

Лекция 5

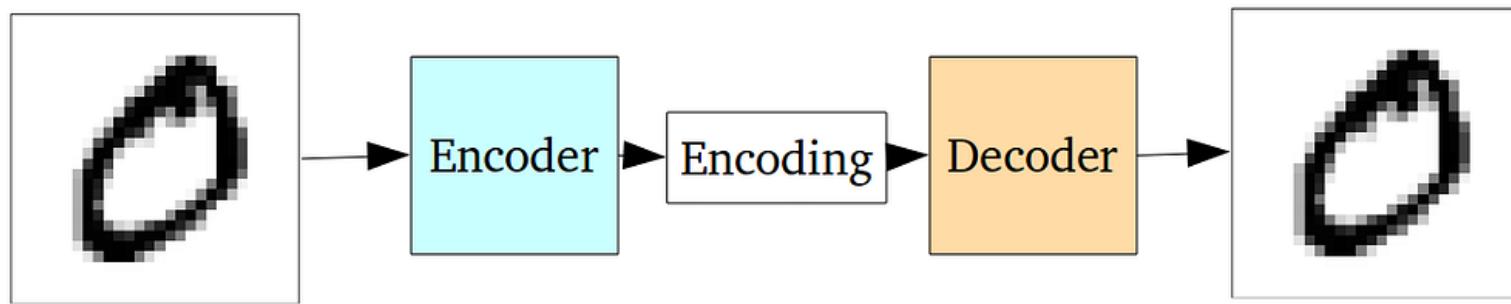
Генеративные сети

Разработка нейросетевых систем

Канев Антон Игоревич

Автоэнкодер

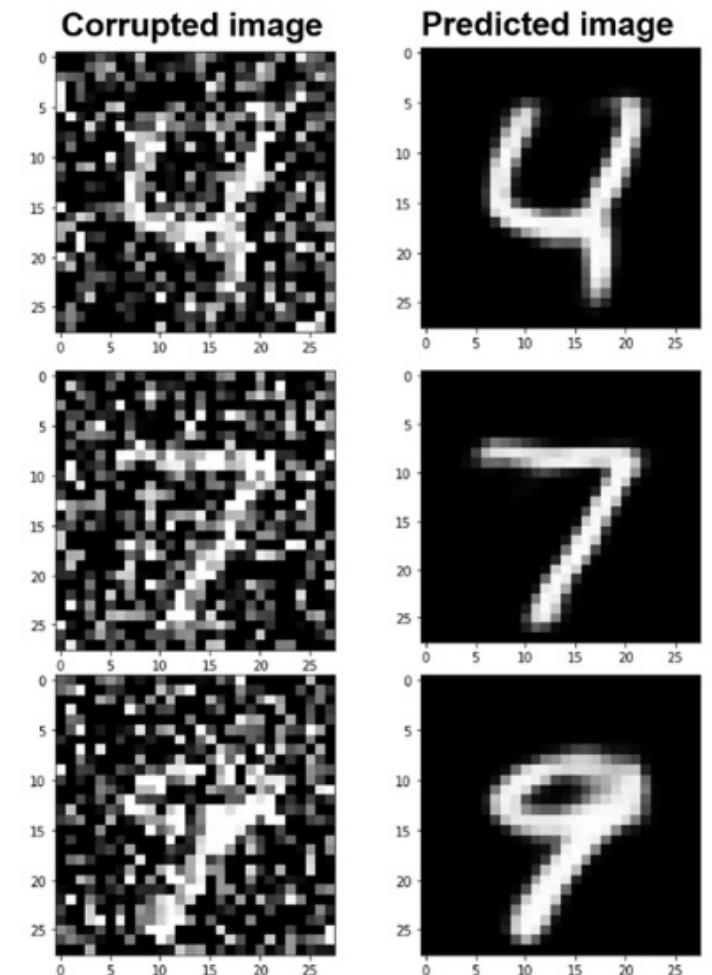
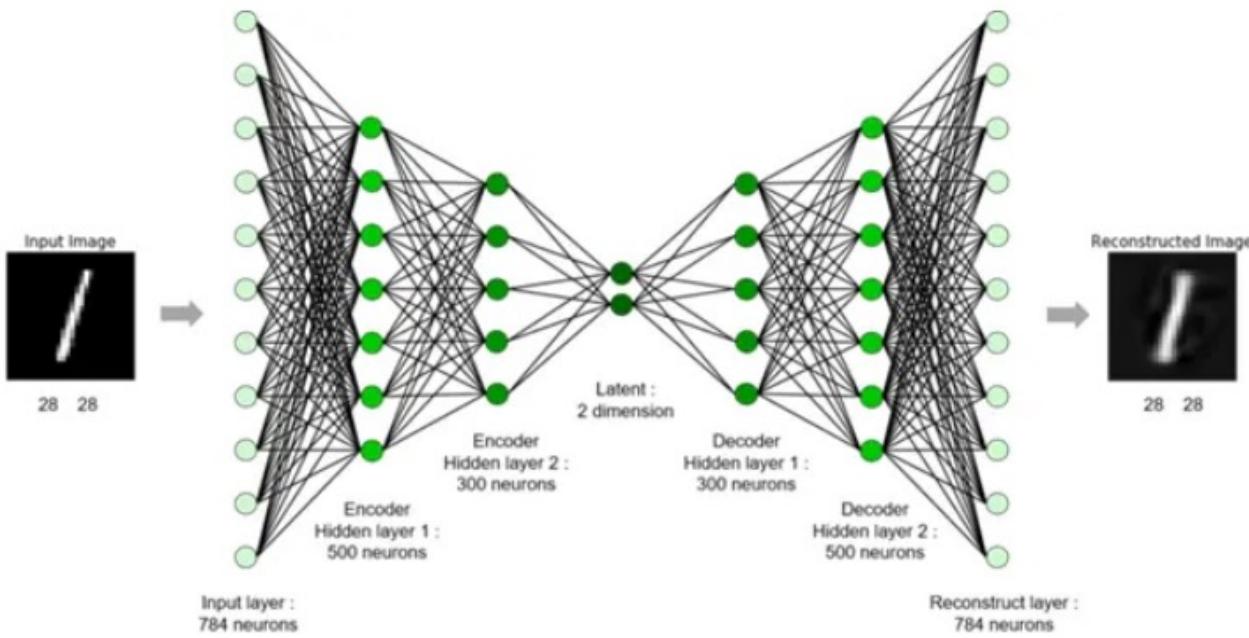
- Автоэнкодеры появились в начале 90-ых годов. Это нелинейное обобщение метода главных компонент.
- Энкодер генерирует закодированные данные таким образом, чтобы они были пригодны для восстановления входных данных



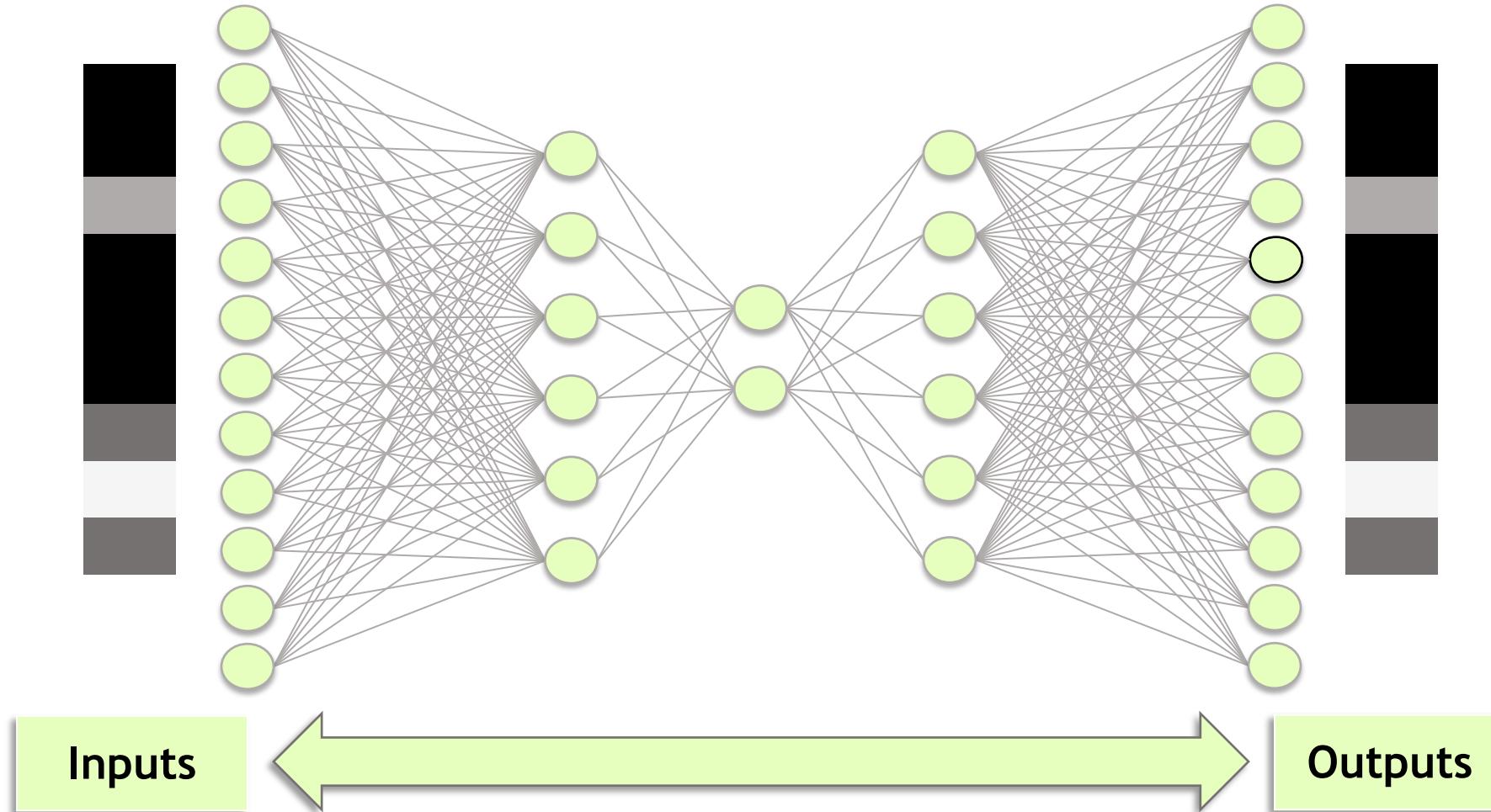
- Функция потерь выбирается как среднеквадратичная ошибка или как кросс-энтропия между входными и выходными данными
- Она не позволяет нейросети создавать выходные данные, сильно отличающиеся от входных

