Генеративно-состязательные нейронные сети и их друзья

Лекция 4

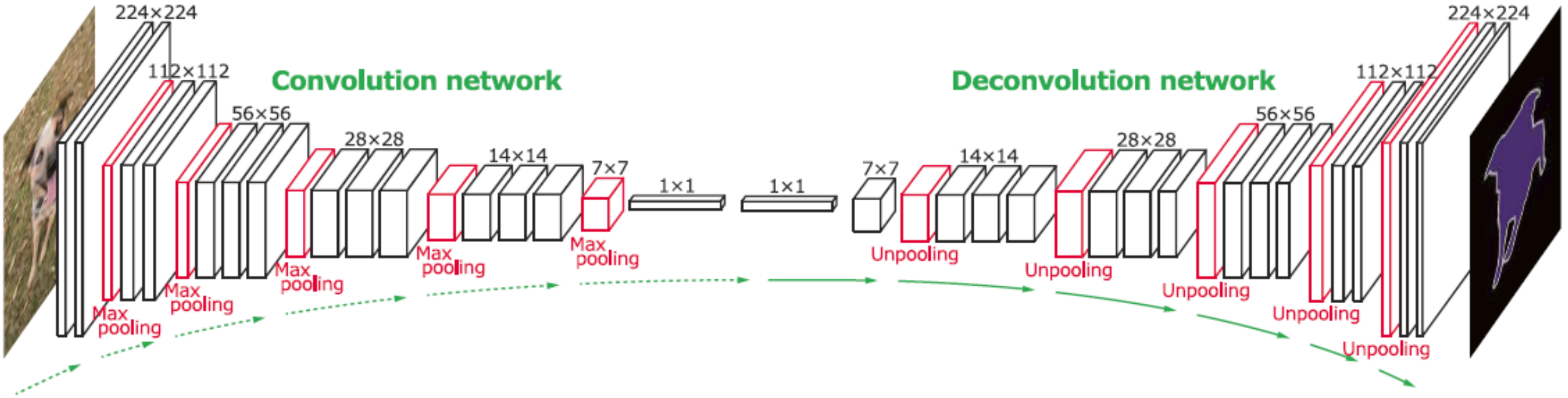
They don't appear to want to compete. They just want to dance.

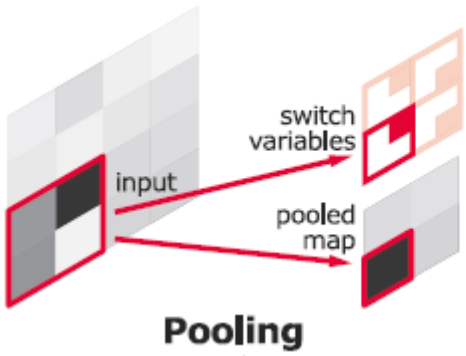
Перминов Андрей Игоревич

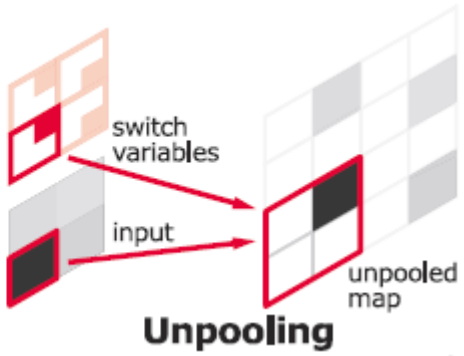
14 октября 2020

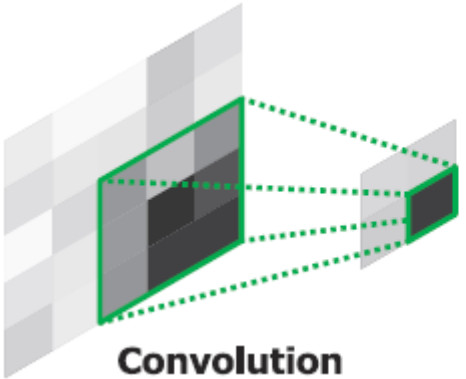
ИСП РАН

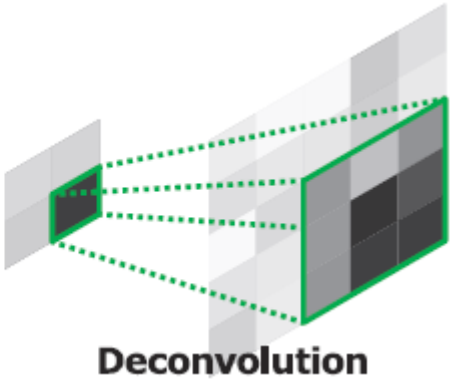
Деконволюционная нейронная сеть



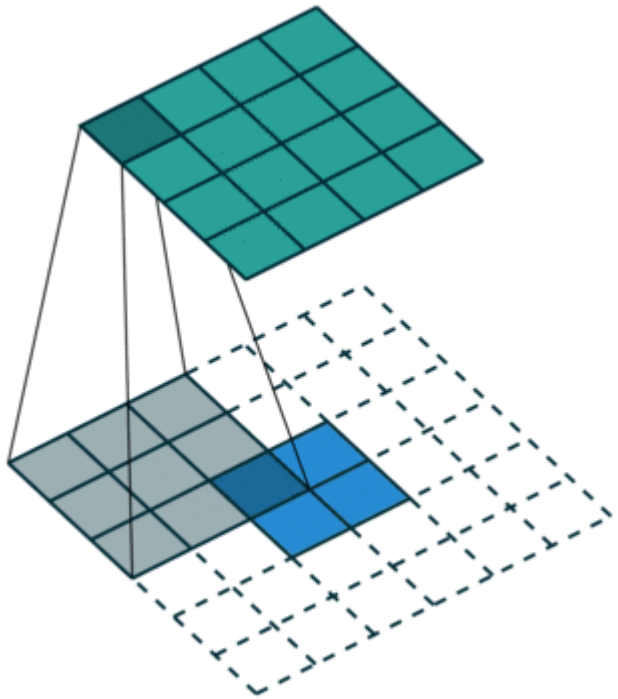








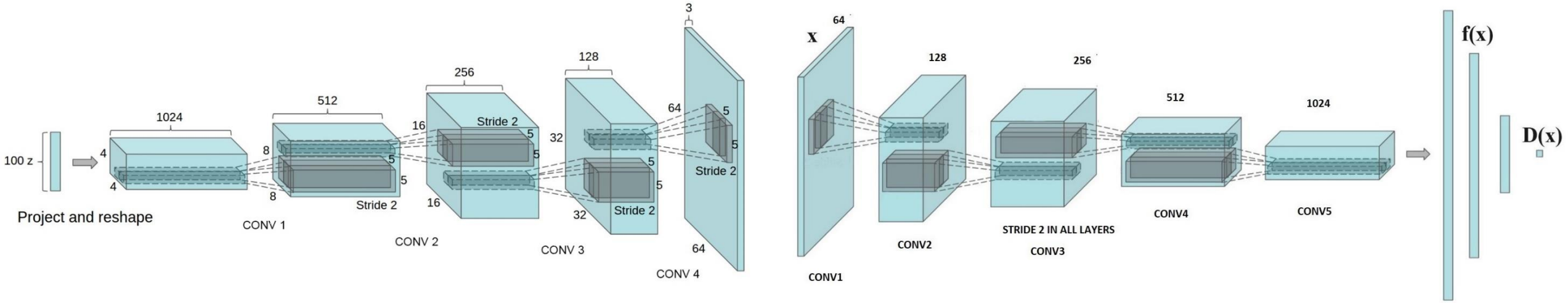
Обратная (транспонированная) свёртка



- позволяет увеличивать размер входного тензора;
- работает аналогично шагу обратного распространения ошибки в свёрточном слое;
- в отличие от обычных интерполяционных слоёв обучается и показывает более высокие результаты;

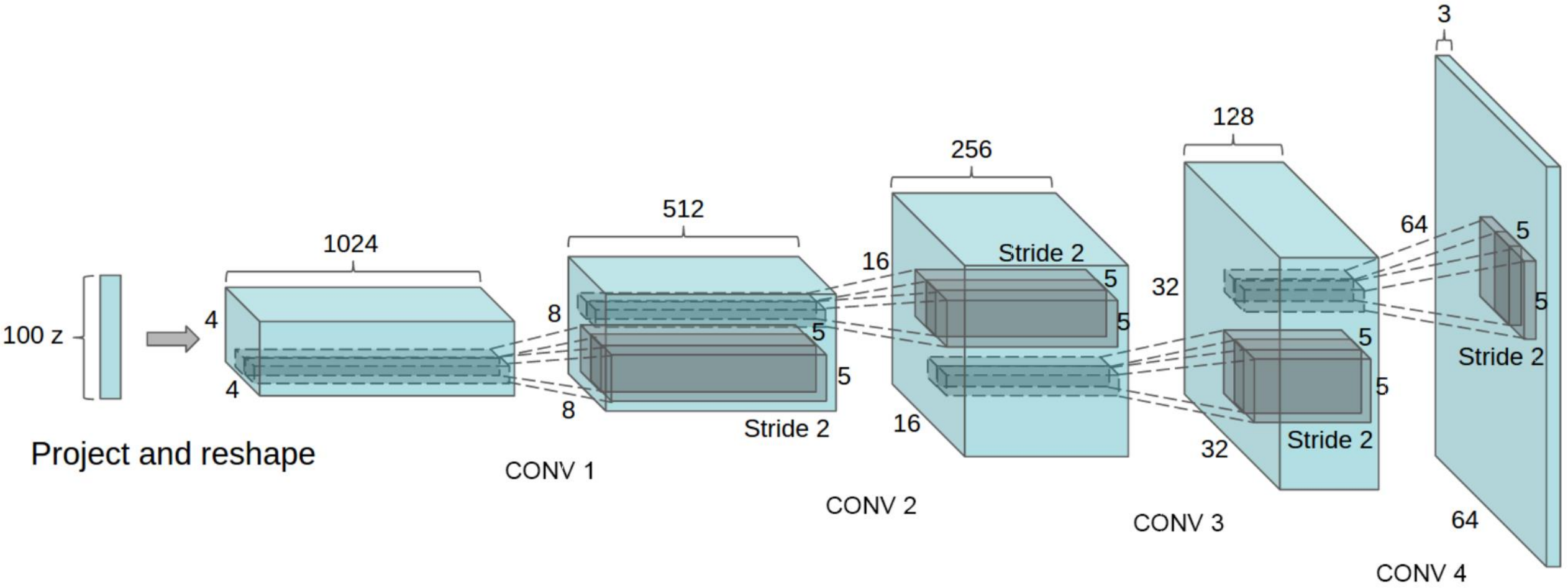
Генеративно-состязательная сеть

Генеративно-состязательная нейросеть (generative adversarial network, GAN) — архитектура, состоящая из двух независимых нейронных сетей, настроенных на работу друг против друга: **генератора** и **дискриминатора**.



