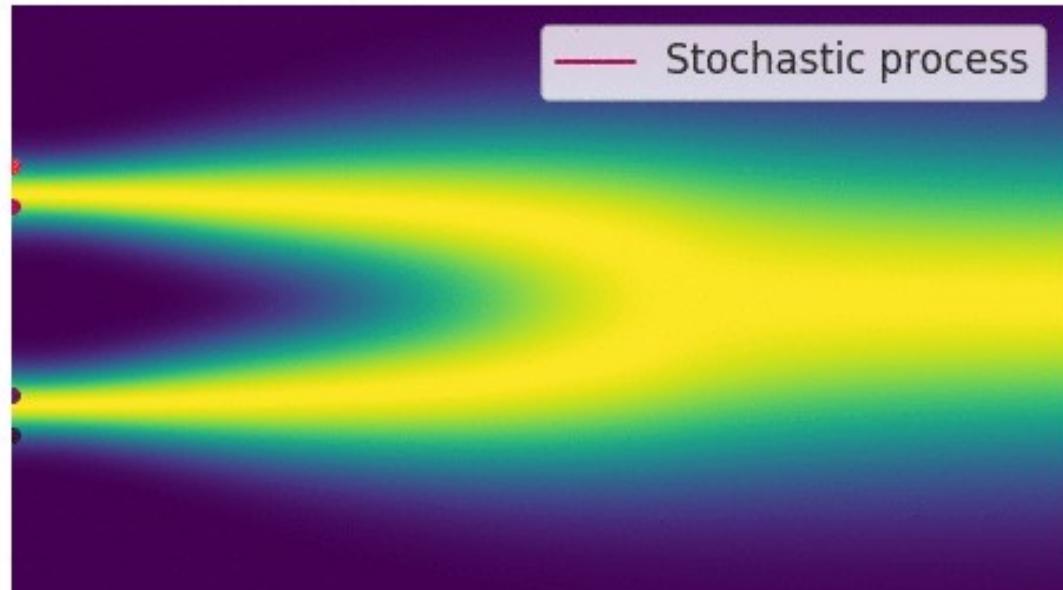
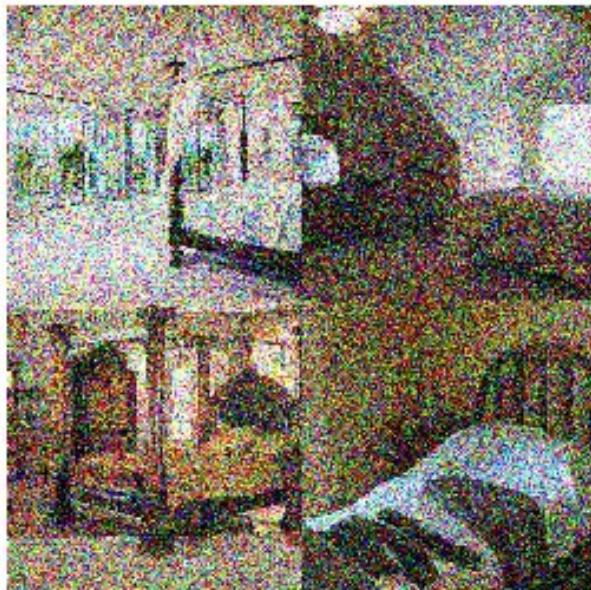
Константа