Denosing Autoencoder

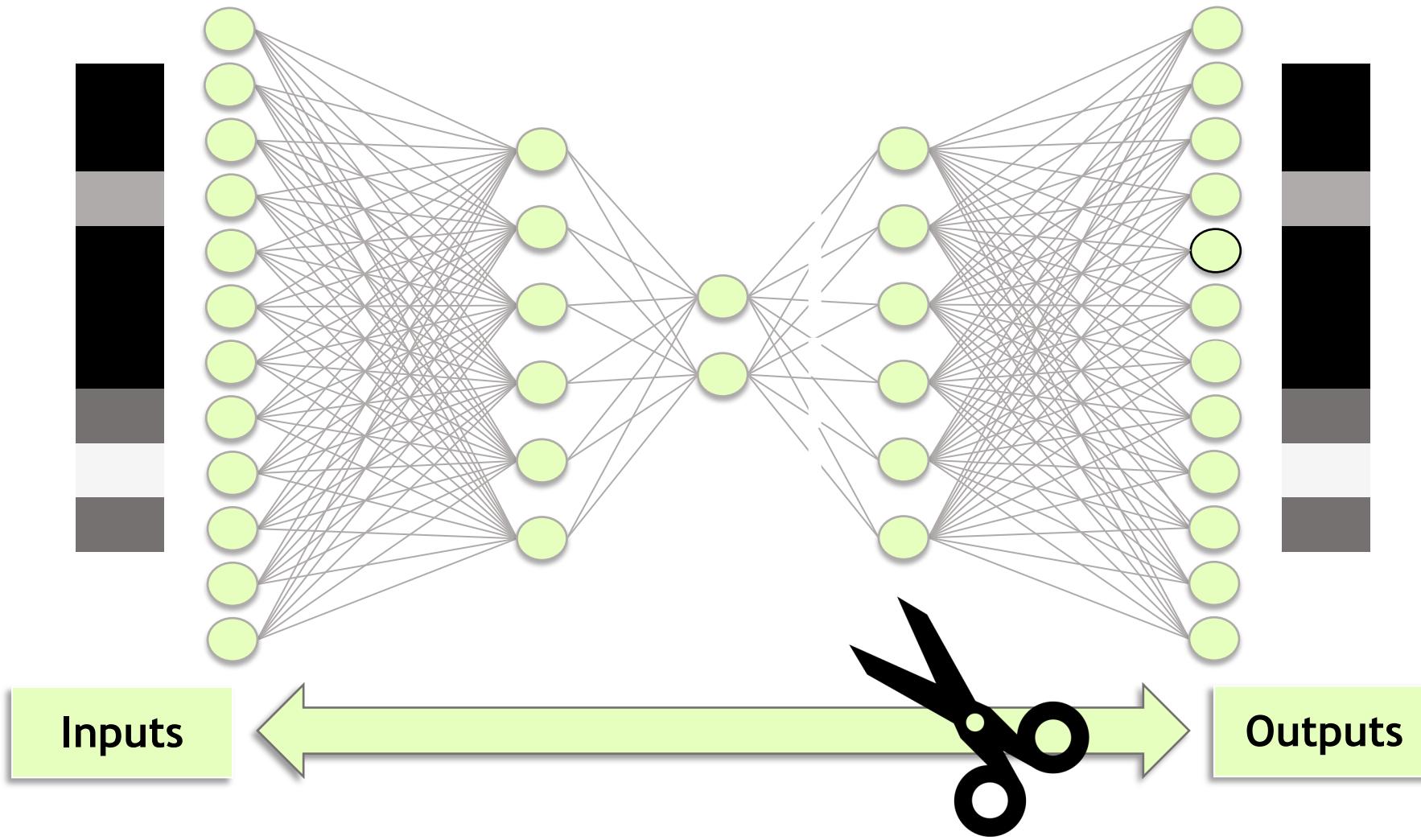
- Обучаем модель на восстановлении изображения
- Используем для удаления шума из изображения



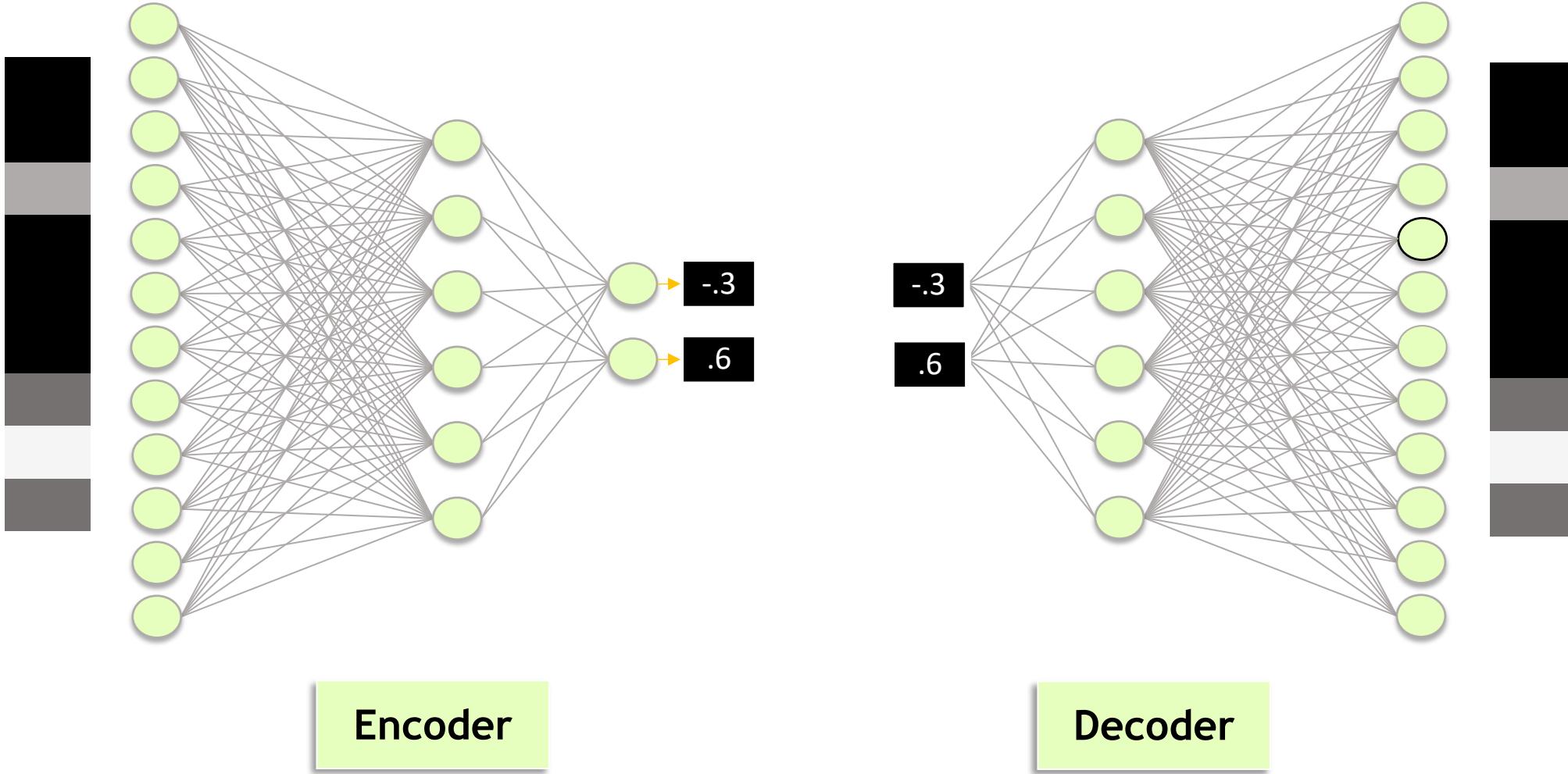
Autoencoders



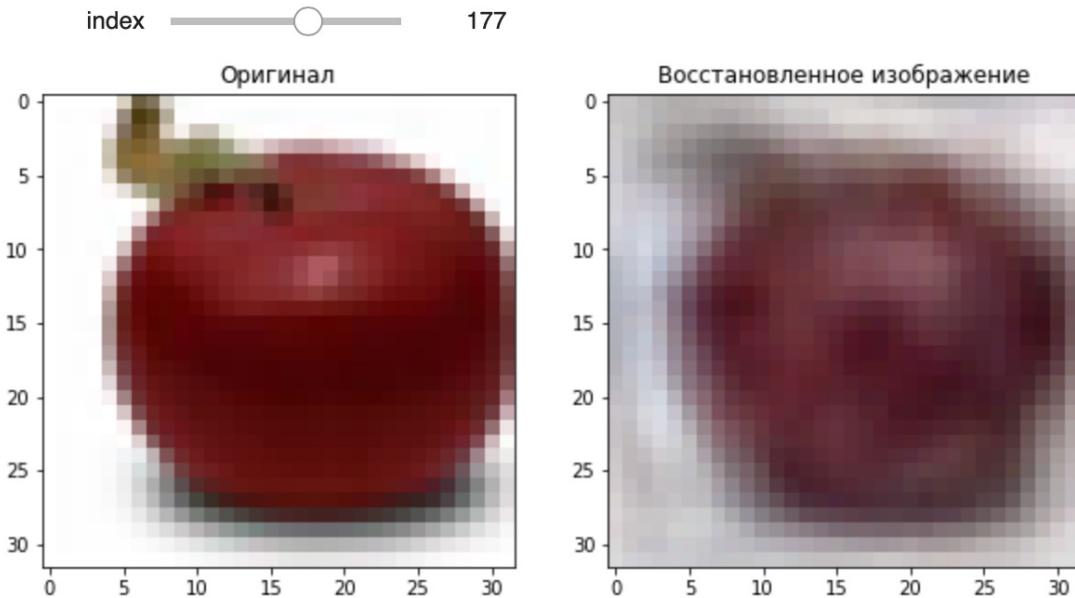
Обучение двух моделей



Latent space

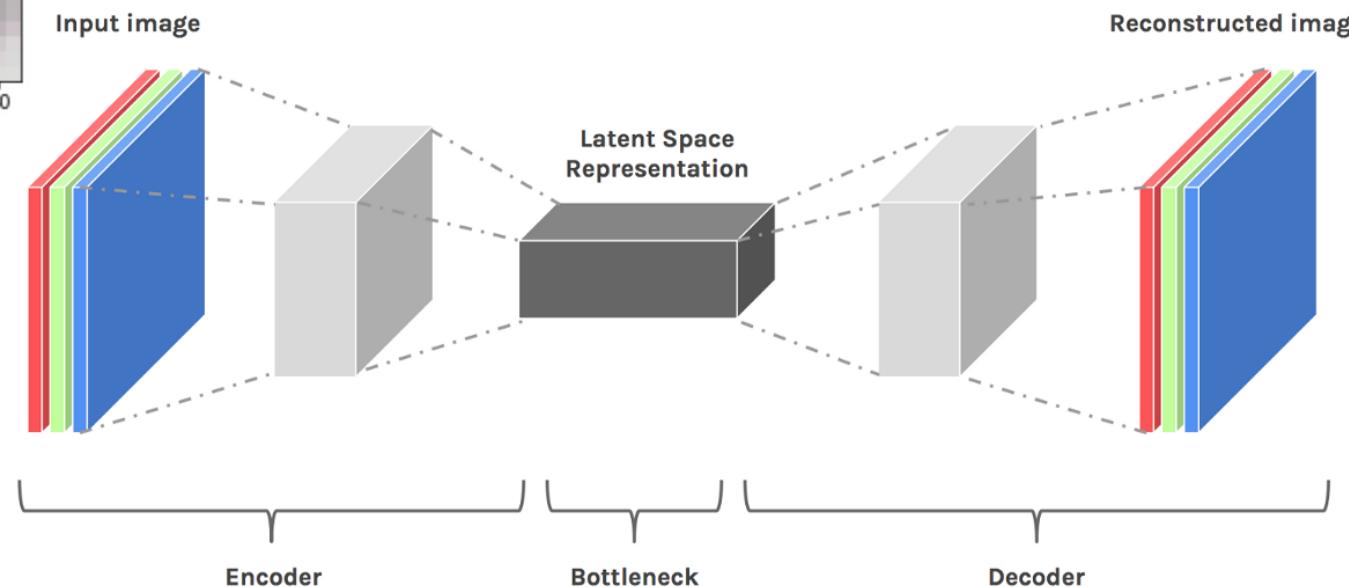


Автоэнкодер для Cifar



- Воспользуемся Cifar100 чтобы обучить автоэнкодер
- Здесь мы не используем метку label – у нас фактически неразмеченный набор данных

```
Cifar100_AE(  
    (norm): Normalize()  
    (encoder): Sequential(  
        (0): Linear(in_features=3072, out_features=512, bias=True)  
        (1): ELU(alpha=1.0)  
        (2): Linear(in_features=512, out_features=256, bias=True)  
        (3): ELU(alpha=1.0)  
        (4): Linear(in_features=256, out_features=64, bias=True)  
        (5): Tanh()  
    )  
    (decoder): Sequential(  
        (0): Linear(in_features=64, out_features=256, bias=True)  
        (1): ELU(alpha=1.0)  
        (2): Linear(in_features=256, out_features=512, bias=True)  
        (3): ELU(alpha=1.0)  
        (4): Linear(in_features=512, out_features=3072, bias=True)  
    )  
)
```