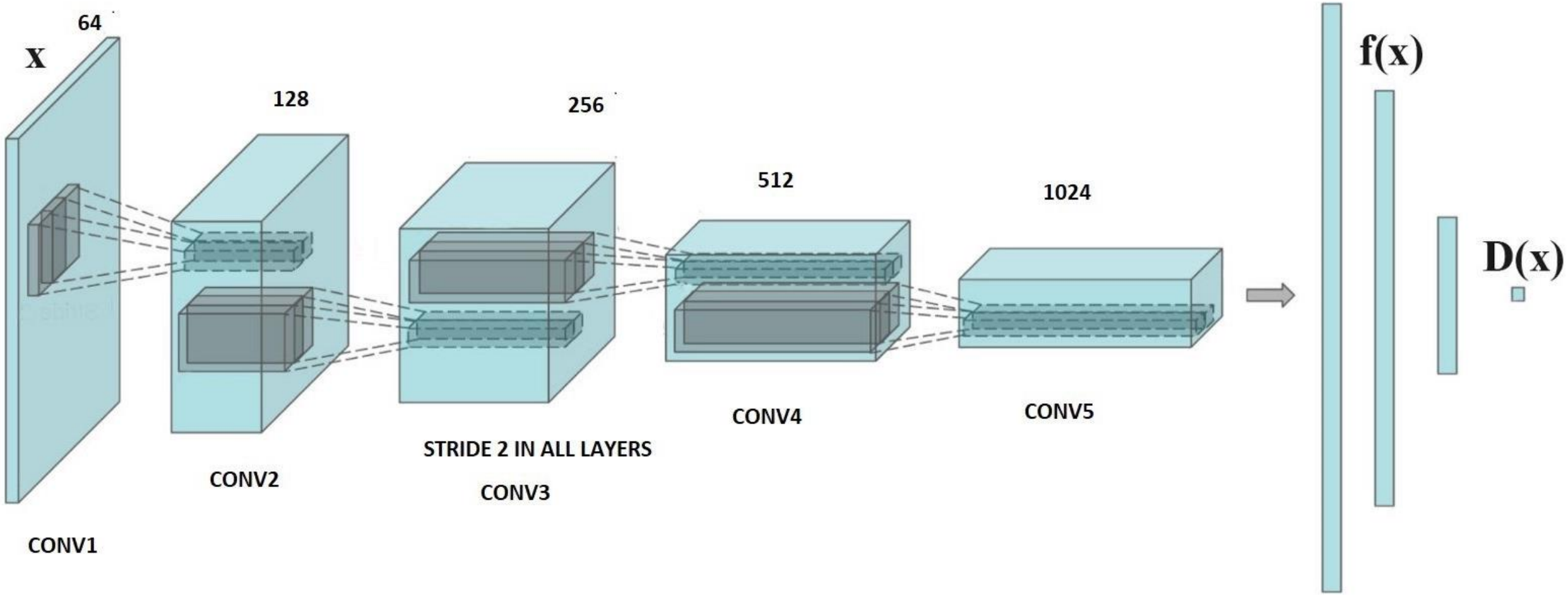
Генератор

Генератор – сеть, создающая изображения, используя вектор небольшой (обычно около 64-256) размерности.



Дискриминатор

Дискриминатор – бинарный классификатор, который учится отличать настоящие изображения от изображений, созданных генератором.



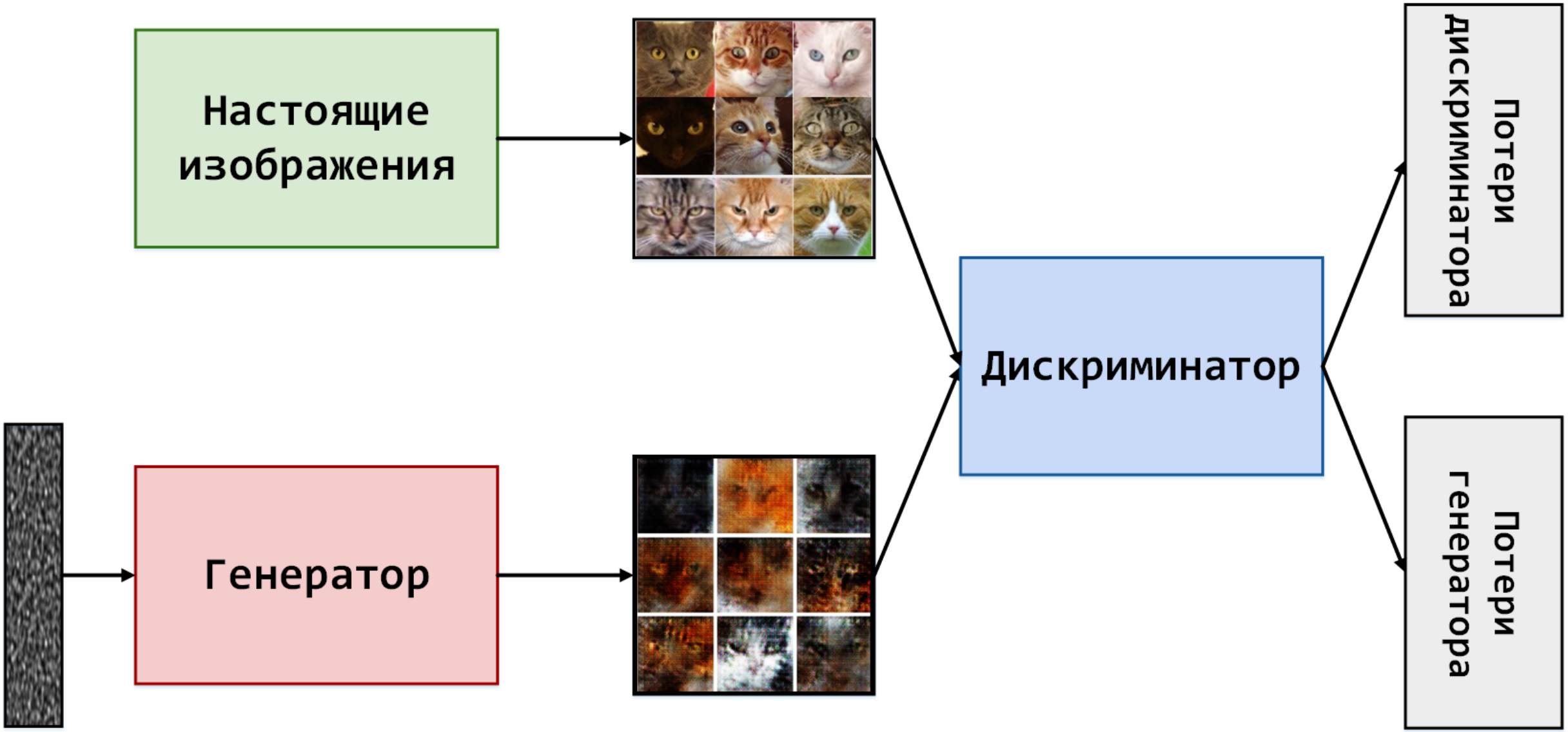
Создаём генератор (tf+keras)

```
generator = keras.Sequential([
    layers.Dense(4 * 4 * 1024, input_shape=(latent_dim,)),
    layers.LeakyReLU(alpha=0.2),
    layers.Reshape((4, 4, 1024)),

    layers.Conv2DTranspose(512, (4, 4), strides=(2, 2), padding='same', kernel_initializer=init),
    layers.BatchNormalization(),
    layers.LeakyReLU(alpha=0.2),

    layers.Conv2DTranspose(256, (4, 4), strides=(2, 2), padding='same', kernel_initializer=init),
    layers.BatchNormalization(),
    layers.LeakyReLU(alpha=0.2),

    layers.Conv2DTranspose(128, (4, 4), strides=(2, 2), padding='same', kernel_initializer=init),
    layers.BatchNormalization(),
    layers.LeakyReLU(alpha=0.2),

    layers.Conv2DTranspose(3, (4, 4), strides=(2, 2), padding='same', activation='tanh'),
], name="generator")
```

Создаём дискриминатор (tf+keras)

```
discriminator = keras.Sequential([
    layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same', kernel_initializer=init, input_shape=image_shape),
    layers.BatchNormalization(),
    layers.LeakyReLU(alpha=0.2),

    layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same', kernel_initializer=init),
    layers.BatchNormalization(),
    layers.LeakyReLU(alpha=0.2),

    layers.Conv2D(256, (5, 5), strides=(2, 2), padding='same', kernel_initializer=init),
    layers.BatchNormalization(),
    layers.LeakyReLU(alpha=0.2),

    layers.Conv2D(512, (5, 5), strides=(2, 2), padding='same', kernel_initializer=init),
    layers.BatchNormalization(),
    layers.LeakyReLU(alpha=0.2),

    layers.Flatten(),
    layers.Dense(1, kernel_initializer=init),
], name="discriminator")
```

Обучение генеративно-состязательной сети



Обучение генеративно-состязательной сети

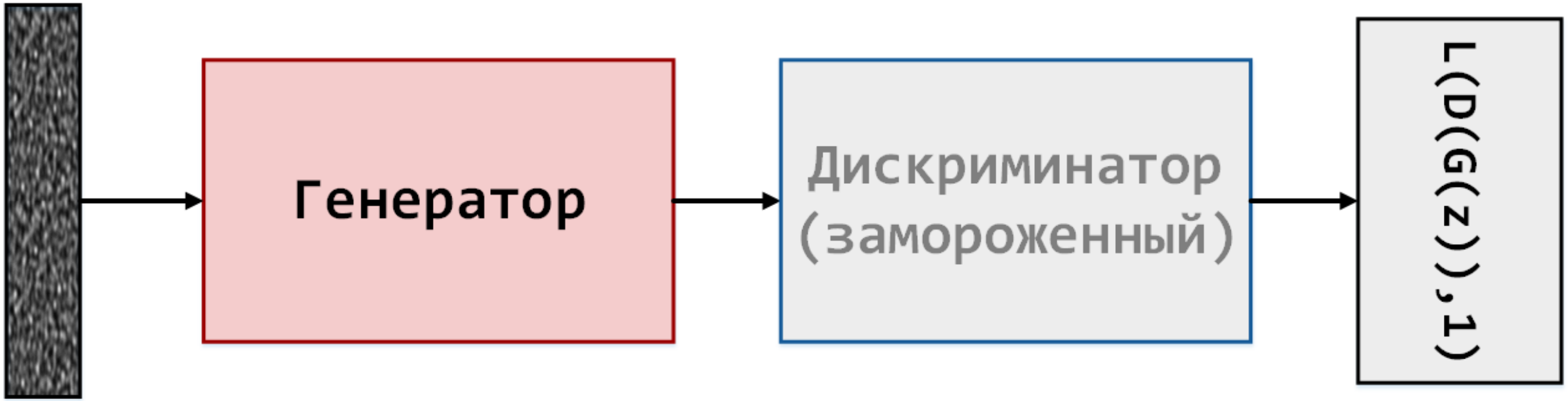
Обучаем дискриминатор предсказывать метку «1» на батче из реальных изображений.

Обучение генеративно-состязательной сети

Обучаем дискриминатор предсказывать метку «0» на батче из изображений, созданных генератором.

Обучение генеративно-состязательной сети

Замораживаем весовые коэффициенты дискриминатора и обучаем полную сеть предсказывать метку «1» для произвольного входного вектора.



Зачем замораживать веса дискриминатора?

Функция потерь дискриминатора

Функция потерь может быть получена из формулы бинарной перекрёстной энтропии:

$$L(\hat{y}, y) = y \cdot ln\,\hat{y} + (1 - y) \cdot \ln(1 - \hat{y})$$

При обучении дискриминатора на настоящих данных $y = 1$, а $\hat{y} = D(x)$, следовательно:

$$L(\hat{y}, y) = L(D(x), 1) = \ln D(x)$$

При обучении дискриминатора на изображениях, созданных генератором $y = 0, \hat{y} = D\big(G(z)\big)$:

$$L(\hat{y}, y) = L(D(G(z)), 0) = \ln\Big(1 - D\big(G(z)\big)\Big)$$

Функция потерь дискриминатора

Цель дискриминатора – правильно классифицировать настоящий и поддельный набор данных, для этого функции должны быть максимизированы, а окончательная функция потерь будет выглядеть так:

$$L^{(D)} = \max\left[\ln D(x) + \ln\left(1 - D\left(G(z)\right)\right)\right]$$

Функция потерь генератора

Генератор постоянно борется с дискриминатором и пытается минимизировать уравнение:

$$L^{(G)} = \min\left[\ln D(x) + \ln\left(1 - D\left(G(z)\right)\right)\right]$$

Комбинированная функция потерь

Оба уравнения можно объединить в одно:

$$L = min_G\, max_D \left[\ln D(x) + \ln\left(1 - D\left(G(z)\right)\right)\right]$$

В реальности используют две разных функции:

$$\mathrm{L}_{discriminator} = \ln D(x) + \ln\left(1 - D\left(G(z)\right)\right)$$

$$L_{generator} = \ln\left(D\left(G(z)\right)\right)$$

Пример реализации функции потерь (tf+keras)

```
class GANLoss:
    def __init__(self, from_logits=True, smoothing=0.4):
        self.bce = tf.keras.losses.BinaryCrossentropy(from_logits=from_logits, label_smoothing=smoothing)

    def discriminator_loss(self, real_output, fake_output):
        real_part = self.bce(tf.ones_like(real_output), real_output)
        fake_part = self.bce(tf.zeros_like(fake_output), fake_output)

        return real_part + fake_part

    def generator_loss(self, real_output, fake_output):
        return self.bce(tf.ones_like(fake_output), fake_output)
```