# Диффузионные модели: очистка от шума

Учим с помощью нейронной  
сети

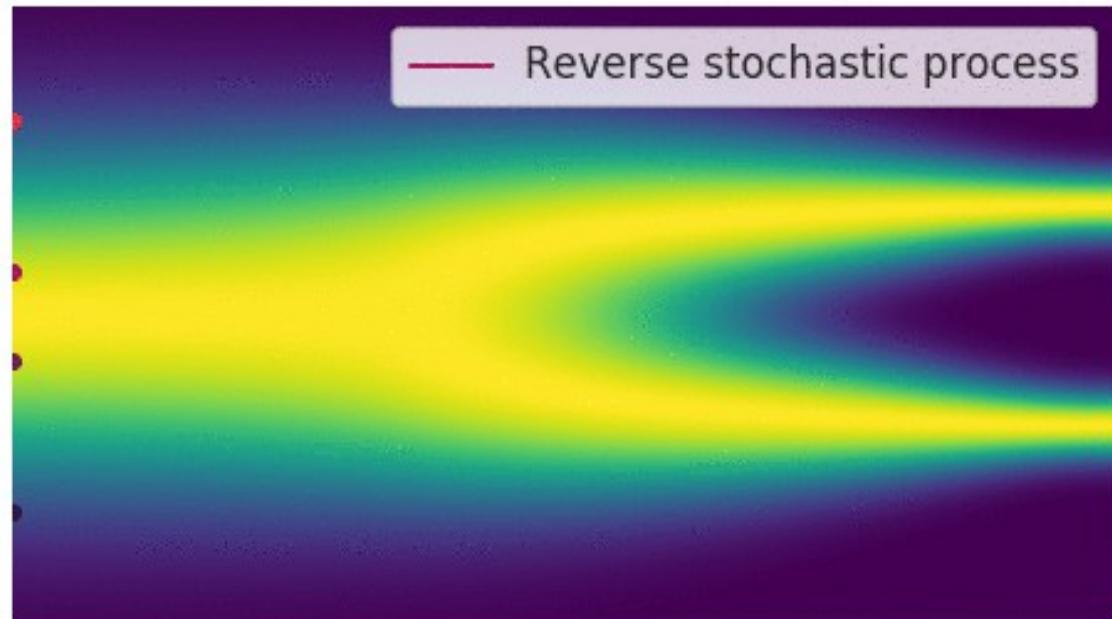
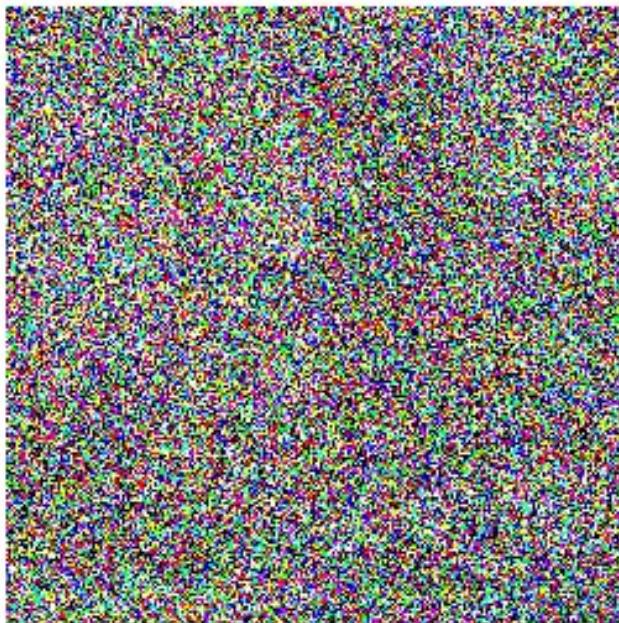


# Диффузионные модели: demo



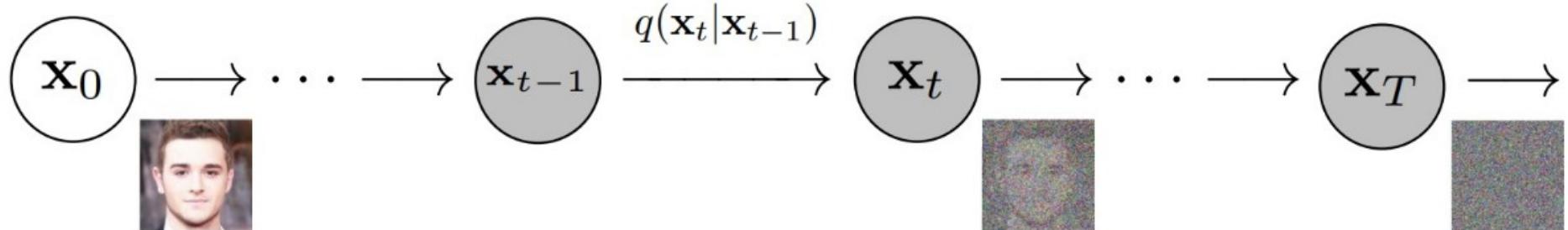
<https://yang-song.net/blog/2021/score/>

# Диффузионные модели: demo



<https://yang-song.net/blog/2021/score/>

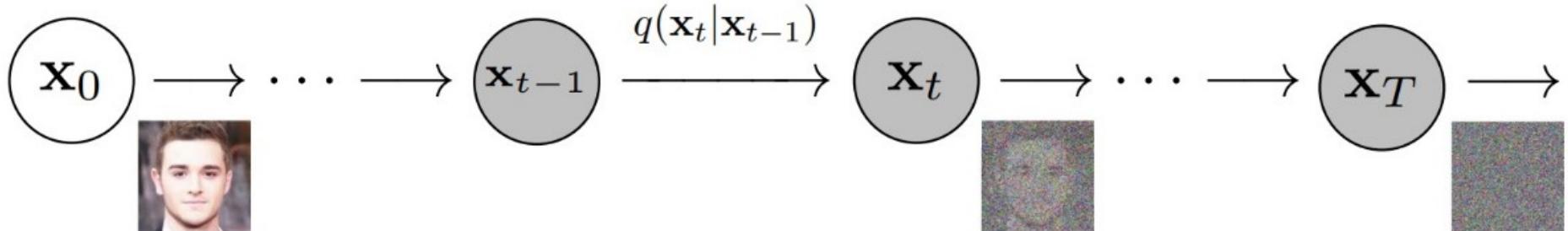
# Диффузионные модели: зашумление



►  $x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1 - \alpha_t}\epsilon_t$        $\epsilon_t \sim N(0, I), \quad \alpha_t = 1 - \beta_t$