Автоэнкодер для Cifar

```
class Cifar100_AE(nn.Module):
    def __init__(self, hidden_size=32, classes=100):
        super(Cifar100_AE, self).__init__()
        # https://blog.jovian.ai/image-classification-of-cifar100-dataset-us
        self.norm = Normalize([0.5074, 0.4867, 0.4411], [0.2011, 0.1987, 0.2025])
        self.encoder = nn.Sequential(
            nn.Linear(32*32*3, hidden_size),
            nn.ELU(),
            nn.Linear(hidden_size, hidden_size//2),
            nn.ELU(),
            nn.Linear(hidden_size//2, hidden_size//8),
            nn.Tanh()
        )
        self.decoder = nn.Sequential(
            nn.Linear(hidden_size//8, hidden_size//2),
            nn.ELU(),
            nn.Linear(hidden_size//2, hidden_size),
            nn.ELU(),
            nn.Linear(hidden_size, 32*32*3),
        )
```

- Автоэнкодер состоит из 2-ух частей, все слои сгруппированы в эти части
- Это позволяет при предсказании получить больше ответов: выход энкодера и декодера

```
def forward(self, input):
    normed = self.norm(input)
    encoded = self.encoder(normed)
    out = self.decoder(encoded)
    return out, encoded, normed
```

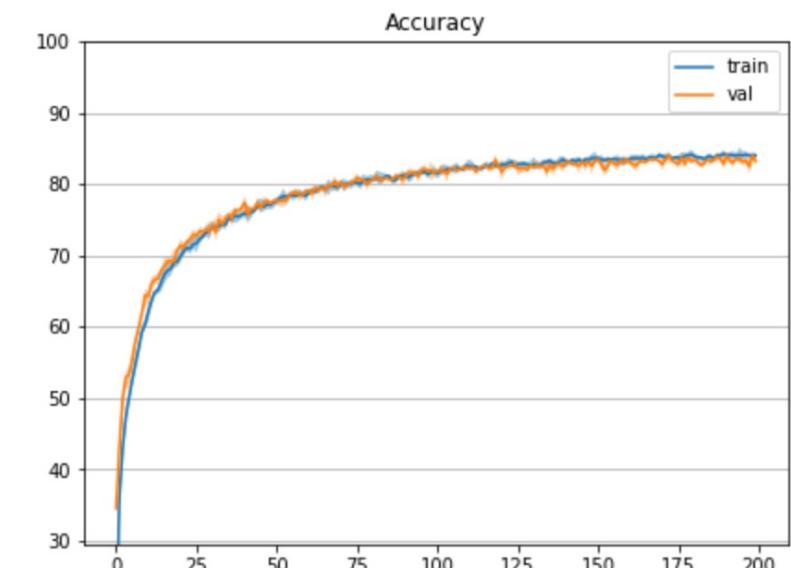
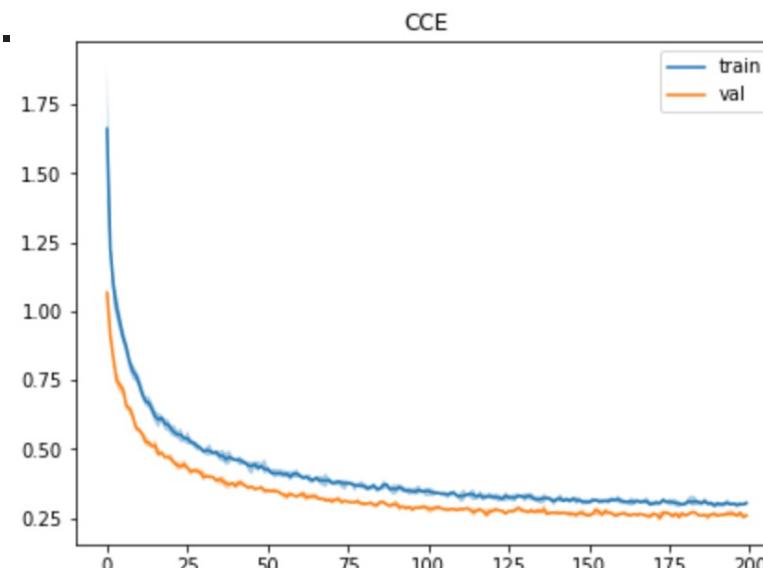
Коэффициент детерминации

- Accuracy – здесь мы считаем не процент правильных ответов, а коэффициент детерминации

$$R^2 = 1 - \frac{D[y|x]}{D[y]} = 1 - \frac{\sigma^2}{\sigma_y^2}$$

- Коэффициент детерминации для модели с константой принимает значения от 0 до 1. Чем ближе значение коэффициента к 1, тем сильнее зависимость.

- Функция потерь – MSE



Эмбеддинги

- Embedding – векторное представление
- Используем выход энкодера в качестве координат для визуализации
- Сделаем проекцию на плоскость с помощью метода главных компонент РСА

```
# прямой + обратный проходы + оптимизация
out, embedding, norm = model(inputs)
embeddings.append(embedding.detach().cpu().numpy())
images.append(inputs.detach().cpu().numpy())
reconstructs.append(out.detach().cpu().numpy())
```

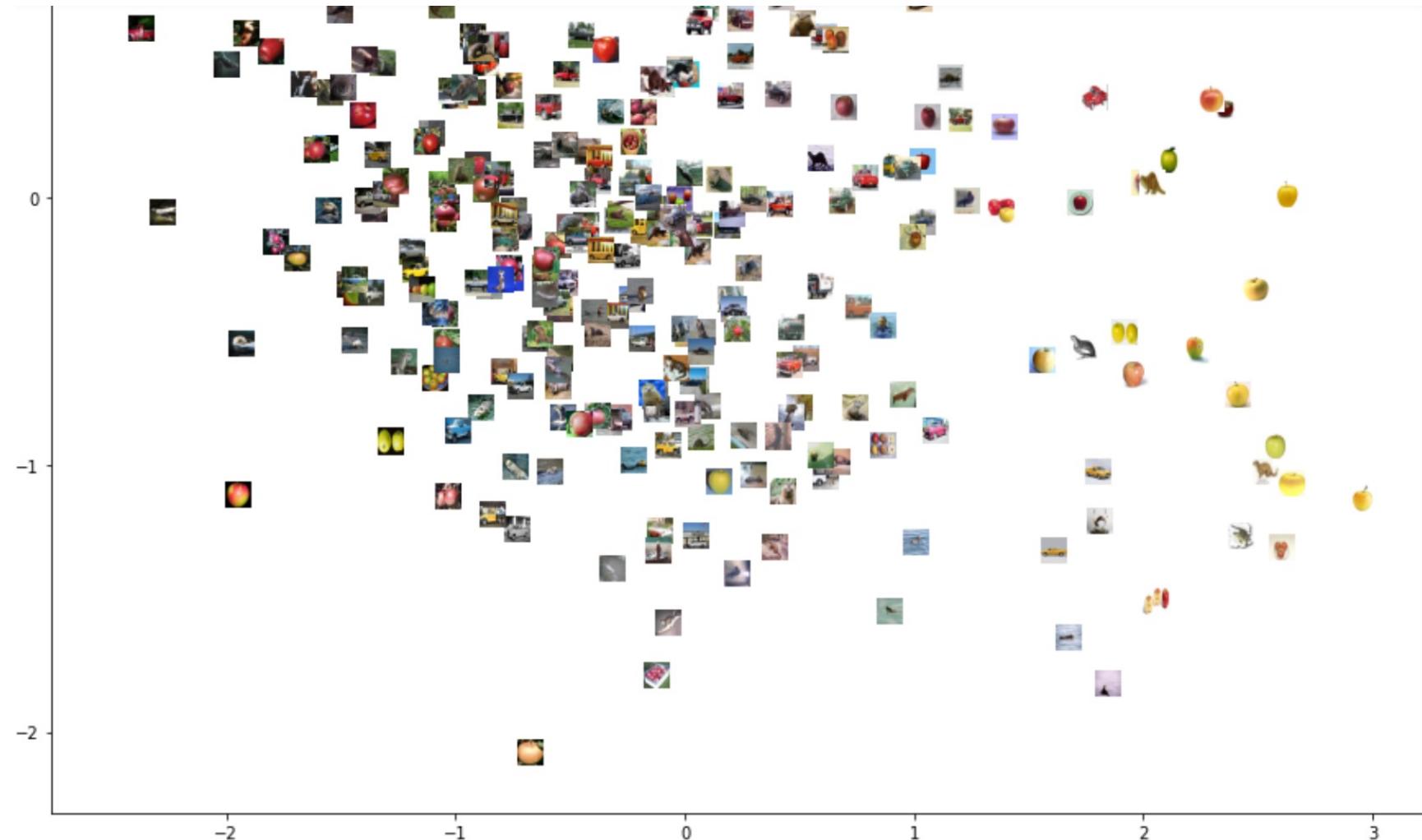
```
projections = PCA(n_components=2).fit_transform(embeddings)

def implot(x, y, image, ax, zoom=1):
    im = OffsetImage(image, zoom=zoom)
    ab = AnnotationBbox(im, (x, y), xycoords='data', frameon=False)
    ax.add_artist(ab)
    ax.update_datalim(np.column_stack([x, y]))
    ax.autoscale()

fig, ax = plt.subplots(1, 1, figsize=(15, 15))
for img, x in zip(images, projections):
    img = img.reshape(32, 32, 3)/255.
    implot(x[0], x[1], img, ax=ax, zoom=0.5)
```