Алгоритм обучения GAN

for количество_итераций **do**

сгенерировать набор из ***m*** случайных векторов $\{z_1, z_2, \dots, z_m\}$

выбрать ***m*** настоящих изображений $\{x_1, x_2, \dots, x_m\}$

обновить веса дискриминатора в сторону возрастания градиента:

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

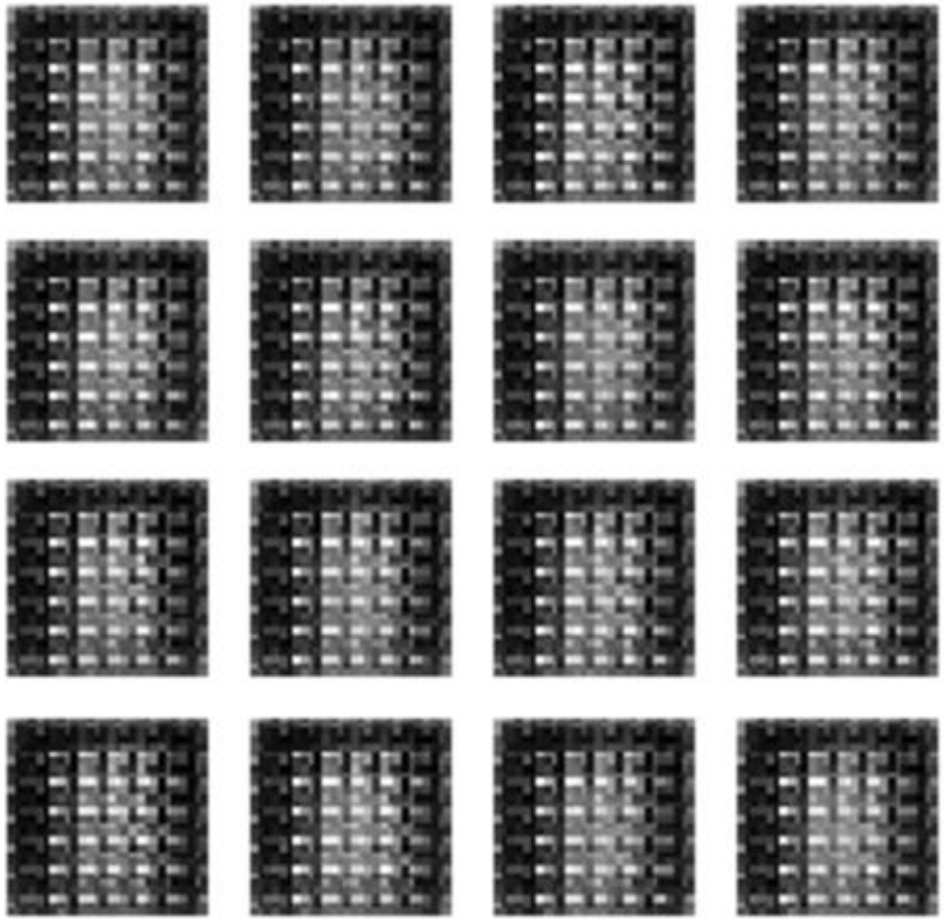
сгенерировать набор из ***m*** случайных векторов $\{z_1, z_2, \dots, z_m\}$

обновить веса дискриминатора в сторону уменьшения градиента:

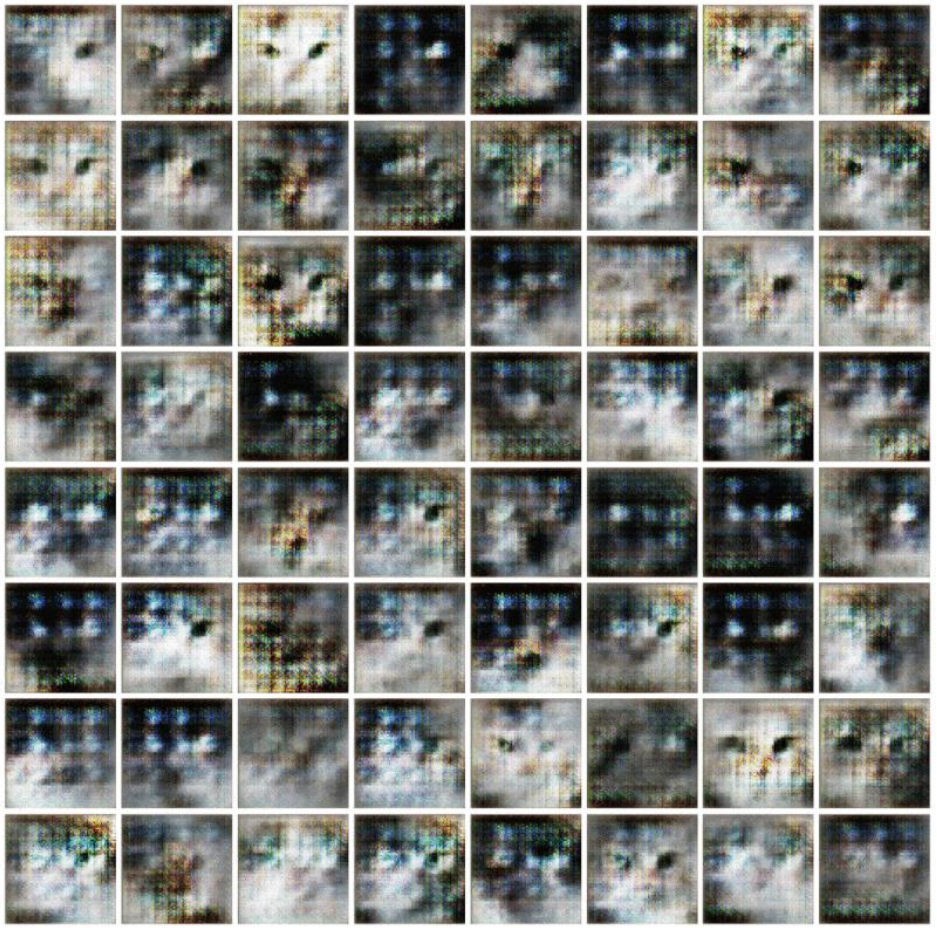
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \left[\ln\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right]$$

end for

Процесс обучения GAN



Обучение на наборе данных MNIST (.gif)



Обучение на наборе данных Cat faces (.gif)

Качественная оценка модели

Для качественной оценки изображений, созданных генератором, можно использовать следующие характеристики:

- сходство с изображениями обучающей выборки;
- отсутствие дубликатов из обучающей выборки;
- разнообразие изображений;
- отсутствие артефактов;

Количественная оценка модели. FID score

Frechet Inception Distance score (FID) – метрика, позволяющая оценить, насколько похожи два набора изображений между собой.

$$FID = \left\|\mu_r - \mu_g\right\|^2 + Tr\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r\Sigma_g\right)^{\frac{1}{2}}\right)$$

В качестве значений r и g используется результат работы сети InceptionV3 без последнего (классифицирующего) слоя:

$$r = InceptionV_3(x_{real}), g = InceptionV_3(x_{fake})$$

При каких значения FID модель лучше?

Количественная оценка модели. FID score

Frechet Inception Distance score (FID) – метрика, позволяющая оценить, насколько похожи два набора изображений между собой.

$$FID = \left\|\mu_r - \mu_g\right\|^2 + Tr\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r\Sigma_g\right)^{\frac{1}{2}}\right)$$

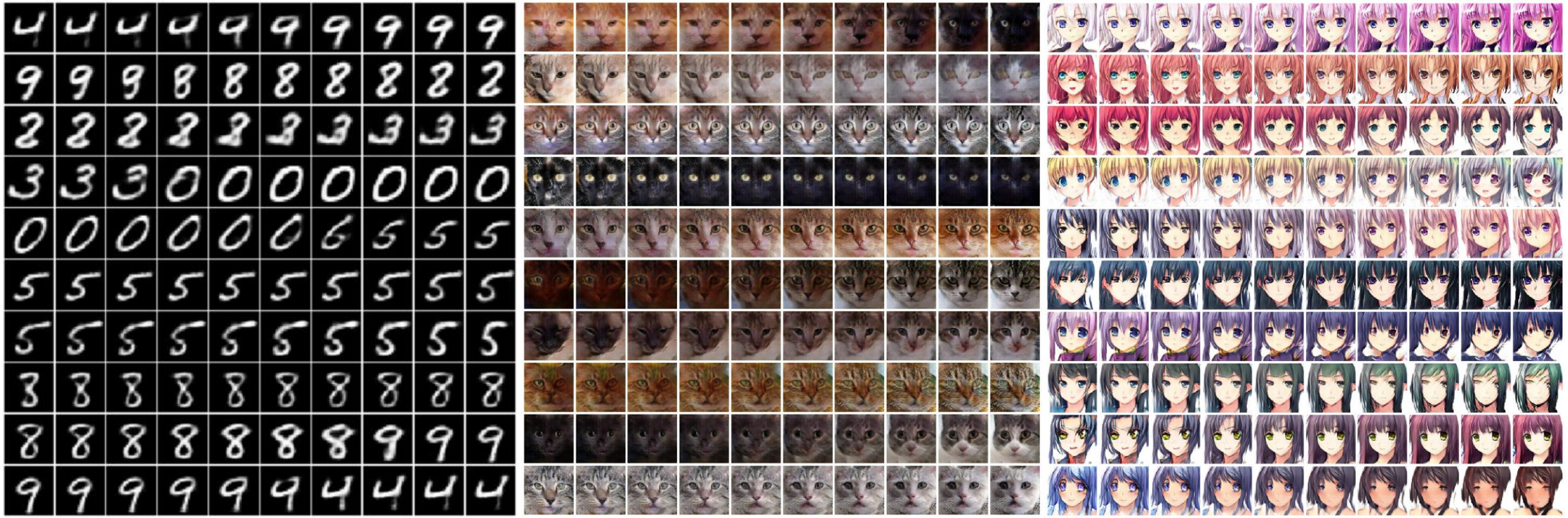
В качестве значений r и g используется результат работы сети InceptionV3 без последнего (классифицирующего) слоя:

$$r = InceptionV_3(x_{real}), g = InceptionV_3(x_{fake})$$

FID – расстояние, поэтому модель тем лучше, чем ниже значение метрики.

Интерполяция в скрытом пространстве

Выбирая два вектора (z_1 и z_2) в скрытом пространстве и выполняя линейную интерполяцию между ними, можно получить интерполяцию в пространстве изображений: $z = t \cdot z_1 + (1 - t) \cdot z_2$, где t — число $\in [0, 1]$



Ошибки при обучении

Затухающие градиенты – если дискриминатор слишком хорош, то обучение генератора может не сработать из-за затухающих градиентов. Фактически, оптимальный дискриминатор не предоставляет достаточно информации для работы генератора.

Неспособность сойтись. GAN’ы сложно обучать. Многие генеративно-состязательные модели вовсе не способны сойтись в приемлемую точку.

Ошибки при обучении

Коллапс моды – режим работы GAN, при котором генератор выдаёт лишь сильно ограниченное количество изображений.