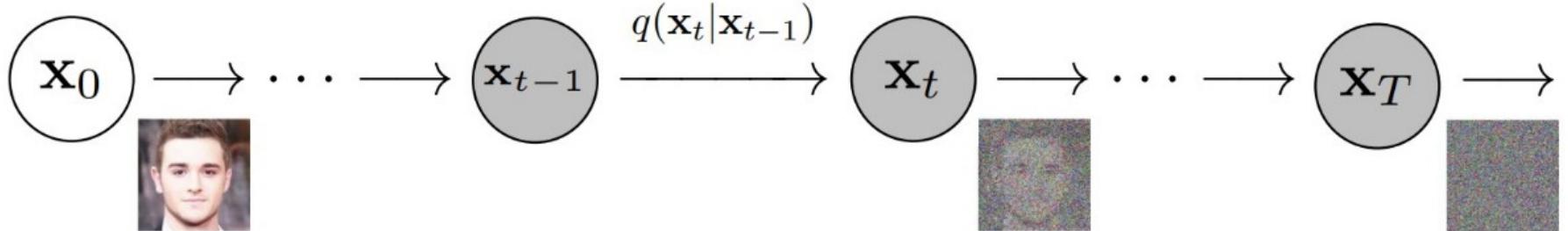
Мы сами так задали процесс

# Диффузионные модели: зашумление



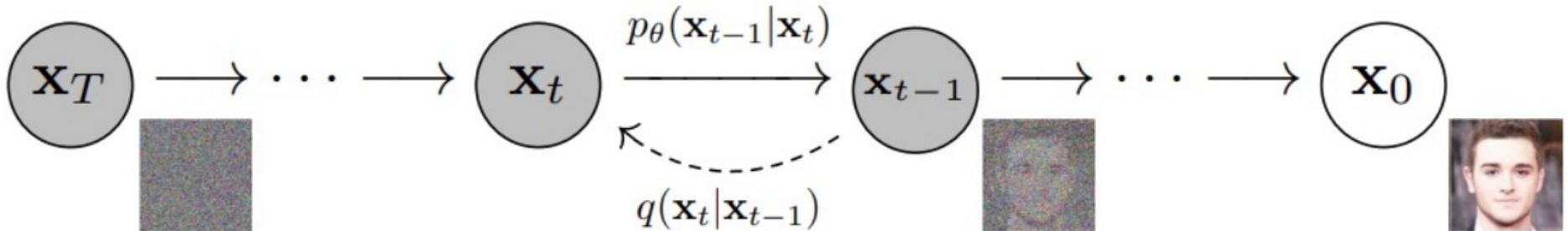
- ▶  $x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1 - \alpha_t}\epsilon_t \quad \epsilon_t \sim N(0, I), \quad \alpha_t = 1 - \beta_t$
- ▶  $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon \quad \epsilon \sim N(0, I), \quad \bar{\alpha}_t = \prod_{i=1}^t \alpha_t$

# Диффузионные модели: зашумление



- ▶  $x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1 - \alpha_t}\epsilon_t \quad \epsilon_t \sim N(0, I), \quad \alpha_t = 1 - \beta_t$
- ▶  $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon \quad \epsilon \sim N(0, I), \quad \bar{\alpha}_t = \prod_{i=1}^t \alpha_t$
- ▶  $x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right) + \tilde{\beta}_t z \quad z \sim N(0, I), \quad \tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t$