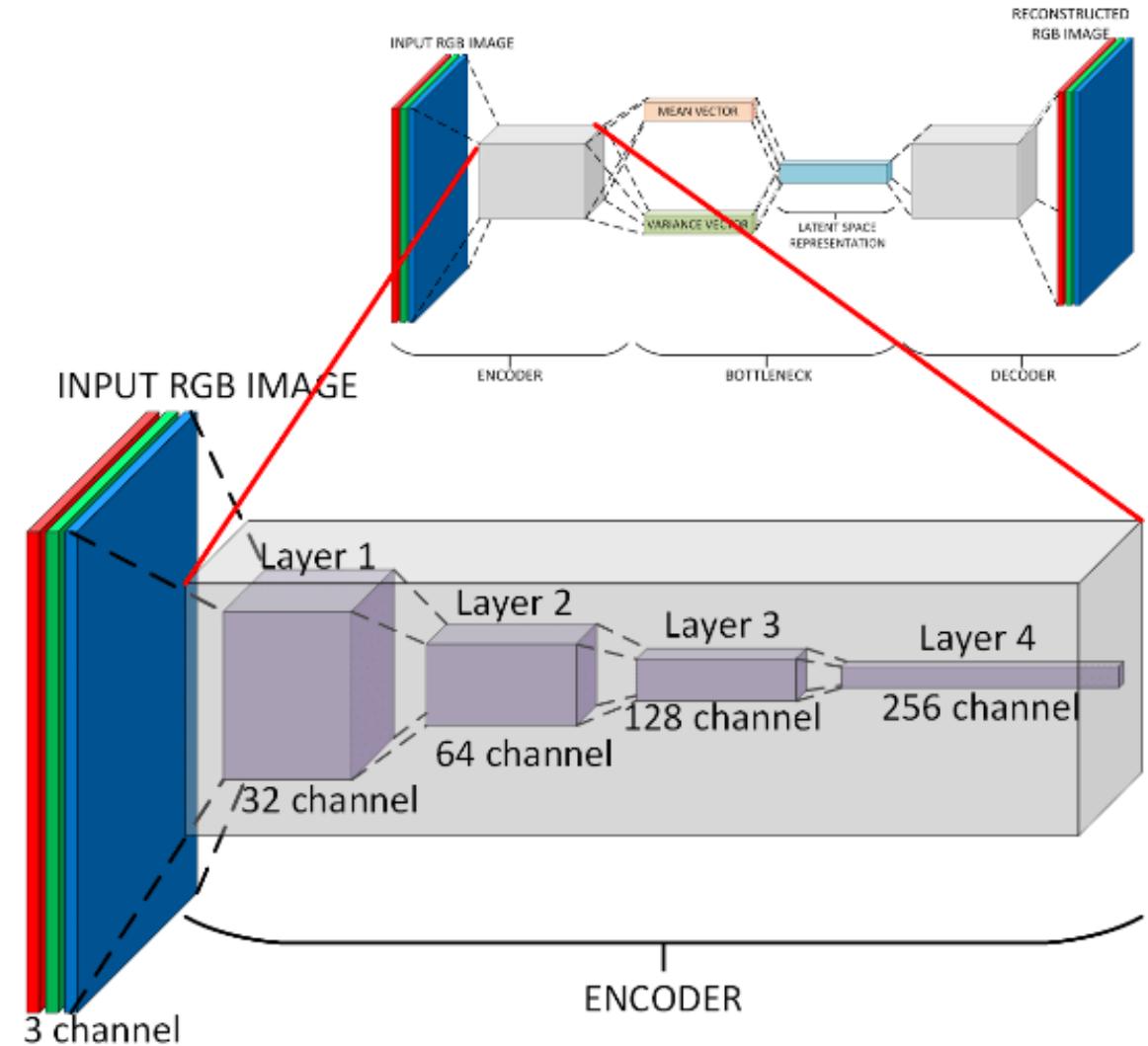
Эмбеддинги

- Отображение каждого примера находятся ближе друг к другу в зависимости от их схожести



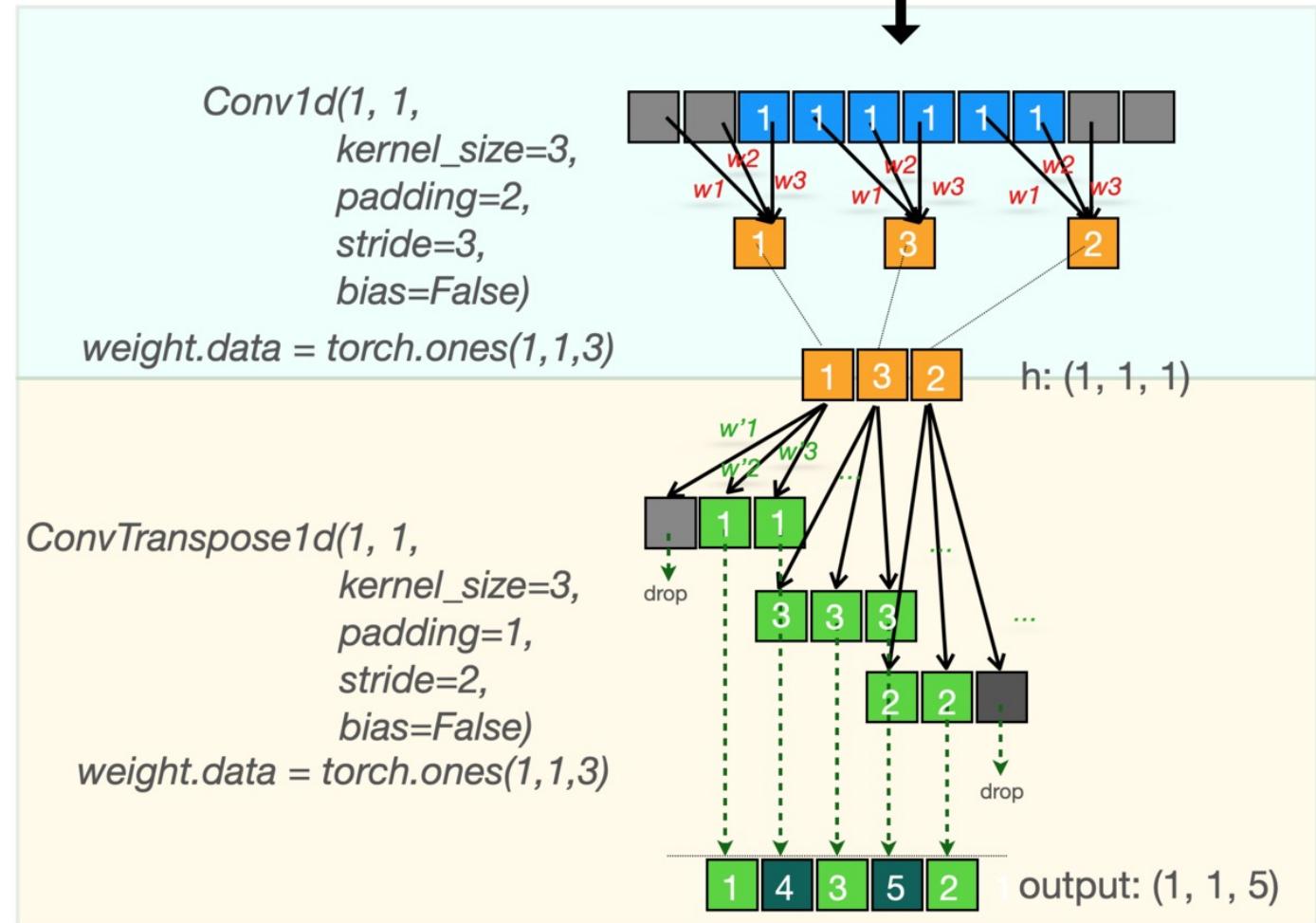
Сверточный кодировщик

- Сверточный кодировщик – это привычная сверточная нейронная сеть
- На самом деле на картинке вариационный автоэнкодер, о нем расскажем позже. Но принцип тот же



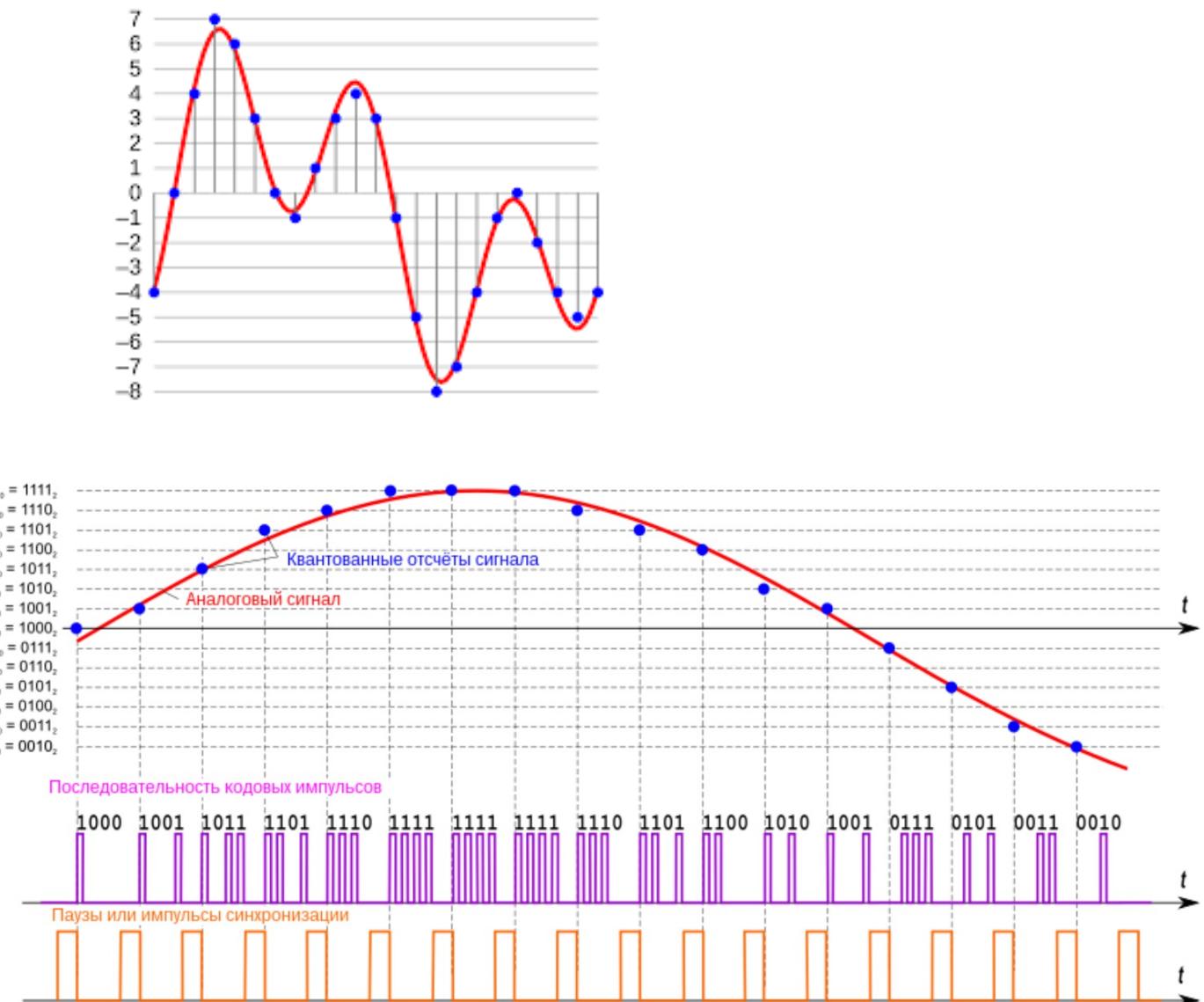
Одномерный автокодировщик

- Сверточный кодировщик – это привычная сверточная нейронная сеть
- Декодировщик – обратные сверточные слои
- Обучаем параметры: их меньше и они тоже разделяемые