Приёмы для стабильного обучения

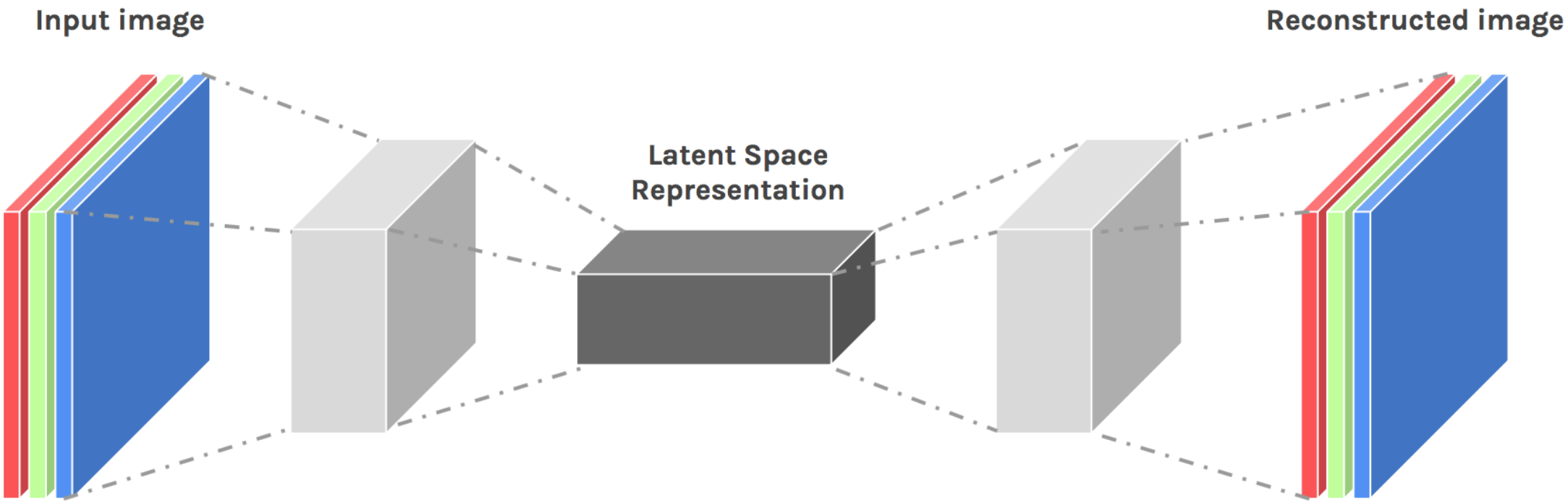
- нормализация входных данных (изображения в диапазоне [-1, 1])
- tanh в качестве функции активации генератора;
- нормальное распределение вместо равномерного для шума;
- раздельные пакеты для настоящих и фейковых изображений;
- использование батч-нормализации;
- использование LeakyReLU вместо ReLU;
- добавление шума в метки ([0, 0.3] и [0.7, 1.2] вместо 0 и 1)
- использование Adam со значением момента 0.5
- использование свёрточных слоёв с шагом > 1 вместо макспулинга;

Друзья GAN’ов

- Автокодировщики
- Вариационные автокодировщики
- WGAN – Wasserstein GAN
- ProGAN – progressive growing GAN
- CGAN – conditional GAN
- Pix2pix
- Cycle GAN
- Stack GAN
- SRGAN – super resolution GAN

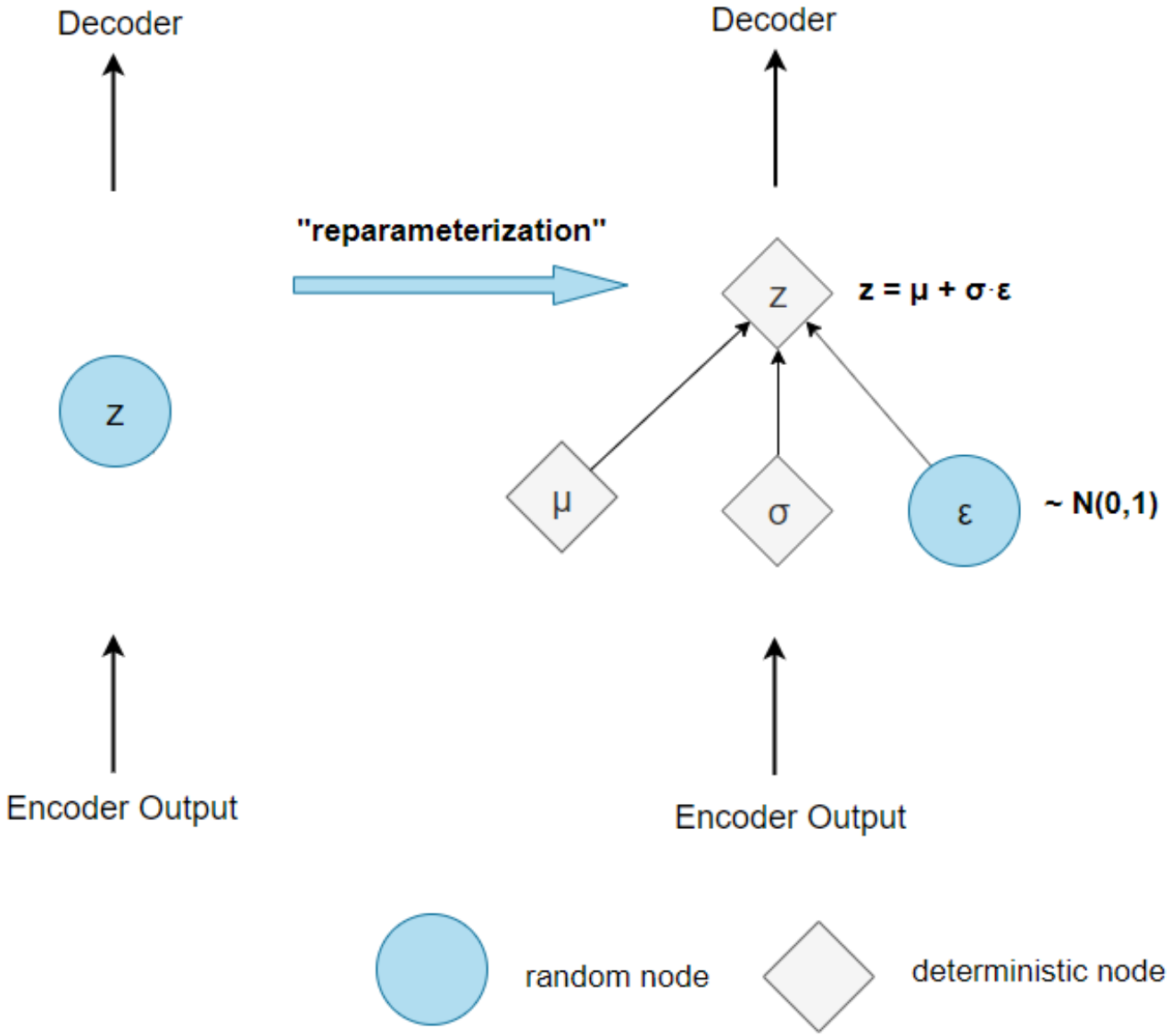
Автокодировщики

Автокодировщик – архитектура нейронной сети, состоящая из кодировщика и декодировщика. Принцип работы заключается в сжатии входного изображения в вектор малой размерности и восстановление исходного изображения из полученного вектора.



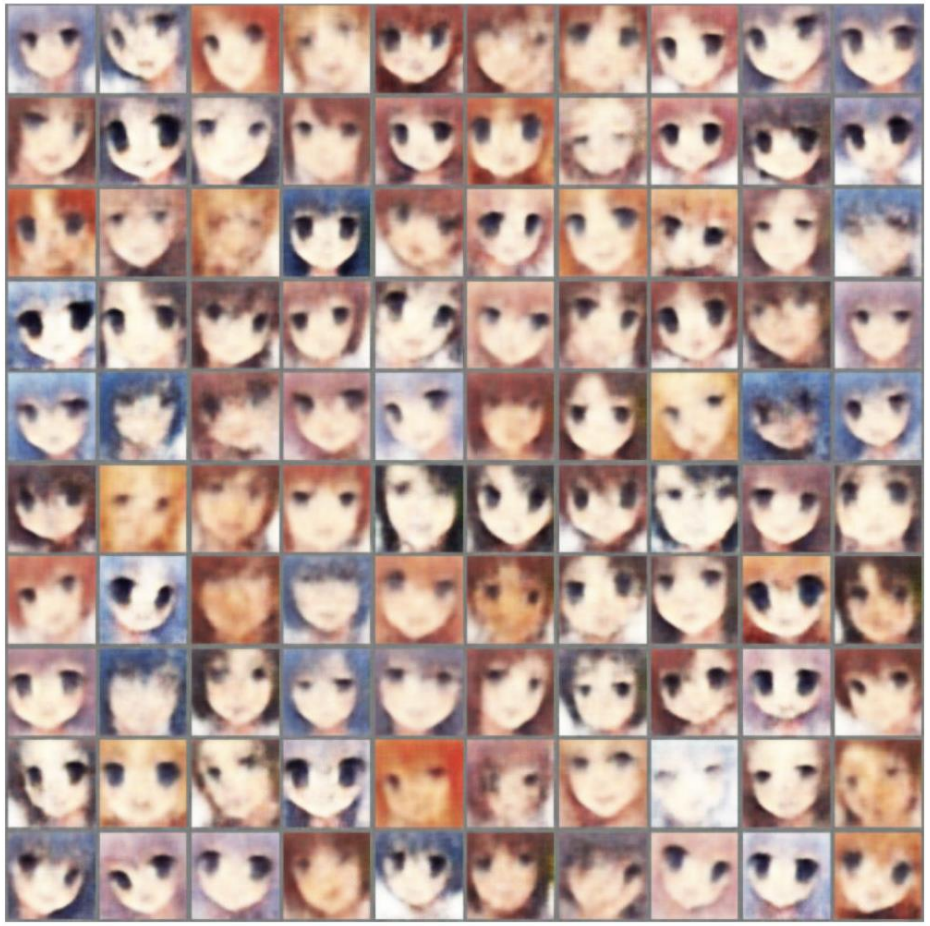
Вариационный автокодировщик

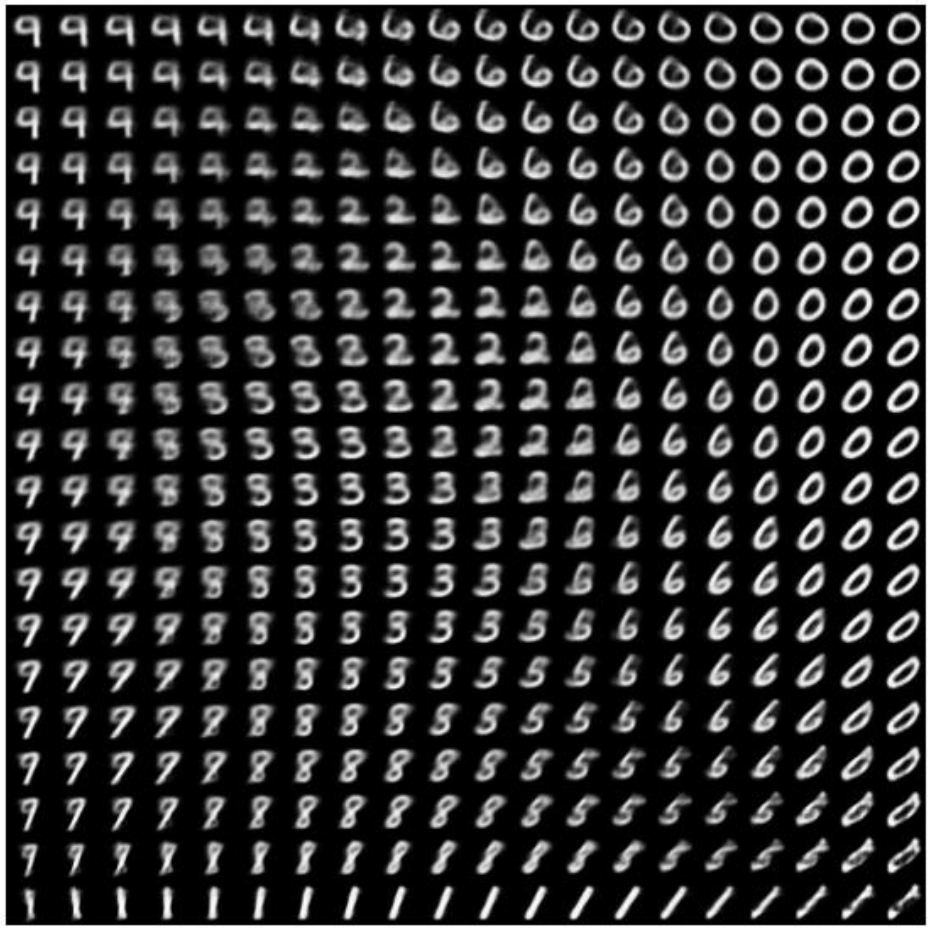
Вариационный автокодировщик – улучшенная версия обычного автокодировщика. Улучшение заключается в добавлении между выходом кодировщика и входом декодировщика слоя репараметризации, который изменяет распределение скрытого пространства, превращая его в нормальное.



Проблема автокодировщика

Изображения, получаемые на выходе вариационных автокодировщиков, зачастую сильно размыты и слабо отличаются от тренировочных примеров.