# Диффузионные модели: очистка

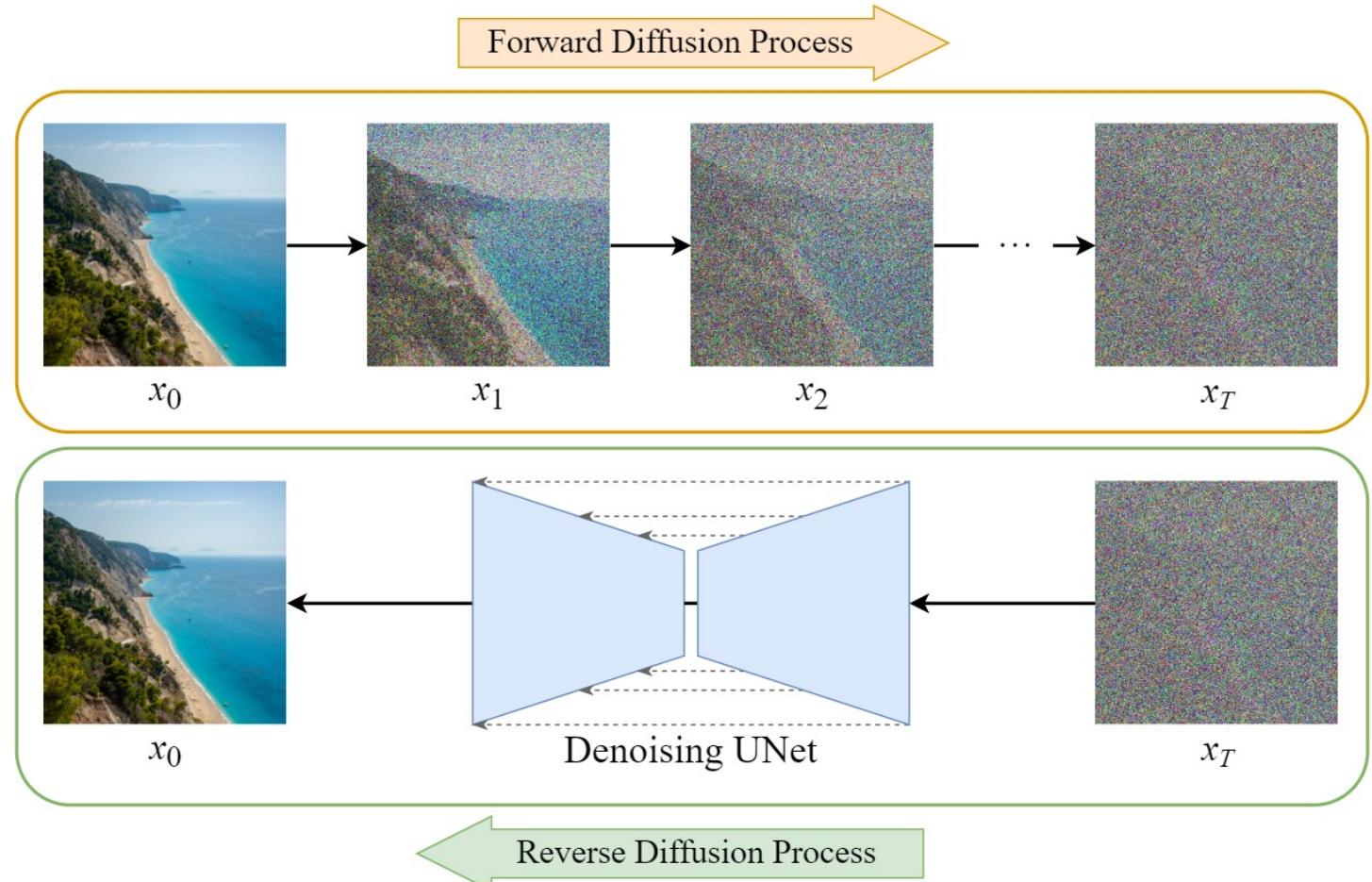


- ▶  $\hat{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t z$        $z \sim N(0, I), \quad \sigma_t = const$

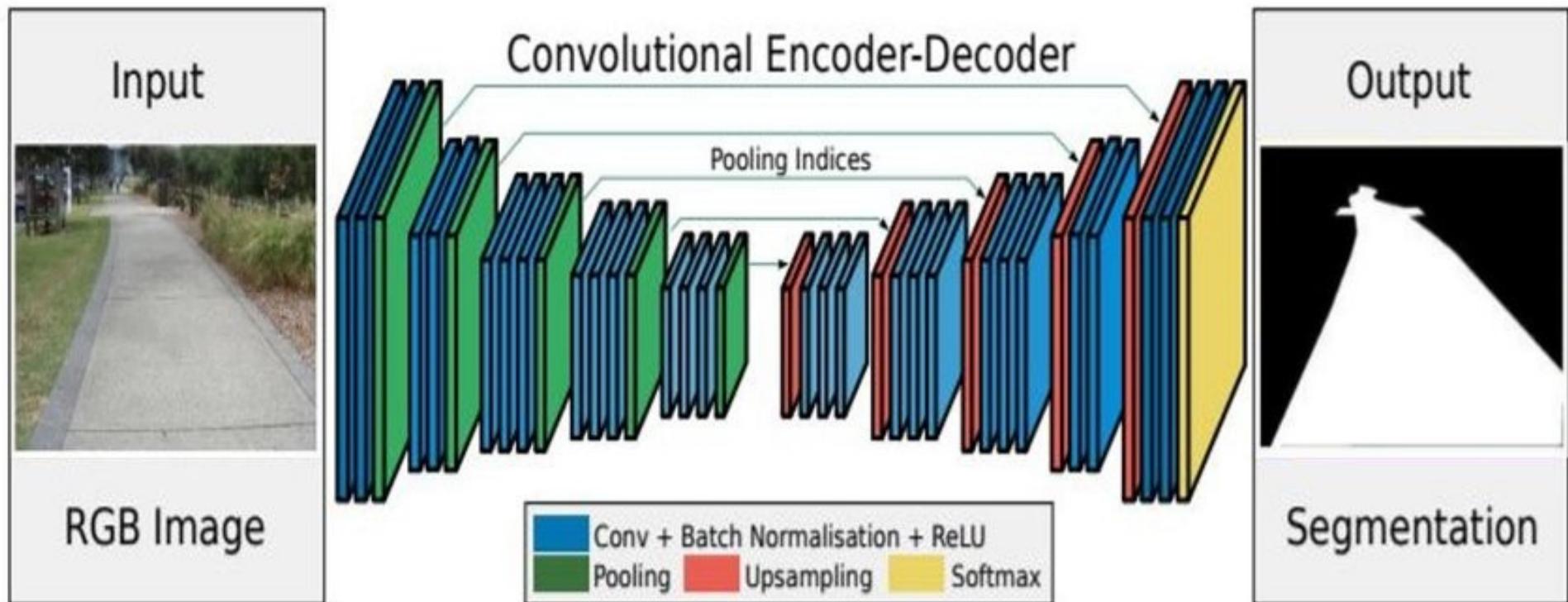
- ▶ Функция потерь для обучения:

$$L_t = \|x_{t-1} - \hat{x}_{t-1}\|_2^2 \propto \|\epsilon - \epsilon_\theta(x_t, t)\|_2^2 \rightarrow \min_{\theta}$$

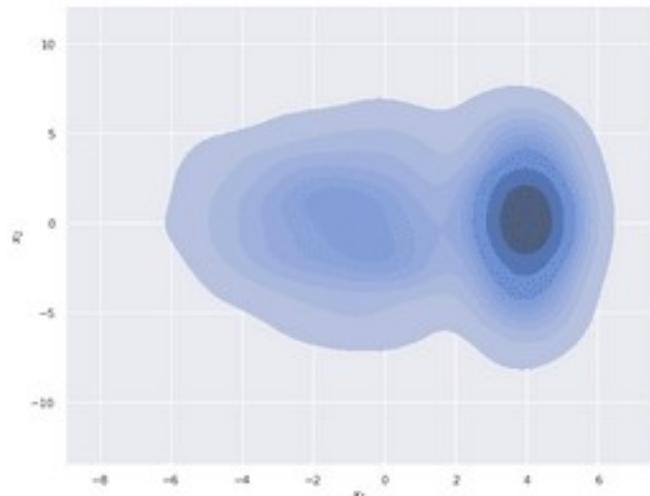
# Архитектура



# Архитектура

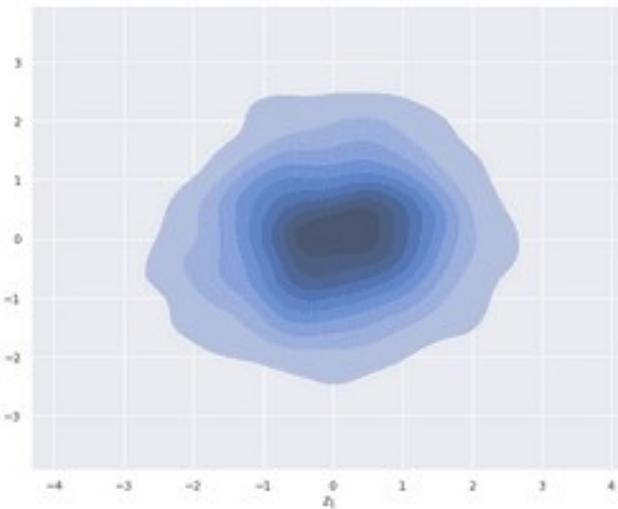


# Номализующие потоки



$\mathbf{x} \sim p_x(\mathbf{x})$   
 $p_x(\mathbf{x}) - ?$

$$\begin{array}{c} \mathbf{x} = f(\mathbf{z}) \\ \longleftrightarrow \\ \mathbf{z} = f^{-1}(\mathbf{x}) \end{array}$$



$\mathbf{z} \sim p_z(\mathbf{z})$   
 $p_z(\mathbf{z}) = \mathcal{N}(0, I)$

# Замена переменных (теорема)

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

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

Отношение объема  
 $\partial z$  к новому объему  
 $\partial x$

где матрица первых производных определяется так:

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

# Пример 1

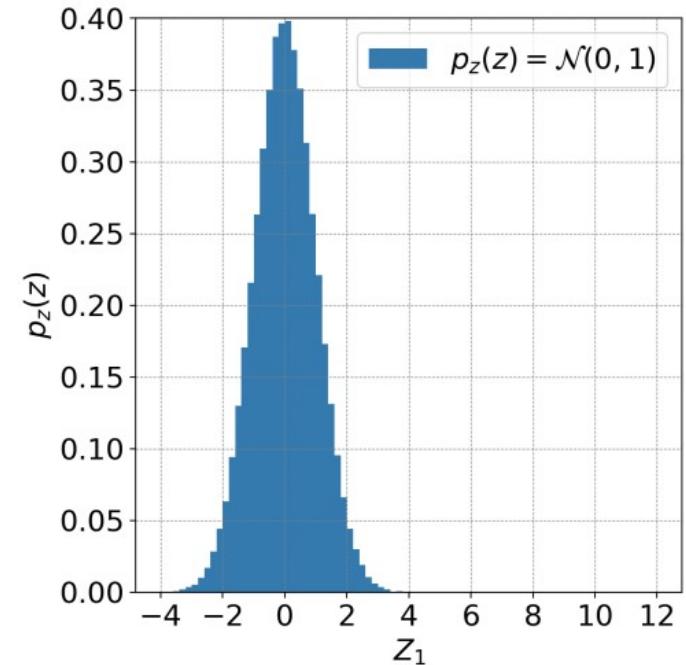


$$z = f(x) = 0.5x - 2.5$$



$$x_i \sim p_x(x)$$

$$p_x(x) - ?$$



$$z_i \sim p_z(z)$$

$$p_z(z) = \mathcal{N}(0, 1)$$

# Пример 1

Итак, дана функция  $\mathbf{f}(x)$ :

$$z = \mathbf{f}(x) = 0.5x - 2.5$$

Тогда, матрица первых производных:

$$\frac{\partial \mathbf{f}(x_i)}{\partial x_i} = (0.5)$$

И значение Якобиана:

$$\left| \det \frac{\partial \mathbf{f}(x_i)}{\partial x_i} \right| = 0.5$$

# Пример 1

Формула замены переменных:

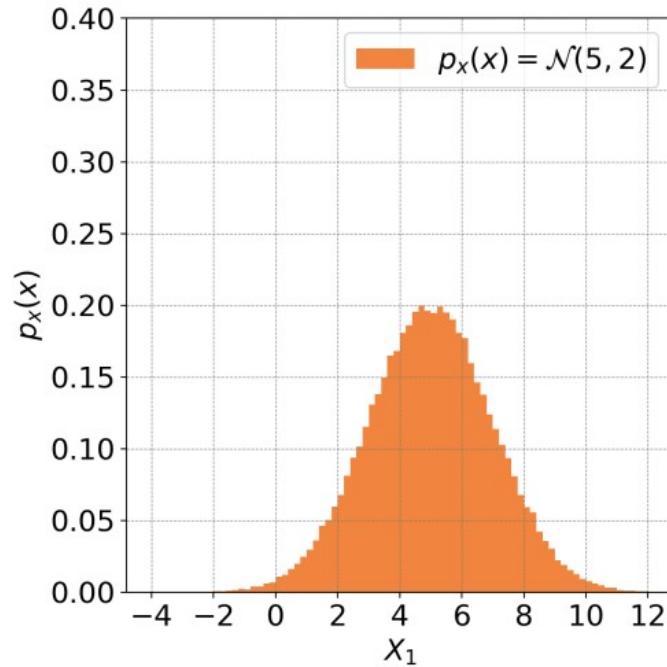
$$p_x(x_i) = \mathbf{p_z}(\mathbf{f}(x_i)) \left| \det \frac{\partial \mathbf{f}(x_i)}{\partial x_i} \right|,$$

Подставим известные выражения:

$$\mathbf{p_z}(z_i) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(z_i)^2}{2}}$$

$$\begin{aligned} p_x(x_i) &= \frac{1}{\sqrt{2\pi}} e^{-\frac{(0.5x_i - 2.5)^2}{2}} * 0.5 \\ &= \frac{1}{2\sqrt{2\pi}} e^{-\frac{(x_i - 5)^2}{2*2^2}} = \mathcal{N}(5, 2) \end{aligned}$$