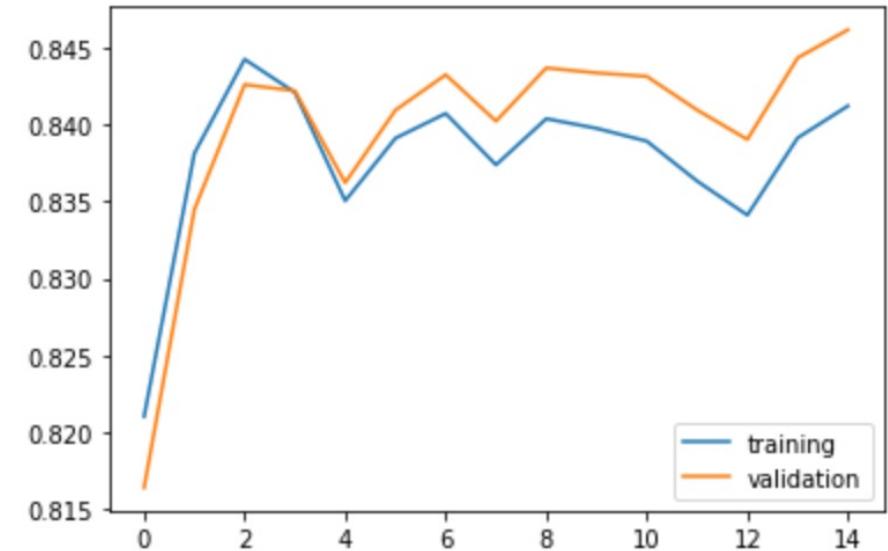
Обработка звука

- wav – формат с использованием импульсно-кодовой модуляции
- В mp3 используется алгоритм сжатия с потерями



Удаление шума из аудио

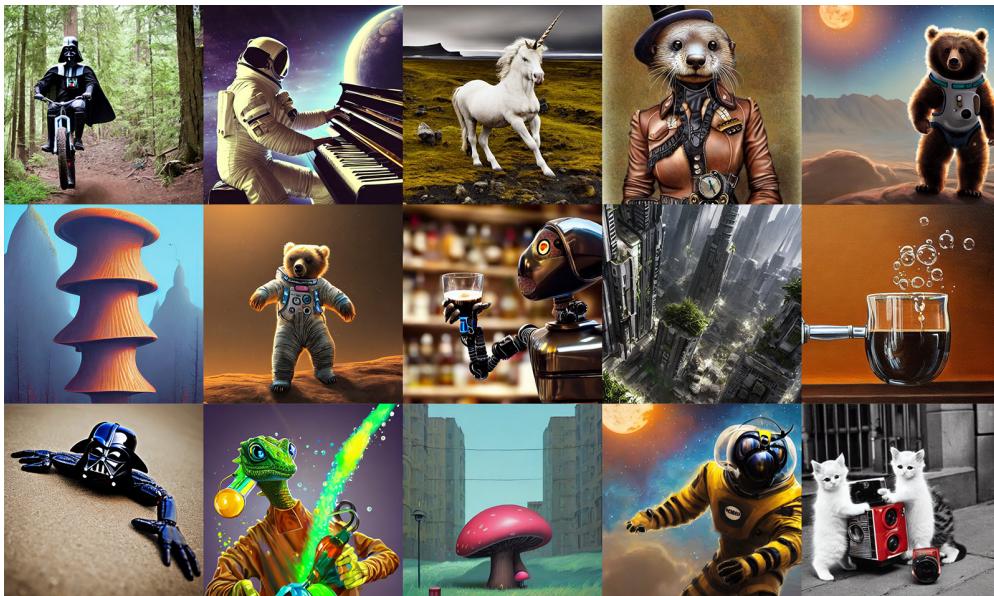
- Используем сверточный автоэнкодер
- В слоях задаем количество каналов и другие настройки сверточных слоев



```
class DenoisingAE(nn.Module):  
    def __init__(self):  
        super().__init__()  
  
        self.encoder = nn.Sequential(  
            nn.Conv1d(2, 256, kernel_size=3, stride=2, padding=1),  
            Mish(),  
            nn.Conv1d(256, 512, kernel_size=3, stride=2, padding=1),  
            Mish(),  
            nn.Conv1d(512, 1024, kernel_size=3, stride=2, padding=1),  
            Mish(),  
        )  
        self.decoder = nn.Sequential(  
            nn.ConvTranspose1d(1024, 512, kernel_size=3, stride=2, padding=1),  
            Mish(),  
            nn.ConvTranspose1d(512, 256, kernel_size=3, stride=2, padding=1),  
            Mish(),  
            nn.ConvTranspose1d(256, 2, kernel_size=3, stride=2, padding=1),  
        )
```

Современные генеративные модели

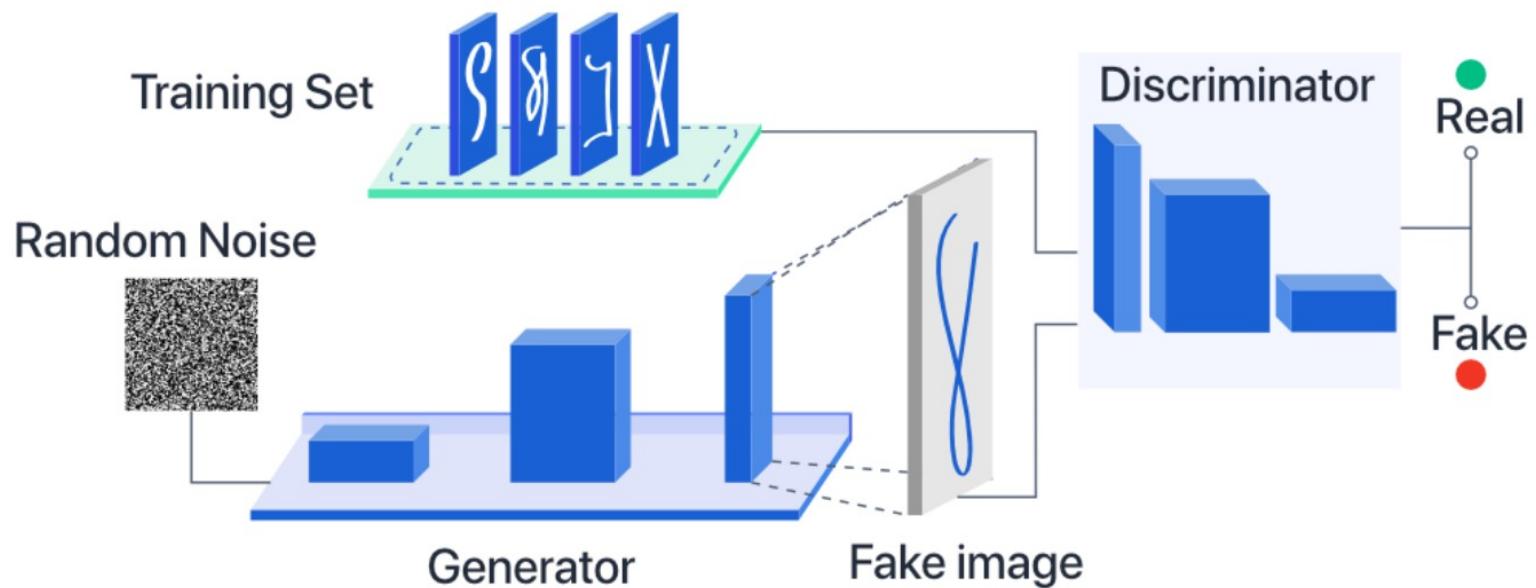
- Dall-E
- Midjourney
- Stable Diffusion



GAN

- Генеративно-состязательная сеть
(Generative adversarial network)

- Дискриминативная отличает правильные от неправильных



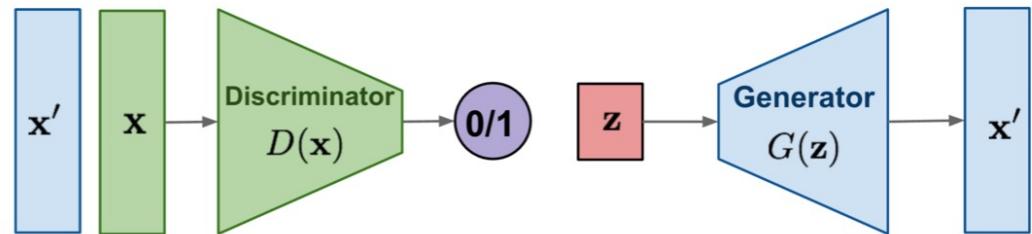
Описал Ян Гудфеллоу в 2014

- Генеративная модель создает образы

Виды генеративных сетей

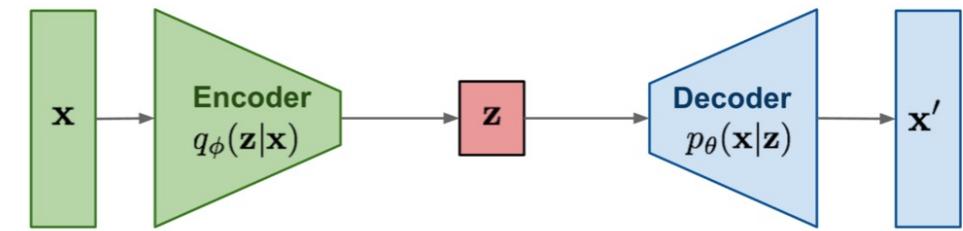
1. Вариационные автоэнкодеры (VAE)

GAN: Adversarial training



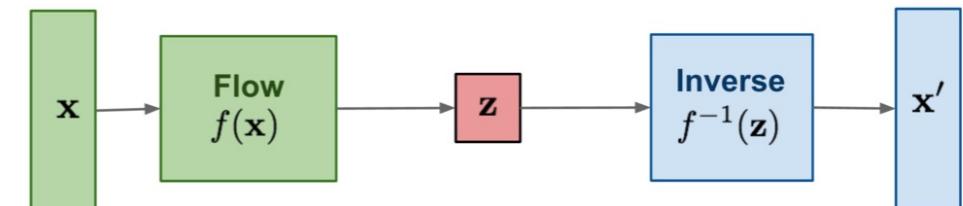
2. Генеративные состязательные сети (GAN).