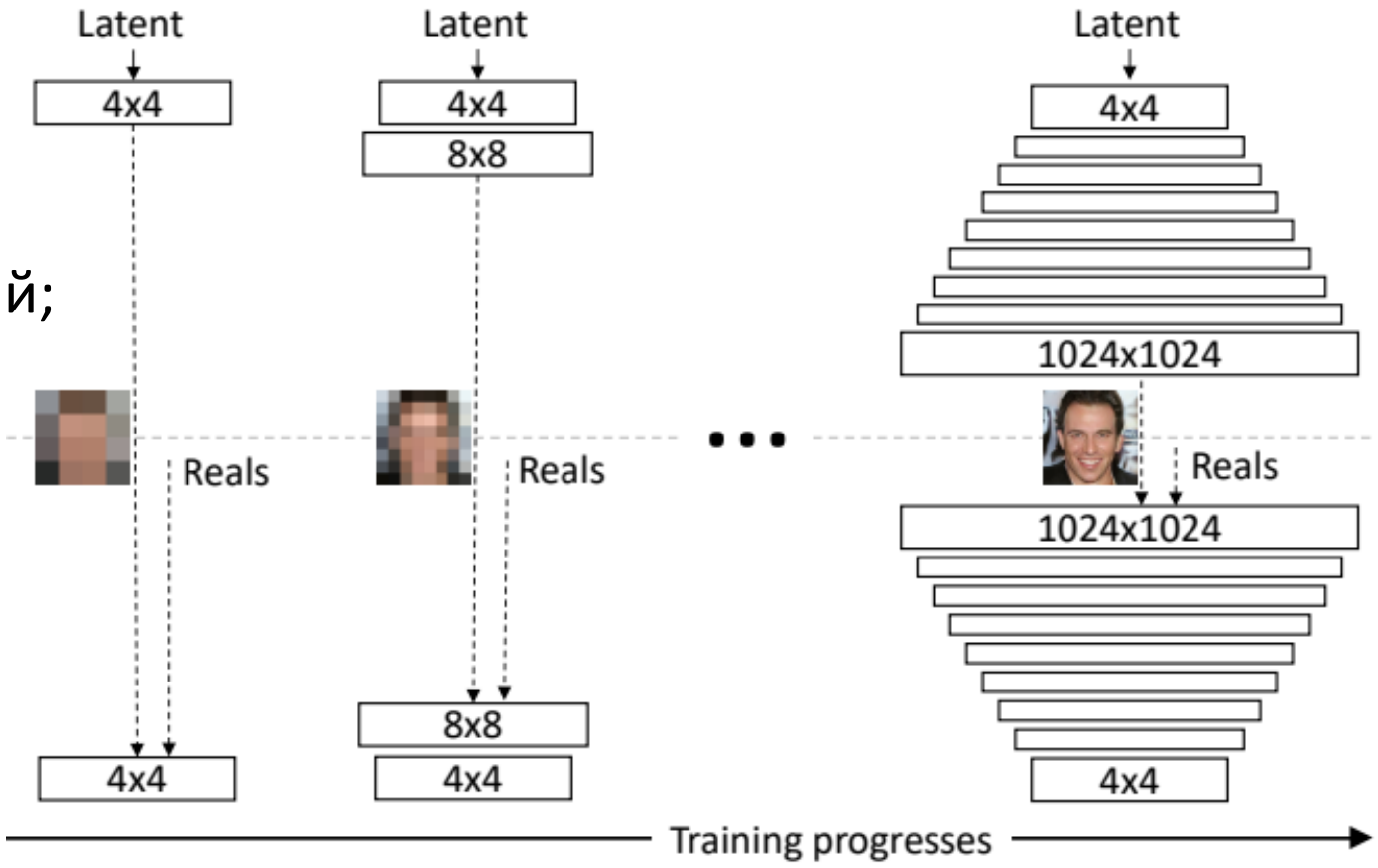


Progressive growing GAN

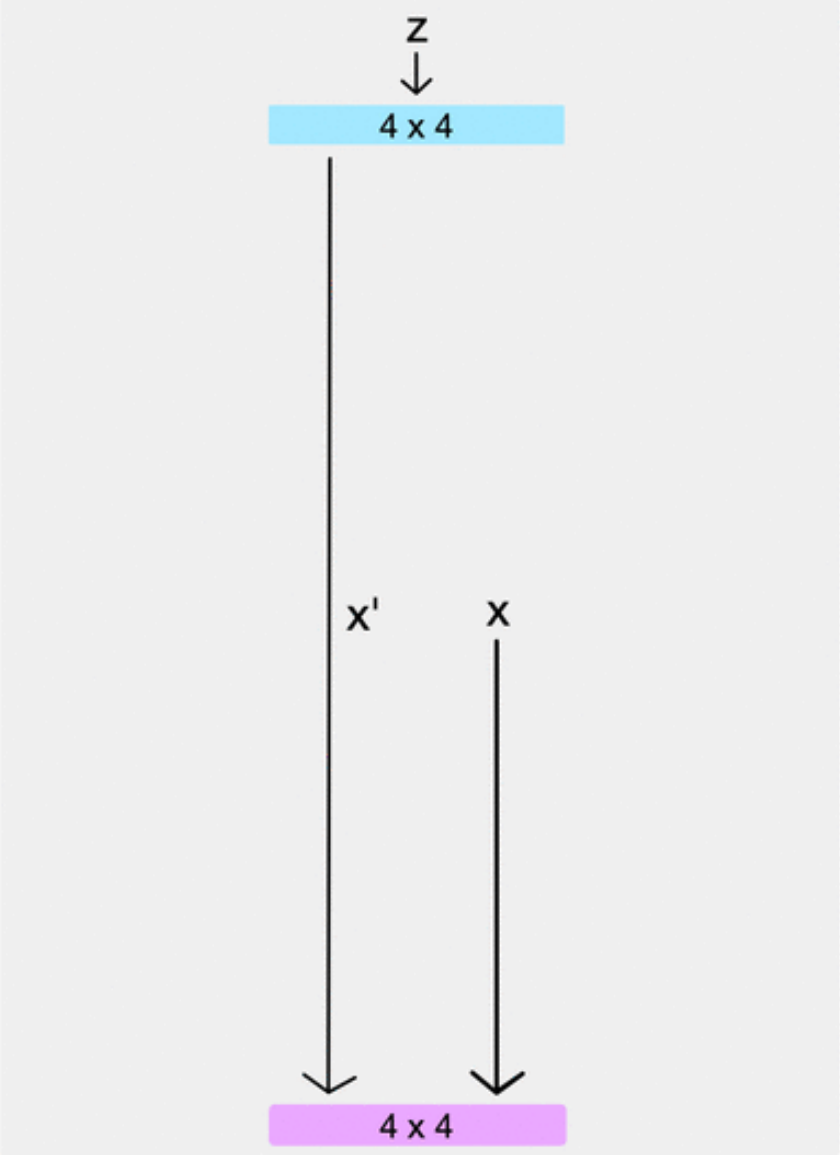
Прогрессивная генеративно состязательная сеть: во время обучения обе нейросети (генератор и дискриминатор) развиваются посредством добавления новых слоёв.

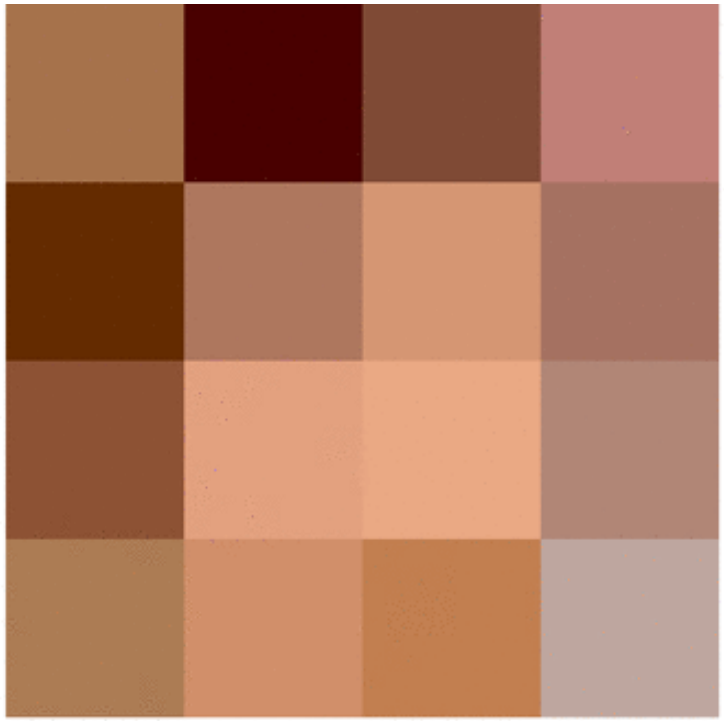
Основные плюсы:

- более реалистичные изображения;
- изображения в высоком качестве;
- большая вариативность изображений;
- легче и быстрее обучается;



Progressive growing GAN (gif)