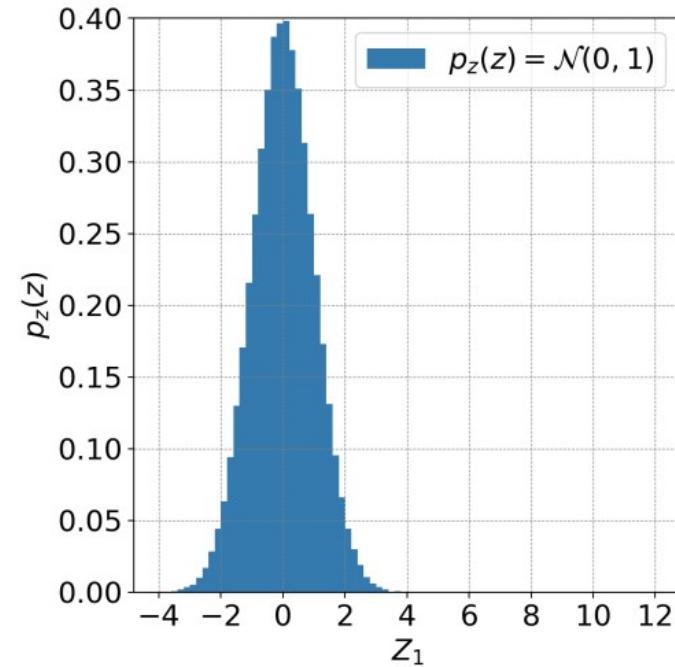
# Пример 1



$$x_i \sim p_x(x)$$

$$p_x(x) - ?$$

$$z = f(x) = 0.5x - 2.5$$



$$z_i \sim p_z(z)$$

$$p_z(z) = \mathcal{N}(0, 1)$$

# Замена переменных: теорема

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

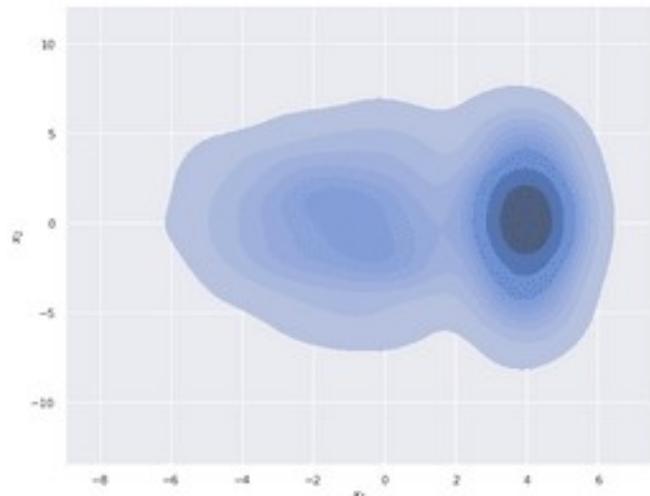
$$p_x(x_i) = p_z(\mathbf{f}(x_i)) \left| \det \frac{\partial \mathbf{f}(x_i)}{\partial x_i} \right|,$$

Отношение объема  
 $\partial z$  к новому  
объему  $\partial x$

Обратная замена переменных:

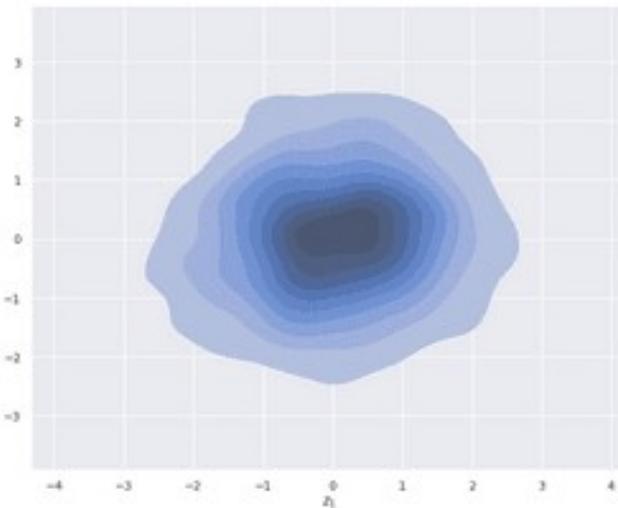
$$p_z(z_i) = p_x(\mathbf{f}^{-1}(z_i)) \left| \det \frac{\partial \mathbf{f}^{-1}(z_i)}{\partial z_i} \right|$$

# Номализующие потоки



$\mathbf{x} \sim p_x(\mathbf{x})$   
 $p_x(\mathbf{x}) - ?$

$$\begin{array}{c} \mathbf{x} = f(\mathbf{z}) \\ \longleftrightarrow \\ \mathbf{z} = f^{-1}(\mathbf{x}) \end{array}$$



$\mathbf{z} \sim p_z(\mathbf{z})$   
 $p_z(\mathbf{z}) = \mathcal{N}(0, I)$

# Постановка задачи

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

- ▶ Как будем находить  $z_i = \mathbf{f}(x_i)$ ?
  - ▶ **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:

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

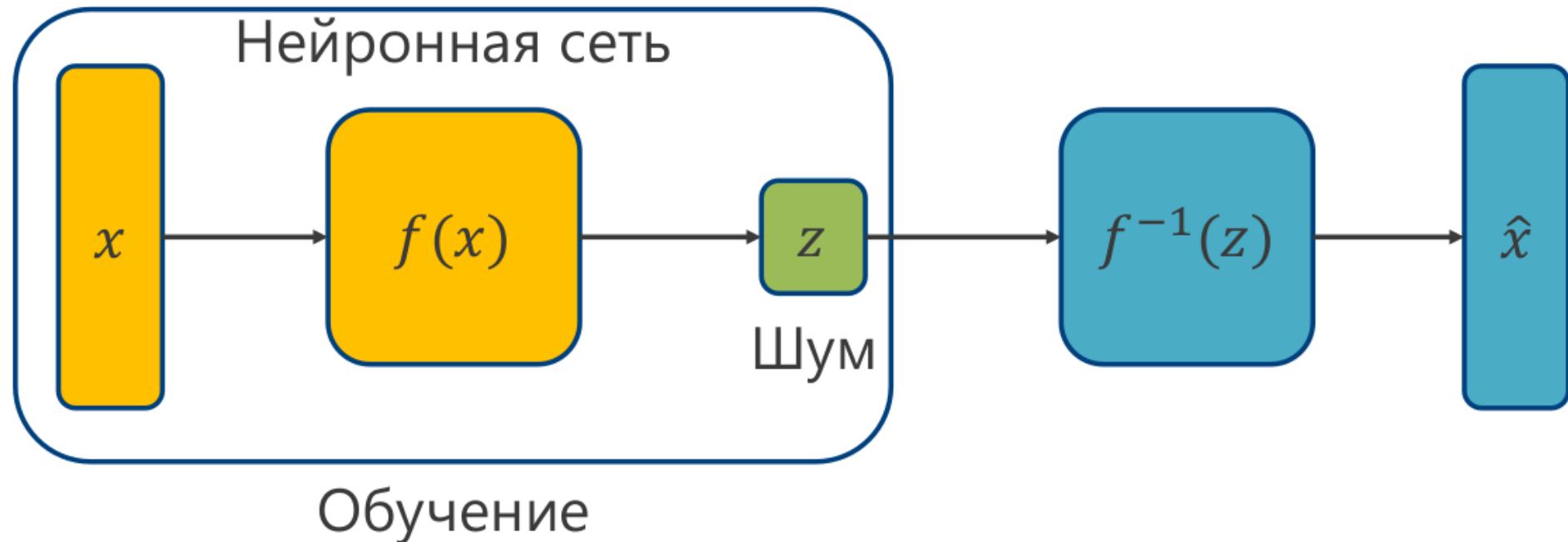
Замена переменных:

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

Подставим в функцию потерь:

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

# Алгоритм обучения



# Алгоритм обучения

Цикл обучения:

- ▶ Берем  $m$  реальных объектов  $\{x_1, x_2, \dots, x_m\}$
- ▶ Считаем функцию потерь:

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

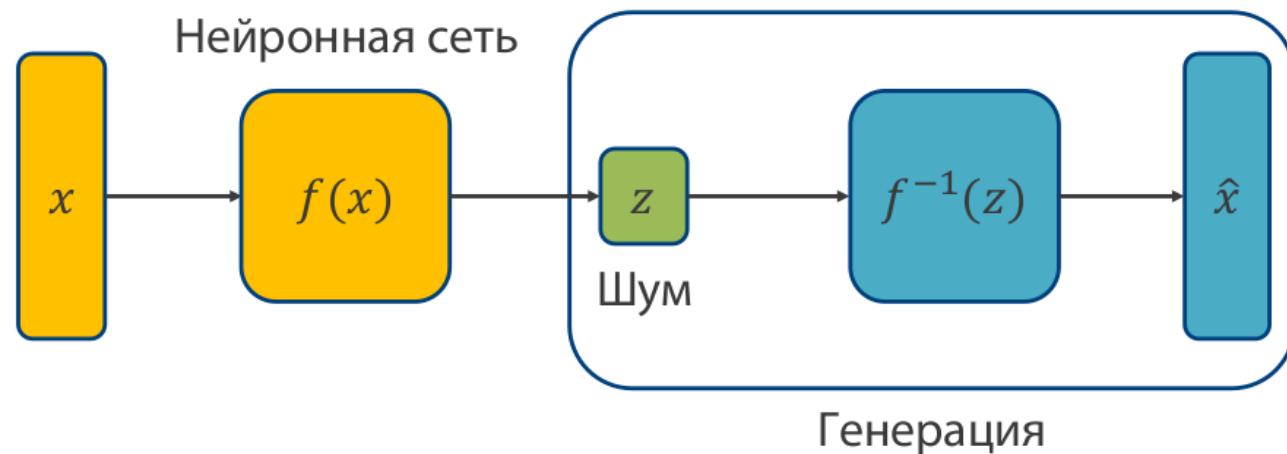
- ▶ Обновляем параметры  $\theta_f$  функции  $z_i = f(x_i)$ :

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

# Алгоритм генерации

- ▶ Генерируем случайный шум  $\{z_1, z_2, \dots, z_m\}$
- ▶ Генерируем новые объекты  $\{x_1, x_2, \dots, x_m\}$  по формуле:

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



# ФУНКЦИИ

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

где:

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

# Якобиан

- ▶ Матрица первых производных:

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

- ▶ Значение Якобиана:

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

# Обратная функция

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

# Пример Glow