VAE: maximize variational lower bound



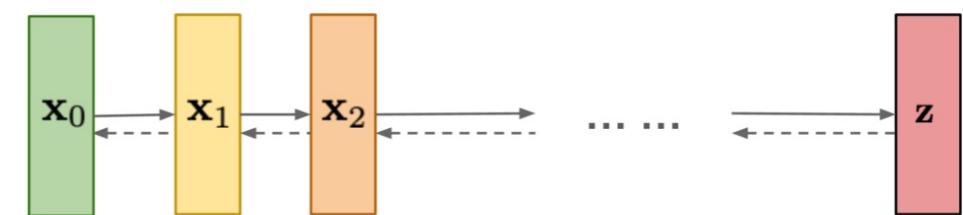
3. Flow-based models

Flow-based models:
Invertible transform of distributions



4. Diffusion (недавняя тенденция)

Diffusion models:
Gradually add Gaussian noise and then reverse

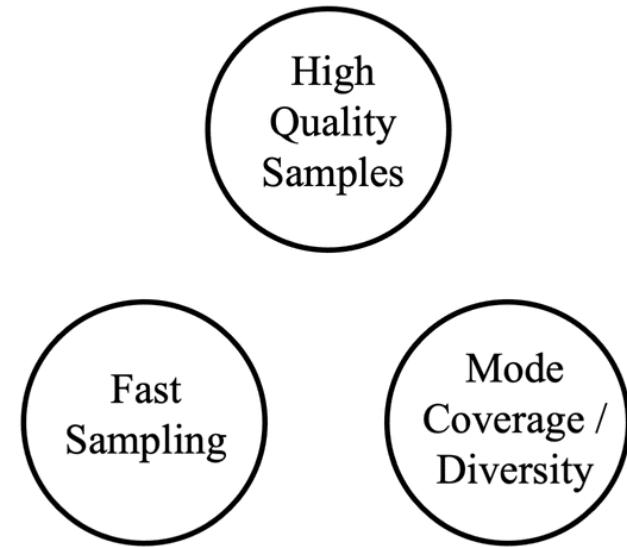


- Источник: <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

Какую модель выбрать?

VAE	Поток	GAN	Диффузия
Высокая частота дискретизации.	Высокая частота дискретизации.	Высокая частота дискретизации. Высокое качество генерации образцов.	Высокое качество генерации образцов. Генерация разнообразных образцов
Генерация разнообразных образцов	Генерация разнообразных образцов		

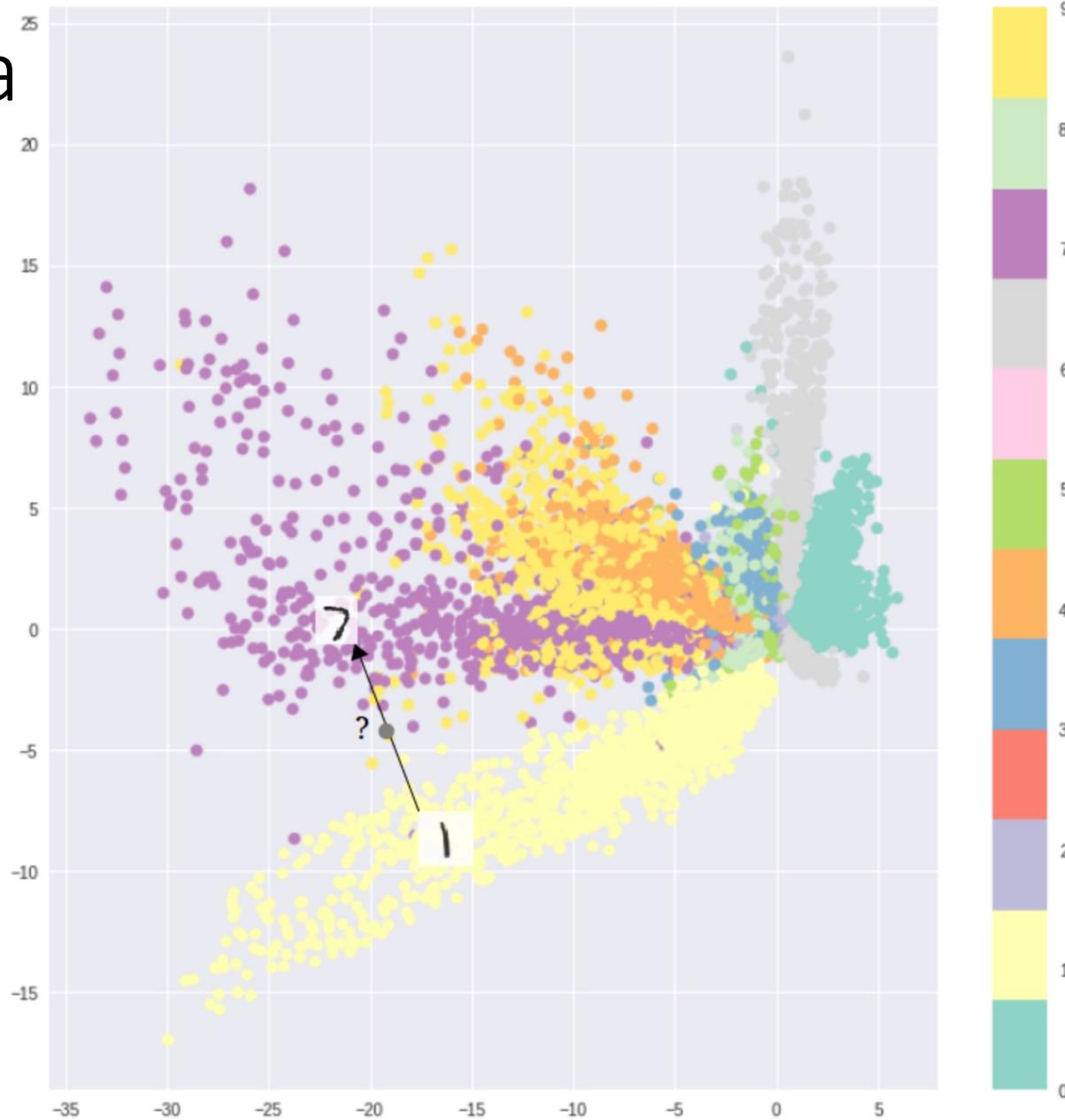
The Generative Learning Trilemma



- Решение трилеммы генеративного обучения
Источник: <https://nvlabs.github.io/denoising-diffusion-gan/>

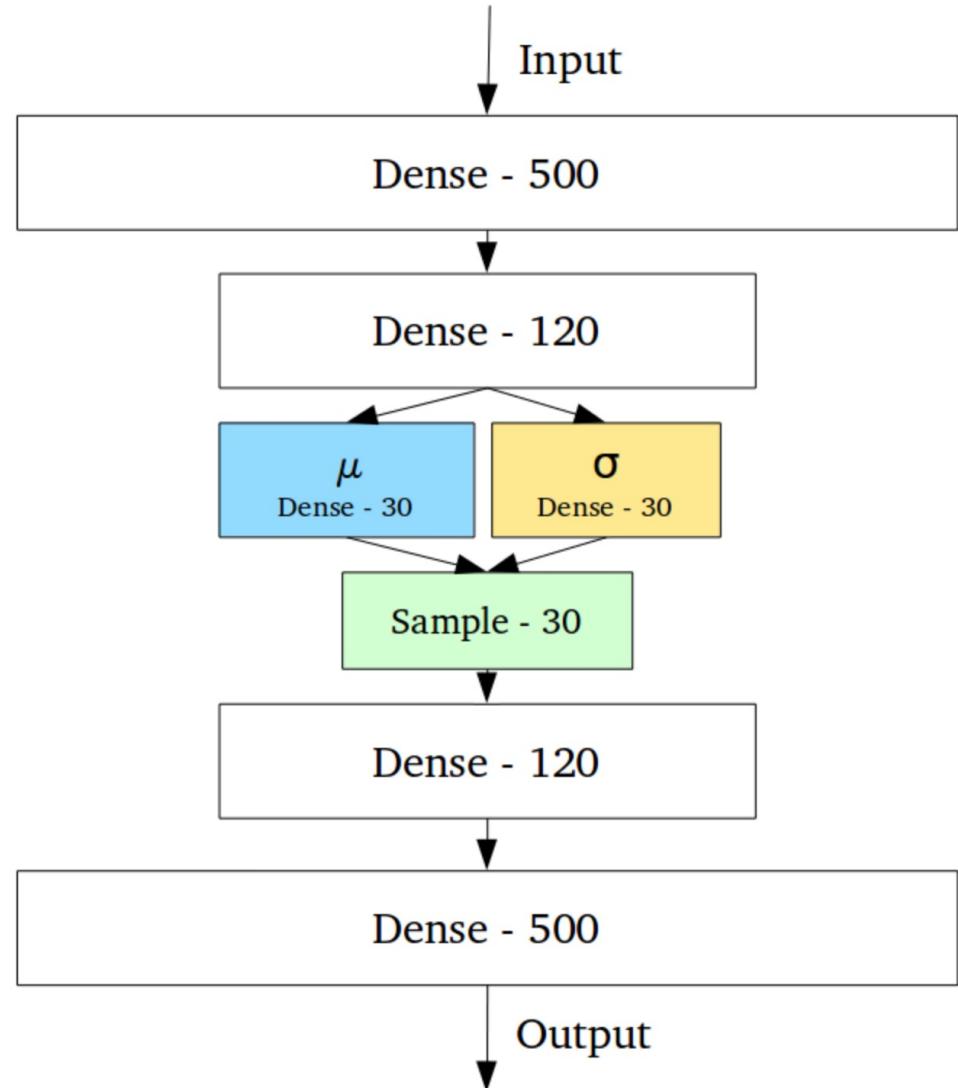
Эмбеддинг автоэнкодера

- Всё замечательно, если вам необходимо просто воспроизвести исходные изображения.
- Однако, если вы строите генеративную модель, ваша цель не простое дублирование изображений.
- Вы хотите получить случайное или видоизмененное изображение, восстановленное из непрерывного скрытого пространства.
- Если пространство имеет разрывы, то декодер выдаст нереалистичное изображение

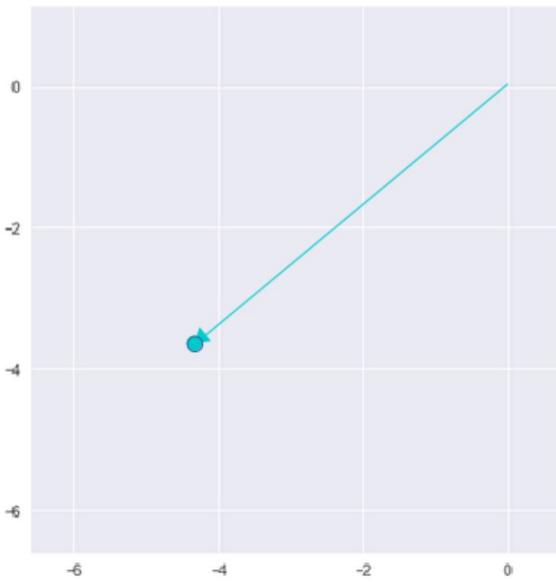


Вариационный автоэнкодер

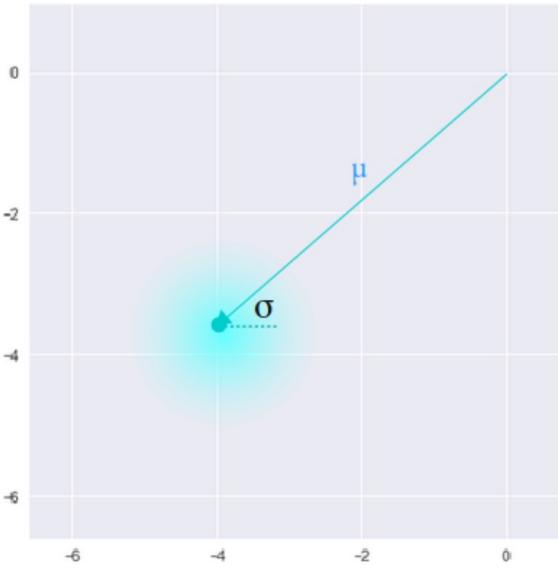
- Вариационный автоэнкодер (Variational Autoencoder – VAE) имеет отличие от стандартного автоэнкодера.
- Их скрытое пространство по построению является непрерывным, позволяя выполнять случайные преобразования и интерполяцию.
- Непрерывность скрытого пространства достигается неожиданным способом: энкодер выдаёт не один вектор размера n , а два вектора размера n – вектор средних значений μ и вектор стандартных отклонений σ .