Training time: 0 days
4x4 resolution

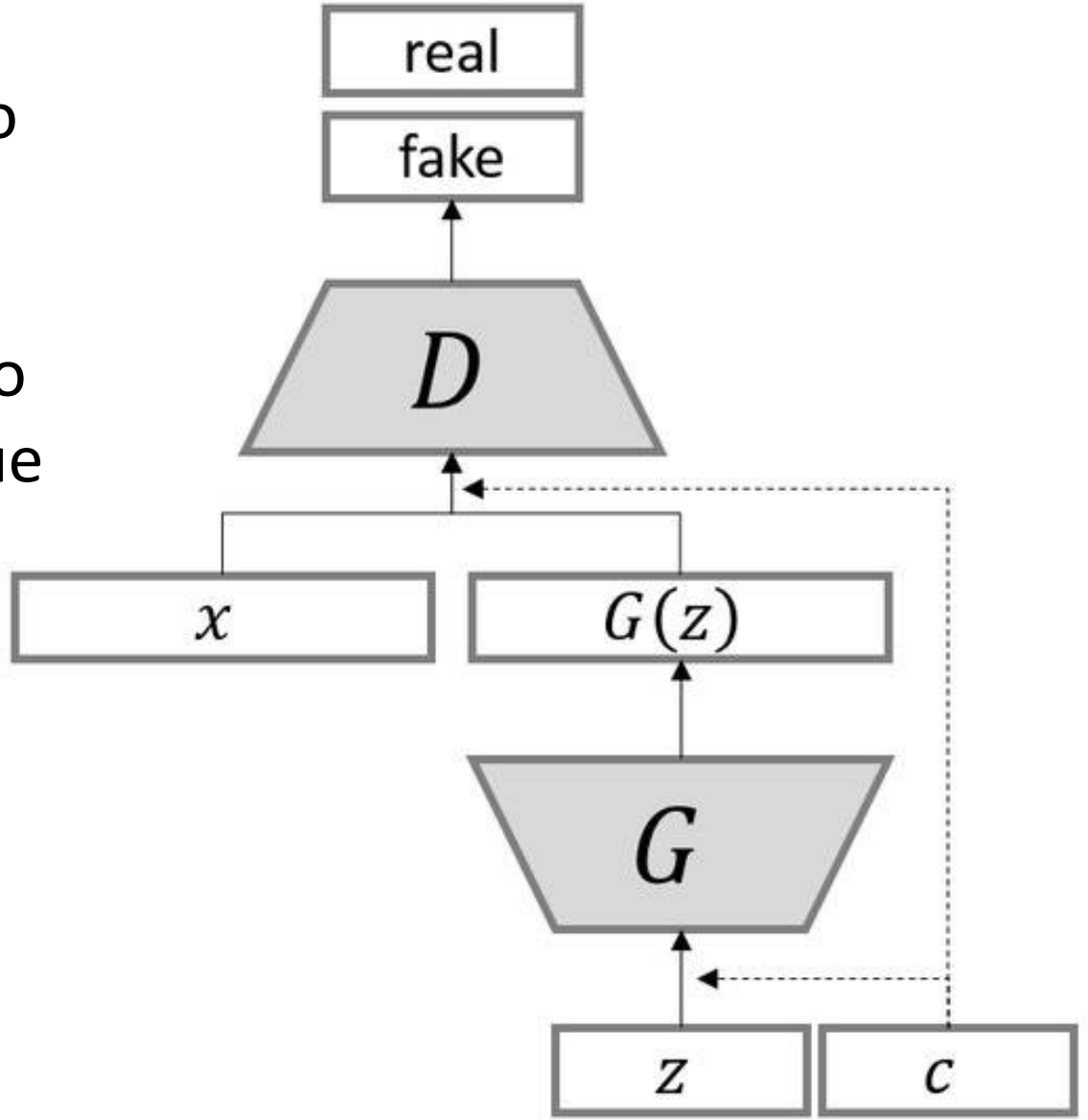
Generator
Discriminator

z = random code
x = real image
x' = generated image

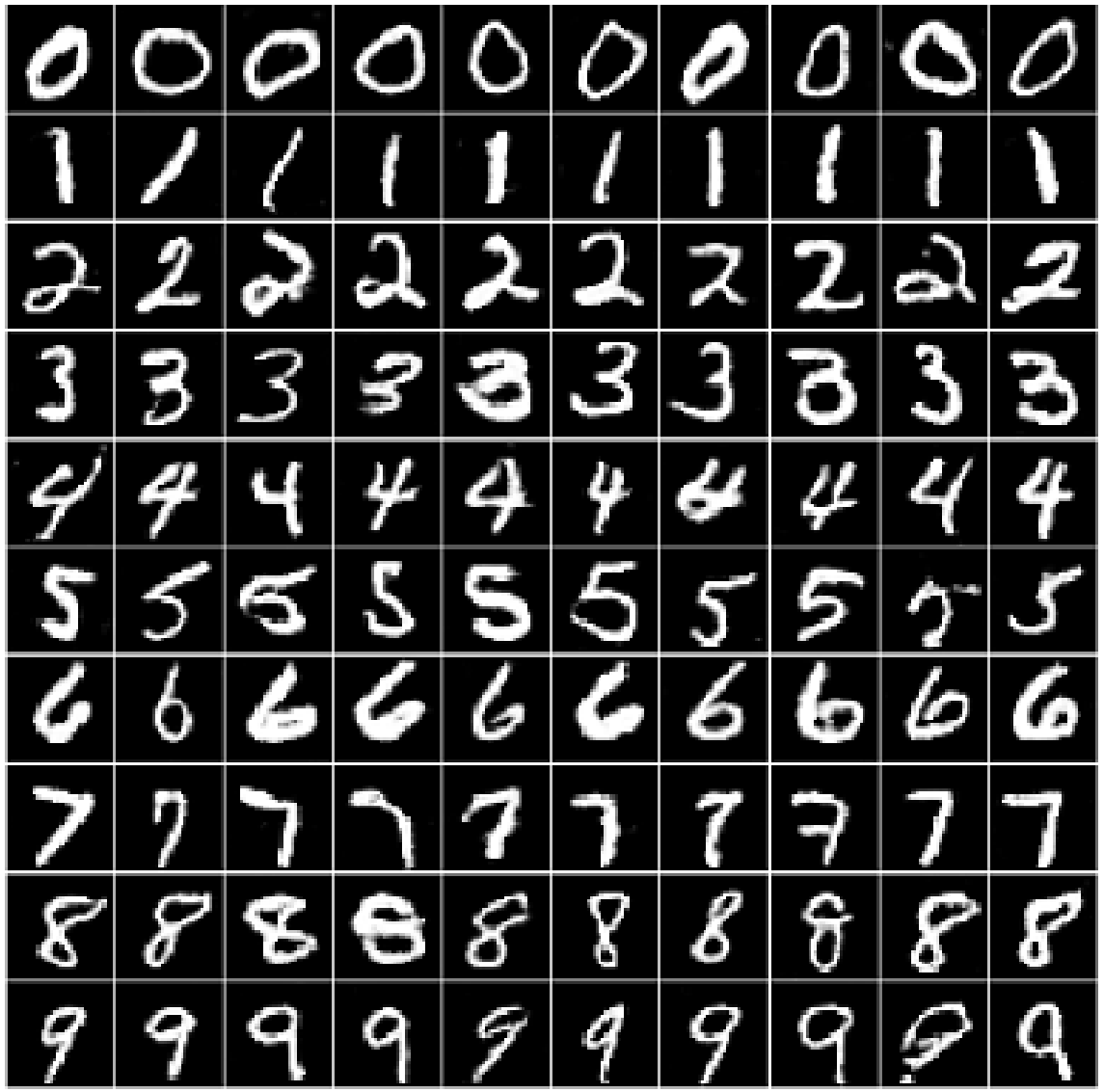
Progressive growing GAN (примеры)

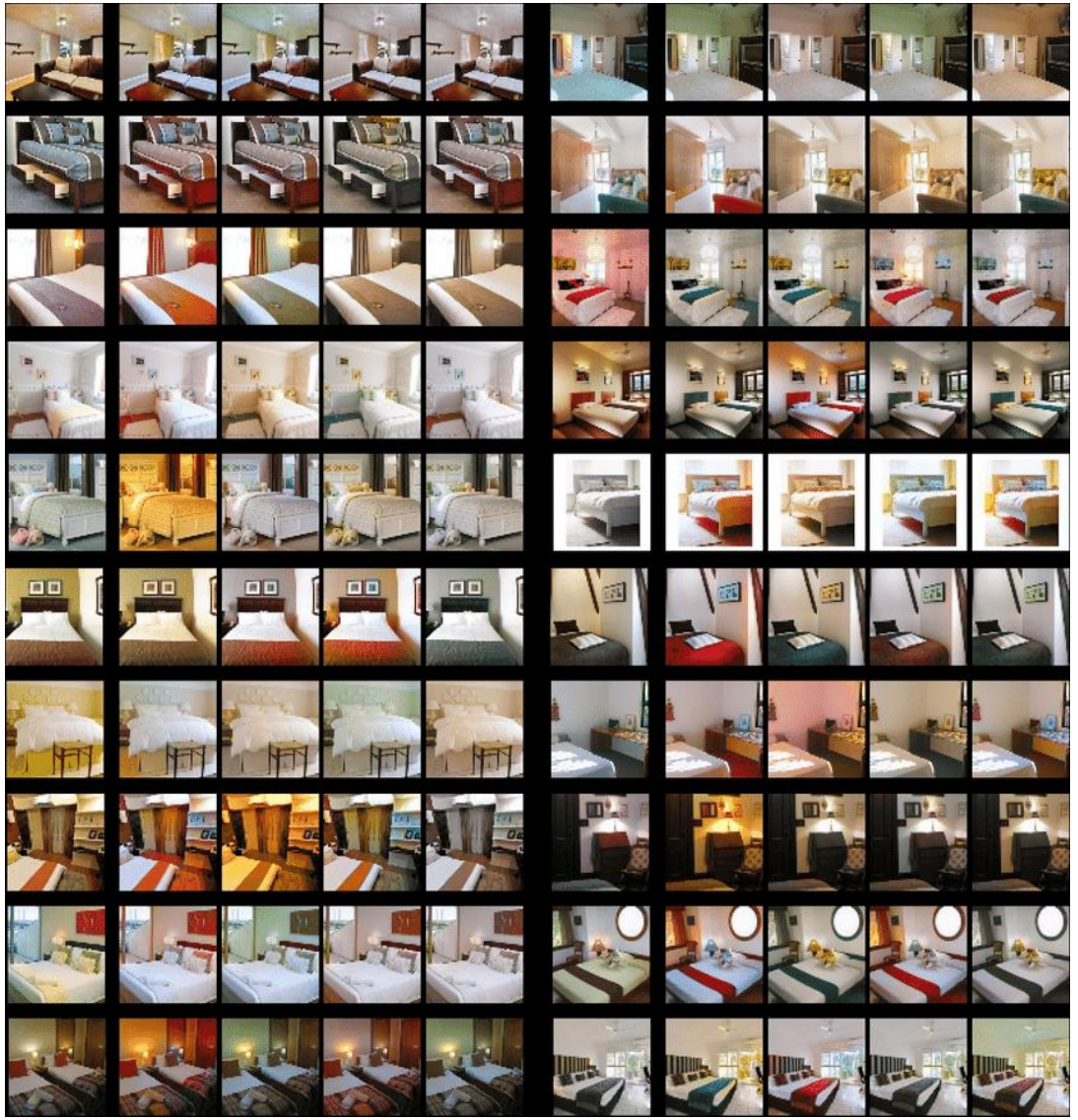
Conditional GAN

Условная генеративно-состязательная сеть – это обычная GAN, в которой помимо изображений ещё используется информация о классах изображений. Подавая на вход генератора метку нужного класса, можно контролировать получаемые изображения.



Conditional GAN (примеры)