Эмбеддинг VAE

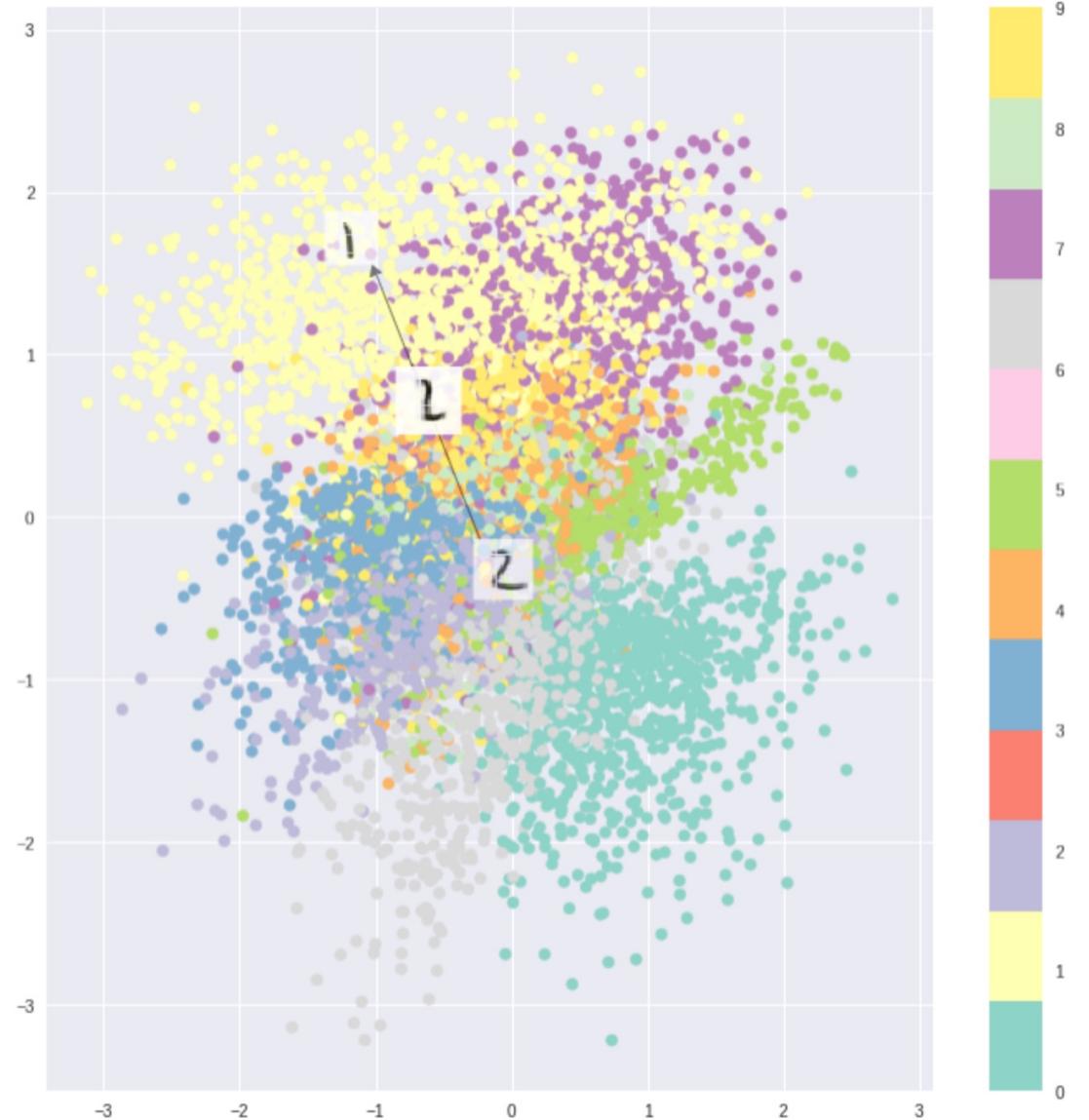


Standard Autoencoder
(direct encoding coordinates)



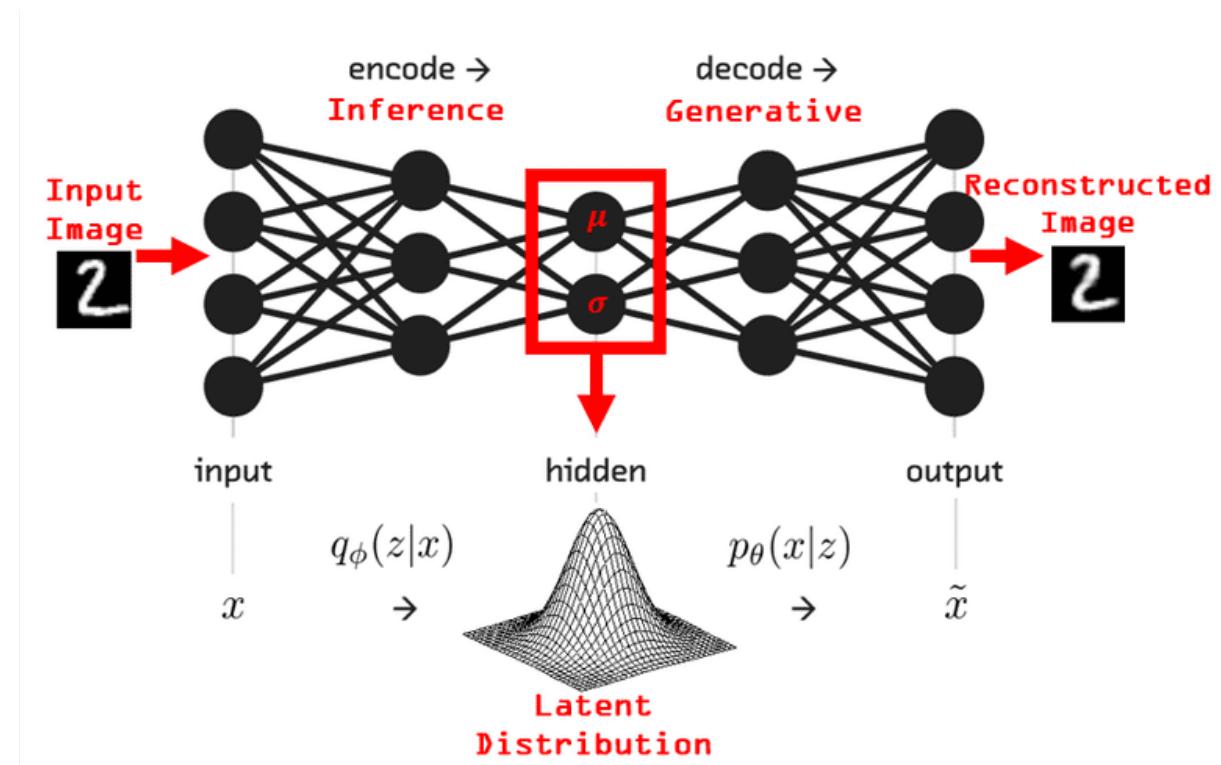
Variational Autoencoder
(μ and σ initialize a probability distribution)

- Если вы хотите восстановить входные данные, вы просто выбираете подходящее распределение и посыпаете его в декодер



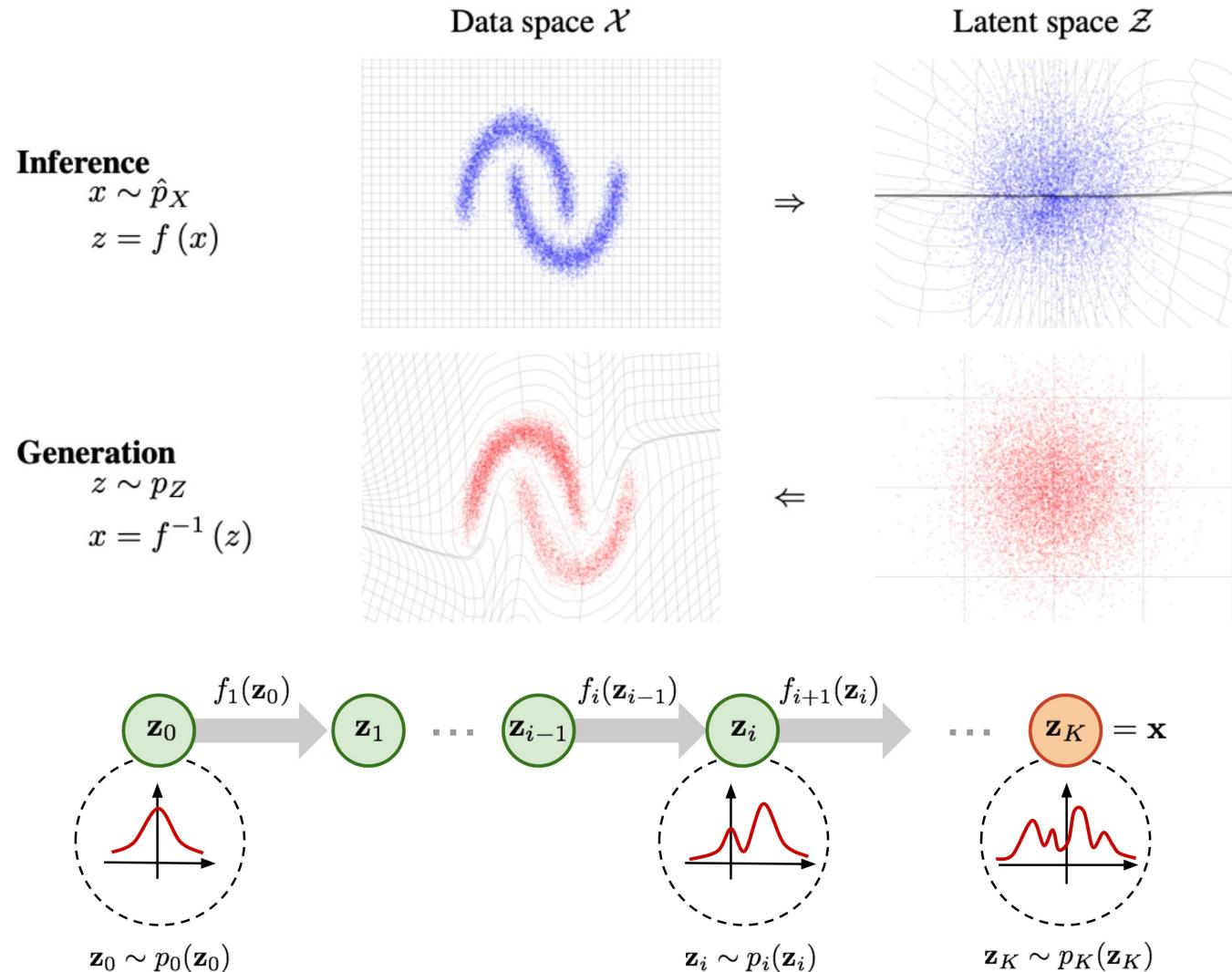
Распределение скрытого состояния

- Исходные изображения имеют свое распределение
- Мы получаем на выходе кодировщика свое распределение и хотим из него получить изображения



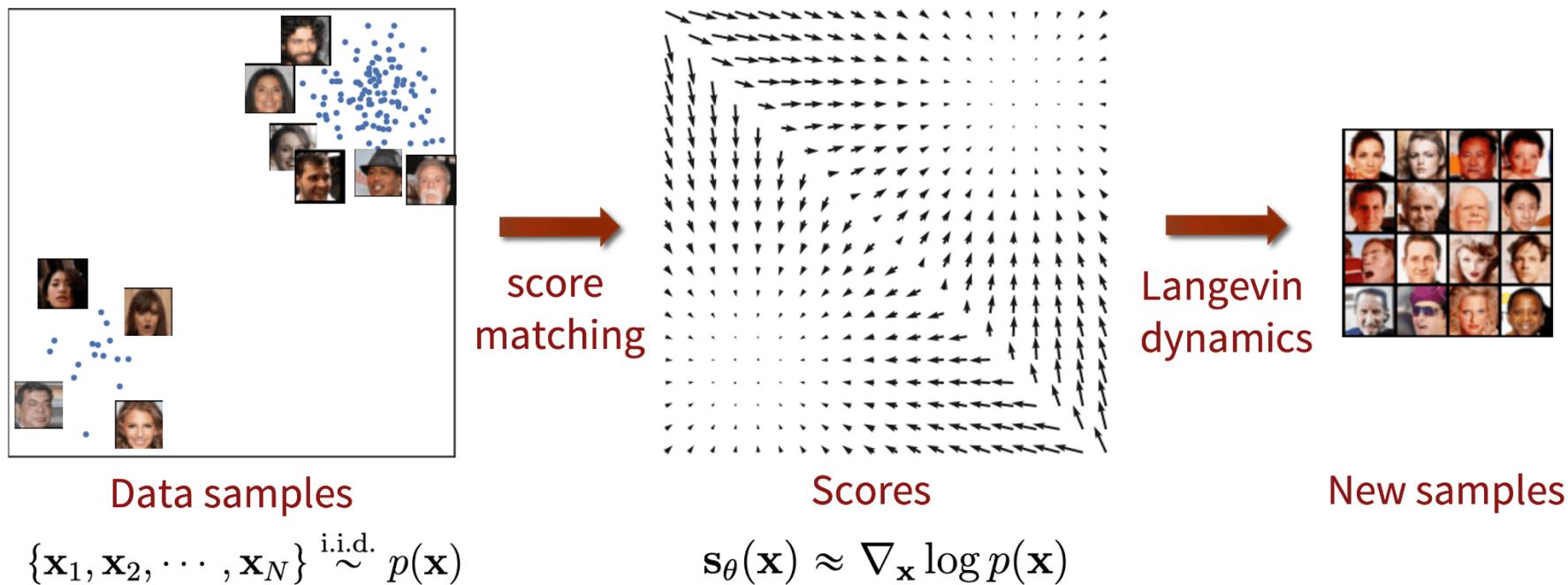
Модели основанные на потоках

- Нам требуется подобрать преобразование, которое позволит отобразить распределение наших изображений в скрытое пространство с Гауссовым распределением
- И мы можем восстановить изображение из скрытого пространства



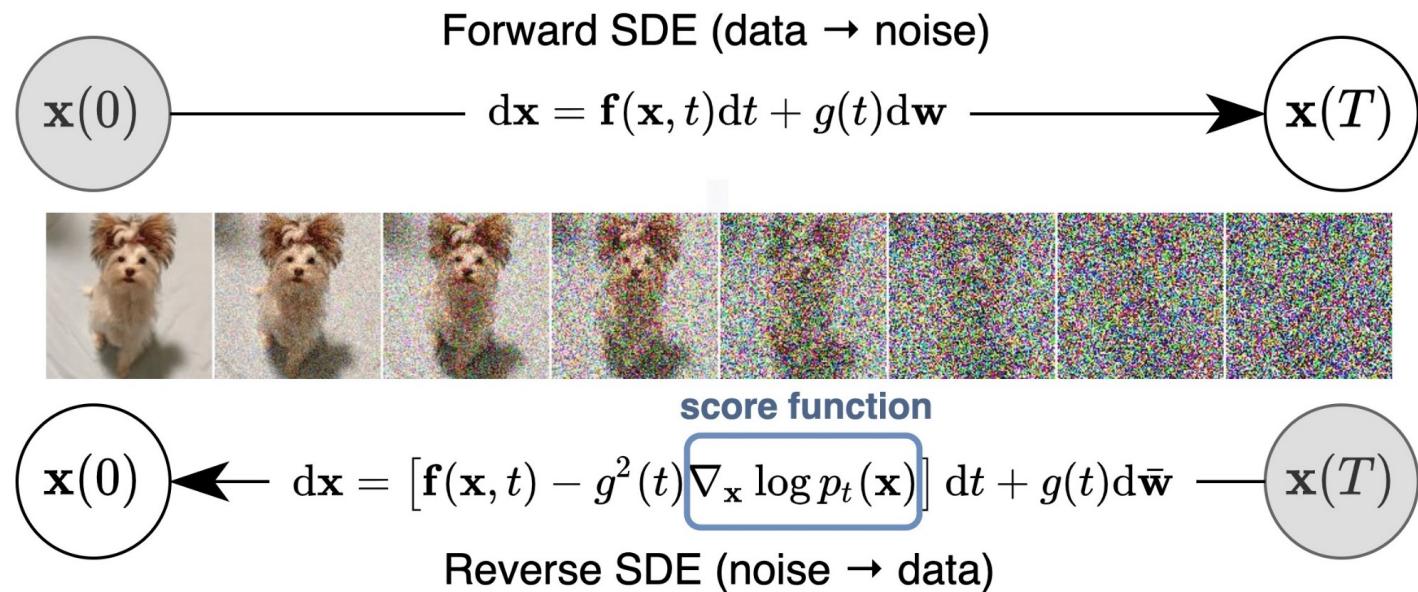
Диффузионные модели

- Появились в 2015, но были хуже GAN. Но в 2021 позволили достичь современных результатов
- Используется Score-функция для обучения



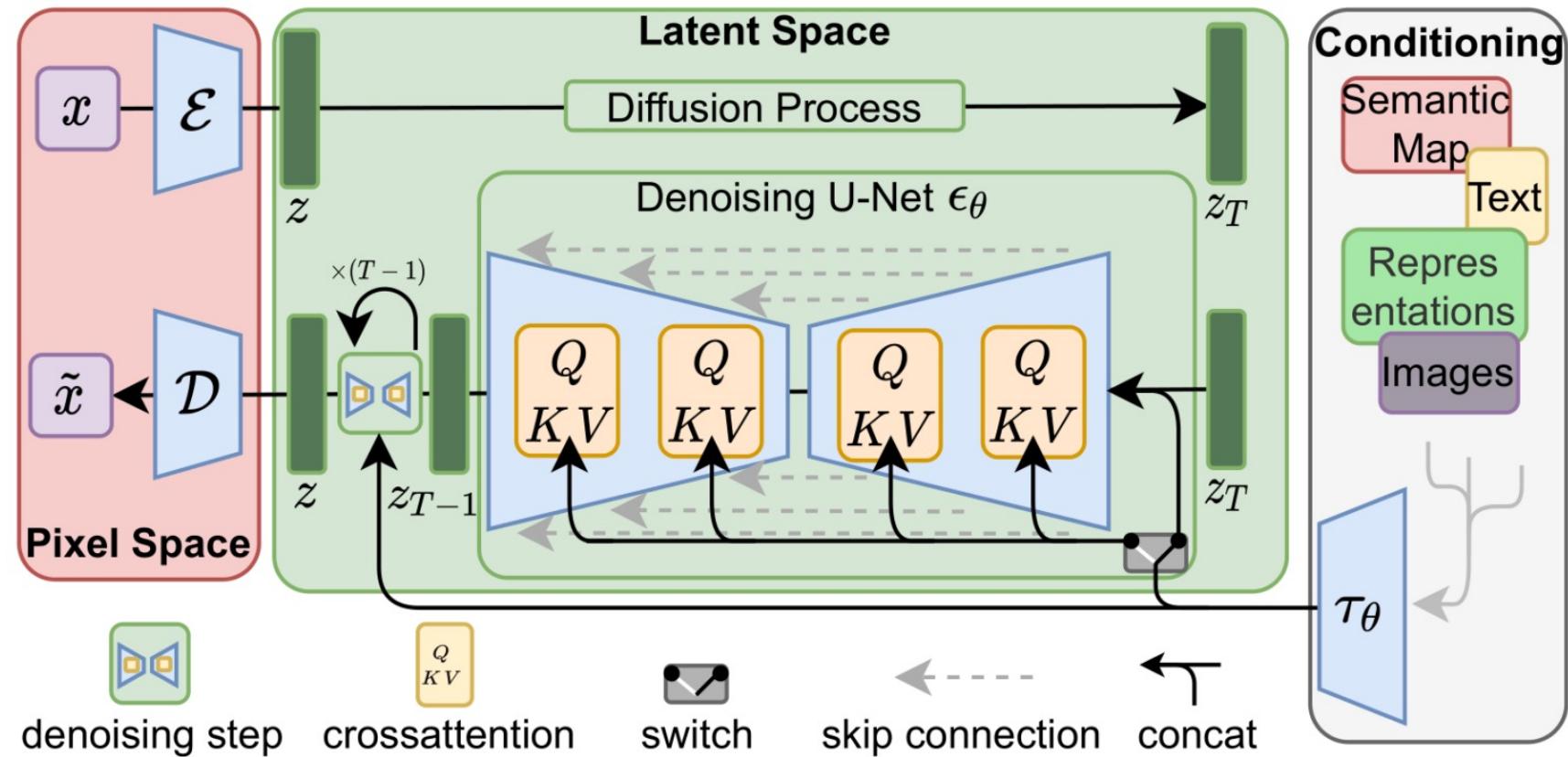
Диффузионные модели

- В диффузионных моделях мы последовательно добавляем Гауссовый шум в исходные данные и учимся восстанавливать его



Stable Diffusion

- Есть кодировщик и декодировщик
- В скрытом пространстве используется диффузионная модель
- Из текста также получаем скрытое состояние и используем в шагах z
- **Модель открытая**



Jay Allamar

<https://jalamar.github.io/illustrated-stable-diffusion/>

Генерация 3D

- Создание новых данных в облаке точек с помощью диффузионных моделей