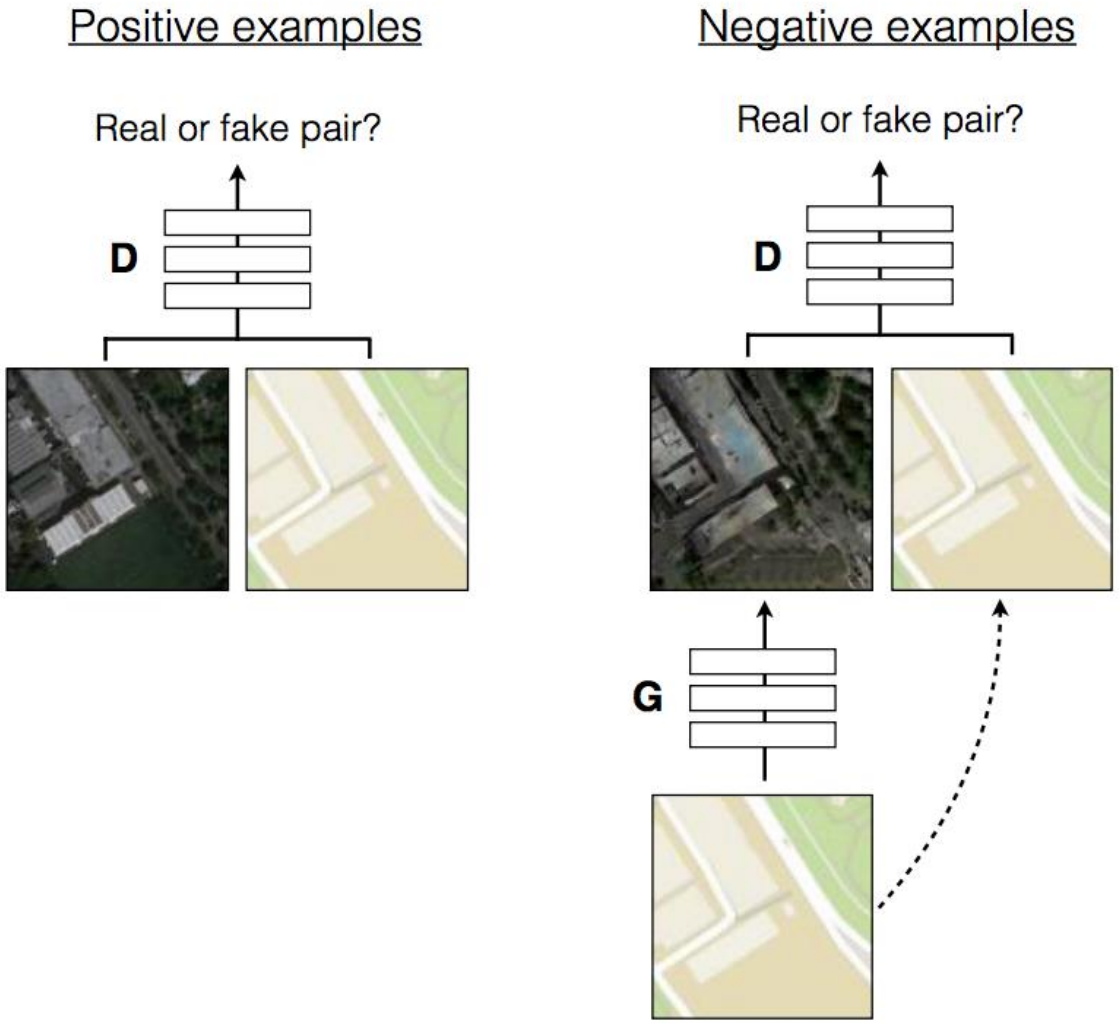




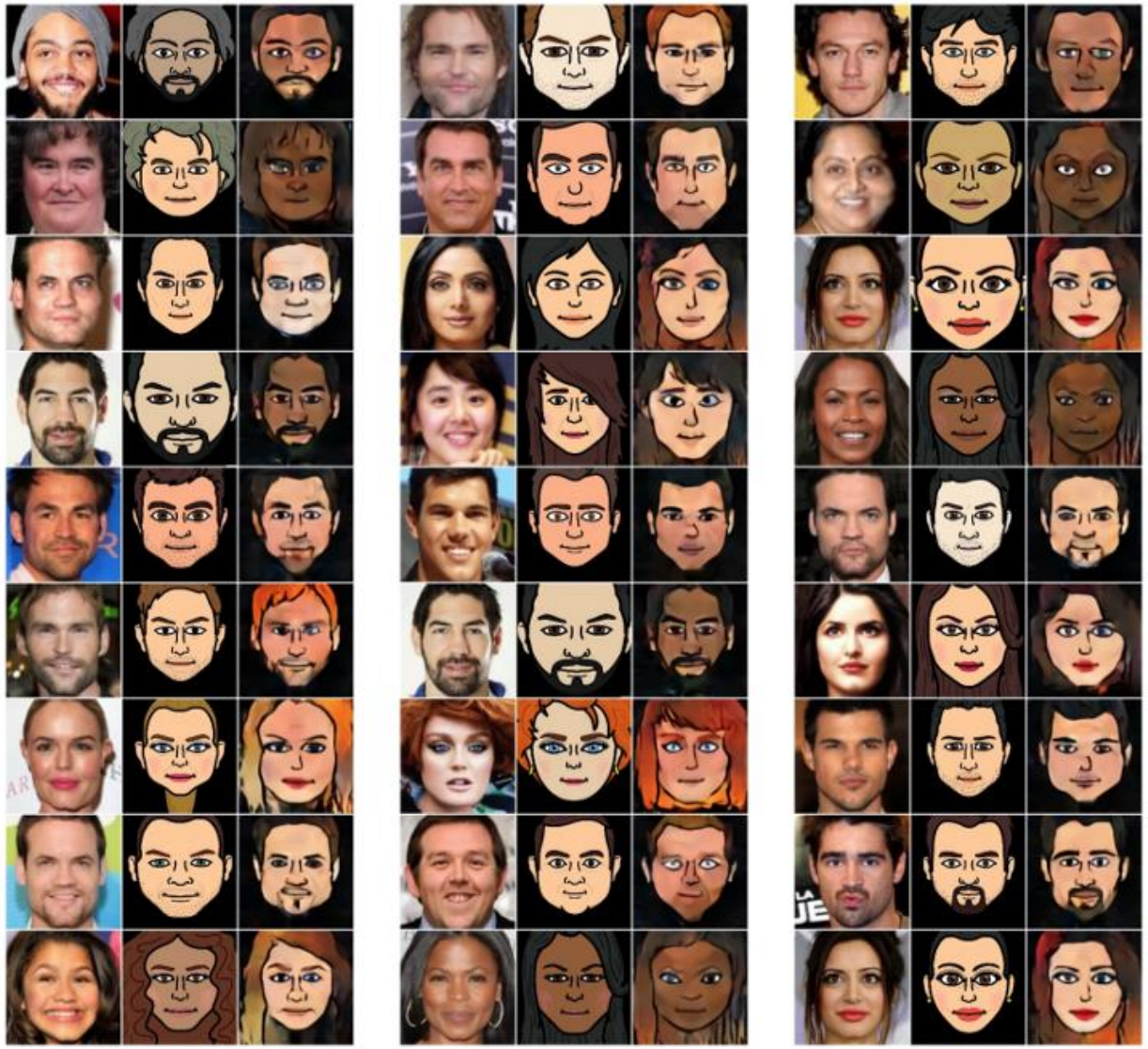
Pix2pix

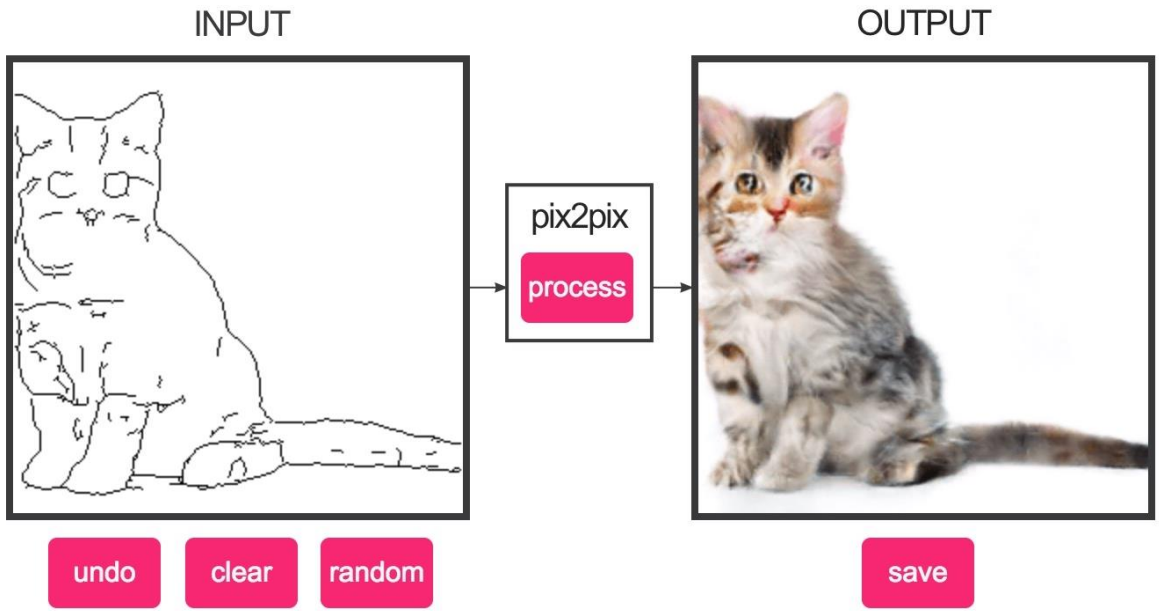
Генератору на вход дается изображение. На ее основе генератор должен сгенерировать картинку на выход.

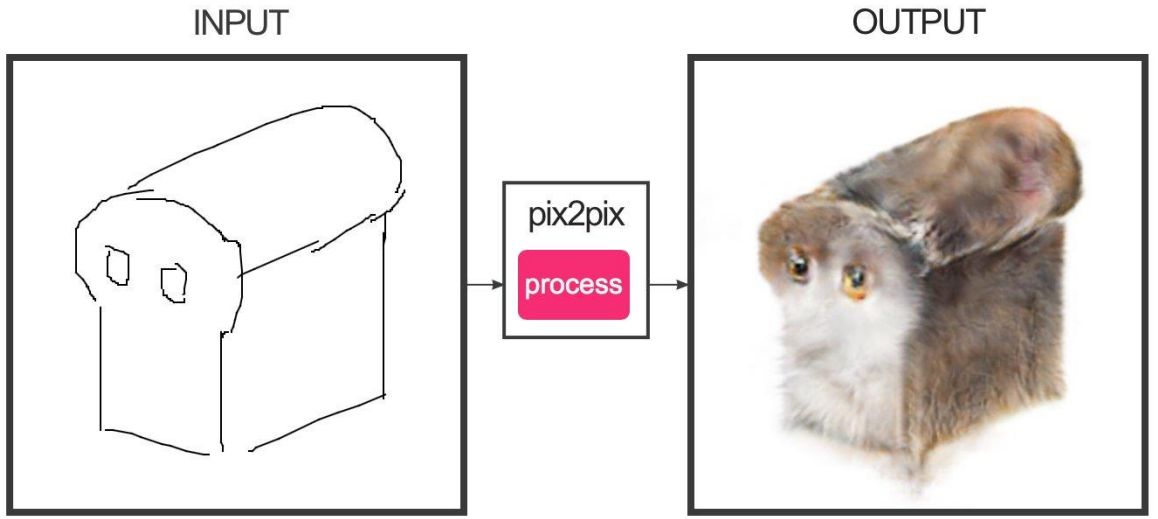
Дискриминатор получает то же изображение и результат работы генератора. Дискриминатор определяет, является ли сгенерированная картинка настоящей или сгенерированной.



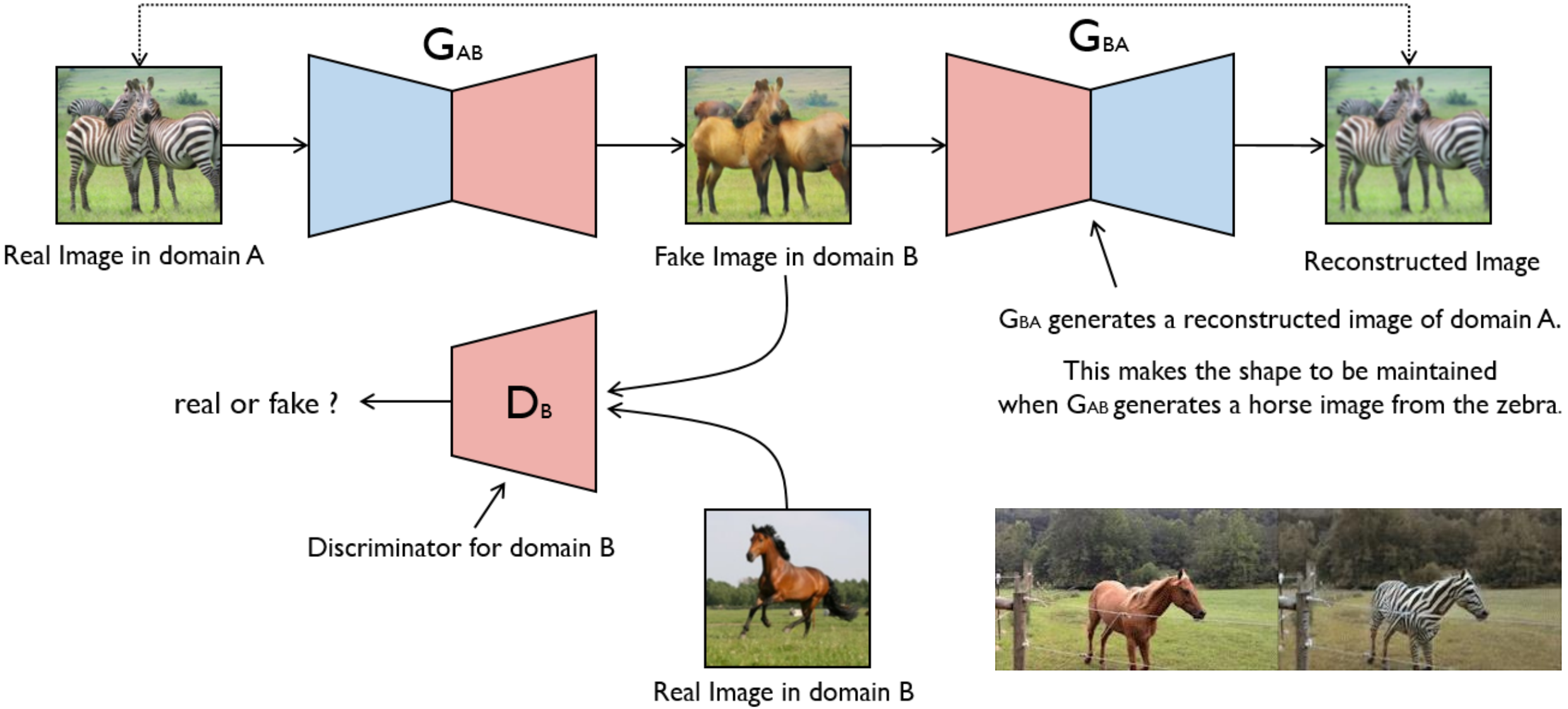
Pix2pix (примеры)



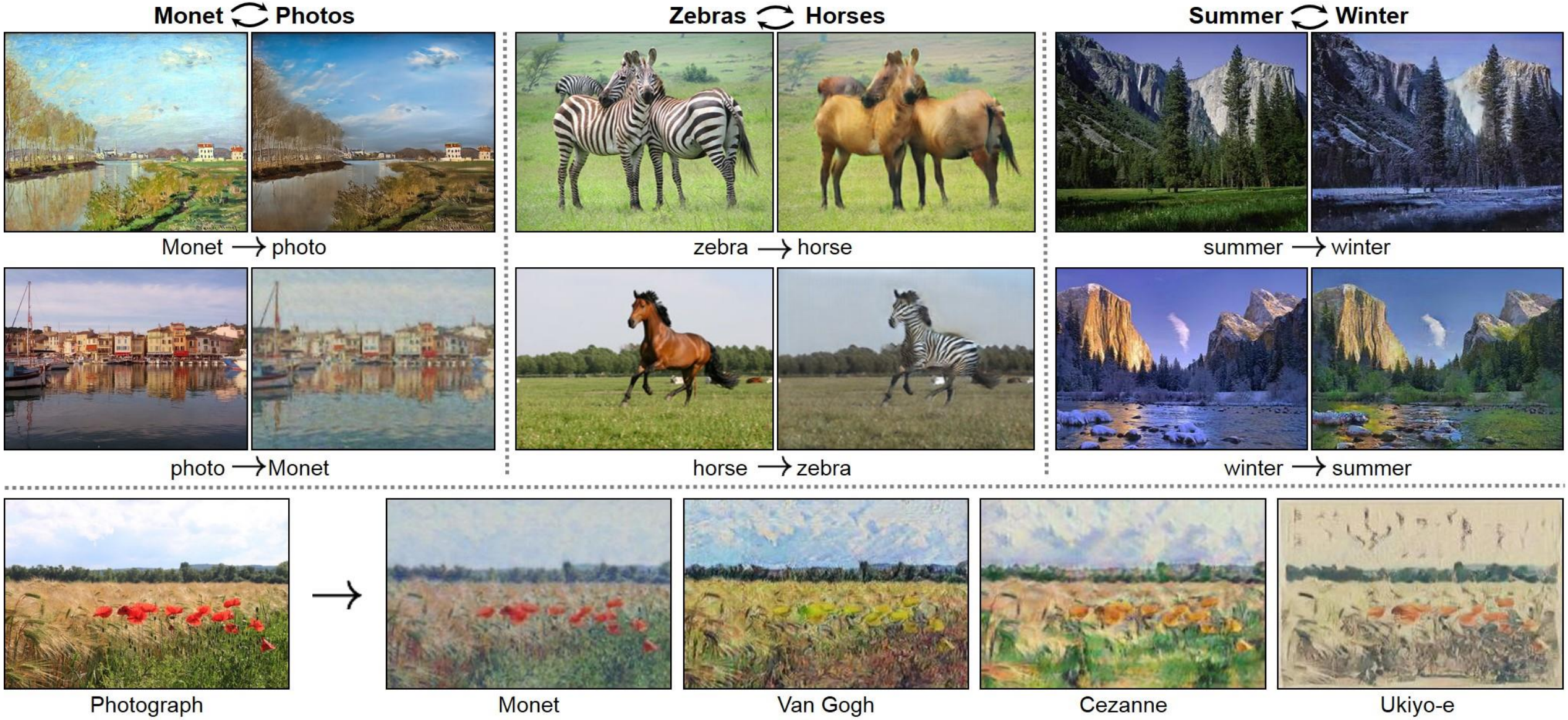




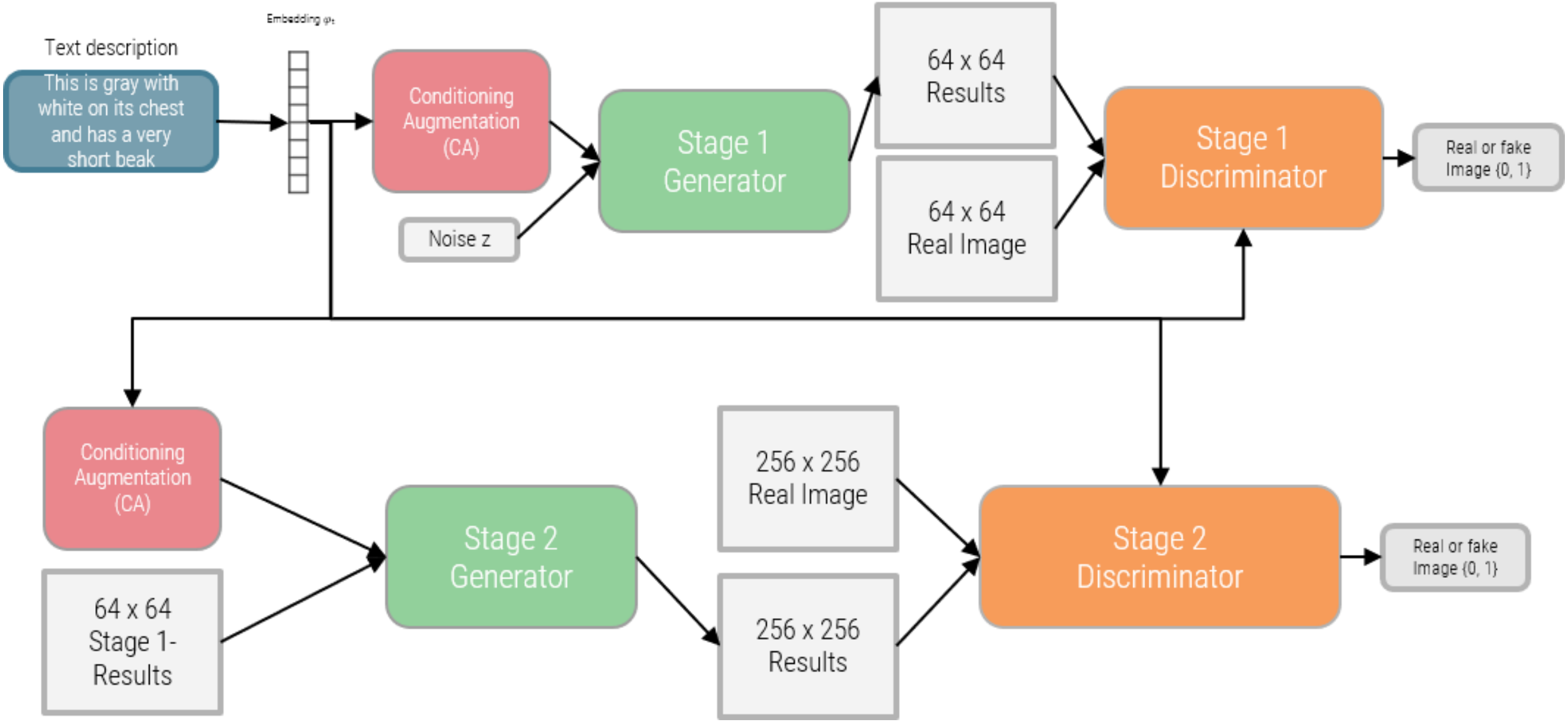
Cycle GAN


G_{AB}
G_{BA}
Real Image in domain A
Fake Image in domain B
Reconstructed Image
G_{BA} generates a reconstructed image of domain A.
This makes the shape to be maintained
when G_{AB} generates a horse image from the zebra.
real or fake ?
D_B
Discriminator for domain B
Real Image in domain B

Cycle GAN (примеры)

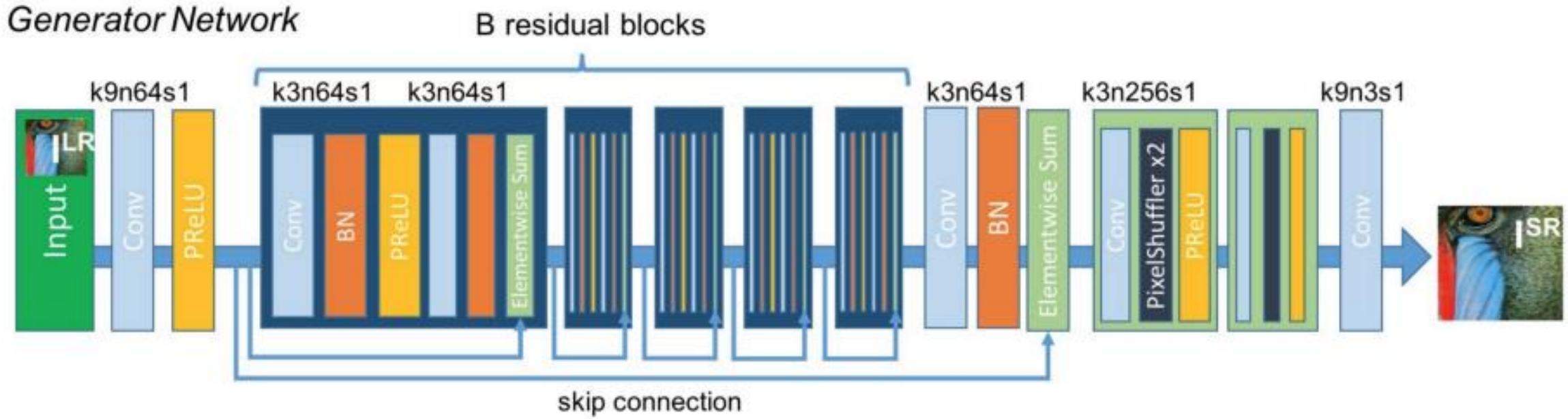


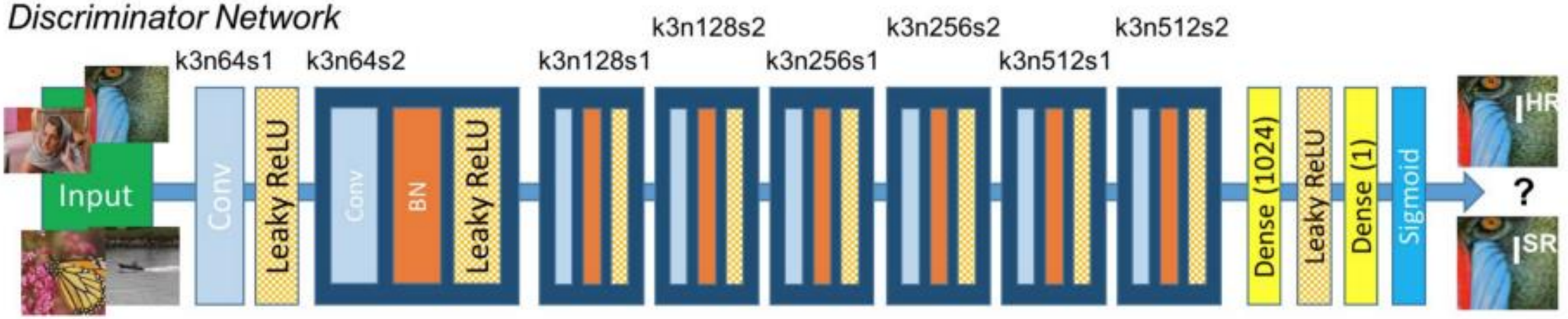
Stack GAN


Text description
This is gray with white on its chest and has a very short beak
Embedding φ_t
Conditioning Augmentation (CA)
Noise z
Stage 1 Generator
64 x 64 Results
64 x 64 Real Image
Stage 1 Discriminator
Real or fake Image {0, 1}
Conditioning Augmentation (CA)
64 x 64 Stage 1-Results
Stage 2 Generator
256 x 256 Real Image
256 x 256 Results
Stage 2 Discriminator
Real or fake Image {0, 1}

Stack GAN (примеры)

Super resolution GAN





Super resolution GAN (пример)

bicubic
(21.59dB/0.6423)

SRResNet
(23.53dB/0.7832)

SRGAN
(21.15dB/0.6868)

original

Полезные ссылки

- GAN — What is Generative Adversarial Networks GAN
- Generative Adversarial Network (GAN) using Keras
- Deep Convolutional Generative Adversarial Network
- How to Train a GAN? Tips and tricks to make GANs work
- Генеративно-состязательная нейросеть (GAN). Руководство для новичков
- GAN loss functions и GAN problems
- Pix2pix: Как работает генератор кошечек
- Introduction to CGAN
- Implementing StackGAN using Keras
- GAN — Wasserstein GAN & WGAN-GP
- GAN — Super Resolution GAN (SRGAN)