

Лекция 13. Генеративные модели. Generative Adversarial Networks.

Глубинное обучение

Антон Кленицкий

Генеративные модели

train data $p_{data}(x) \rightarrow$ model $p_{model}(x) \rightarrow$ sampling

Explicit density estimation

- Явно определяем $p_{model}(x)$
- Autoregressive models, Variational Autoencoder, Diffusion models

Implicit density estimation

- не определяем $p_{model}(x)$ явно, но можем сэмплировать
- Generative Adversarial Networks

Авторегрессионные модели

Авторегрессионная генерация

Упорядочиваем переменные и обрабатываем по очереди, генерируем следующую на основе предыдущих.

$$\begin{aligned} P(x_1, \dots, x_n) &= P(x_1)P(x_2|x_1)\dots P(x_n|x_1, \dots, x_{n-1}) \\ &= \prod_{i=1}^N P(x_i|x_1, \dots, x_{i-1}) \end{aligned}$$

Обучаем модель, которая моделирует это распределение.

Авторегрессионная генерация

Упорядочиваем переменные и обрабатываем по очереди, генерируем следующую на основе предыдущих.

$$\begin{aligned} P(x_1, \dots, x_n) &= P(x_1)P(x_2|x_1)\dots P(x_n|x_1, \dots, x_{n-1}) \\ &= \prod_{i=1}^N P(x_i|x_1, \dots, x_{i-1}) \end{aligned}$$

Обучаем модель, которая моделирует это распределение.

$$P(x_1) \rightarrow x_1$$

$$P(x_2|x_1) \rightarrow x_2$$

$$P(x_3|x_1, x_2) \rightarrow x_3$$

...

Авторегрессионная генерация хороша для текстов, потому что текст - это последовательность

- Языковые модели
- RNN
- Transformer Decoder
- GPT

PixelCNN

Van Den Oord A. et al. (2016) Pixel recurrent neural networks.

- Авторегрессионная модель для генерации изображений
- Предсказываем пиксели как дискретные значения (от 0 до 255) строка за строкой

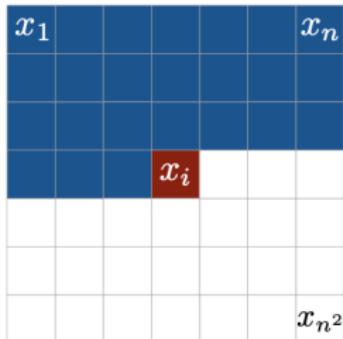
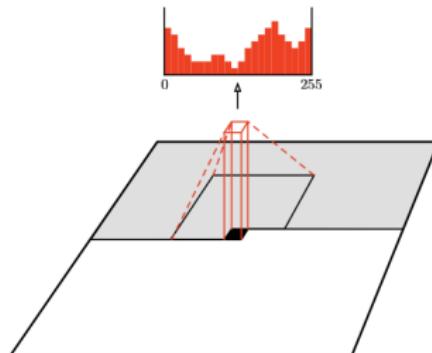


Image credit

Очевидный минус - долгая генерация

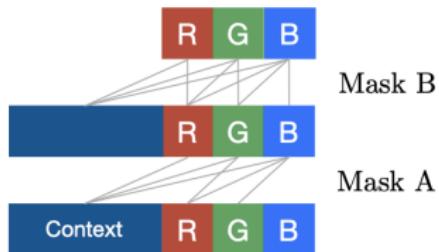
PixelCNN

Masked convolution



1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

Как предсказывать разные каналы?



WaveNet

Van Den Oord A. et al. (2016) WaveNet: A generative model for raw audio.



- Сырые аудио данные очень длинные
- Сырые данные непрерывные - нужна квантизация

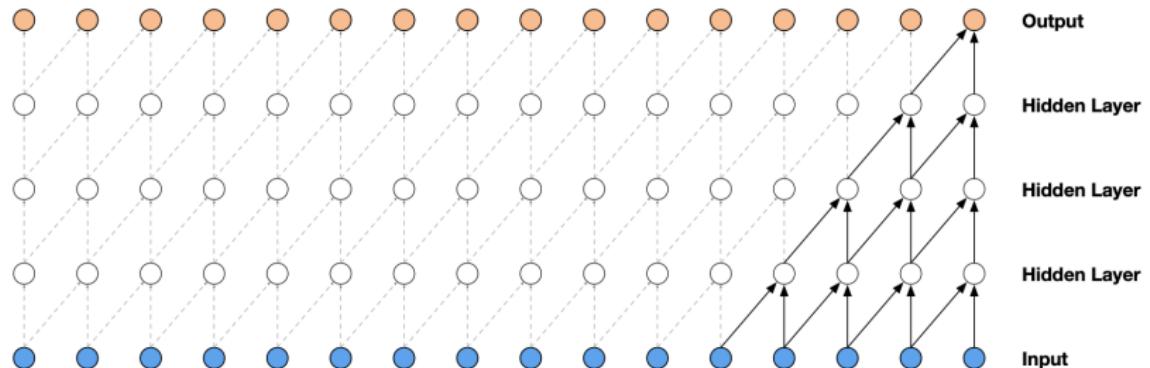
μ -law transformation + quantization

$$f(x) = \text{sign}(x) \frac{\ln(1 + \mu|x|)}{\ln(1 + \mu)},$$

$$-1 \leq x \leq 1, \mu = 255$$

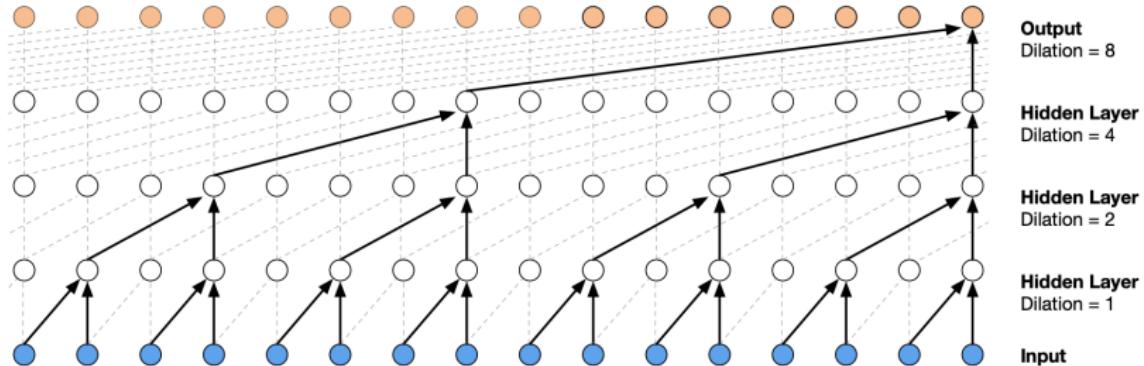
WaveNet

Causal convolutions



WaveNet

Dilated causal convolutions



Последовательность слоев с dilation

1, 2, 4, ..., 512, 1, 2, 4, ...512, 1, 2, 4...

+ residual connections

Generative Adversarial Networks (GAN)

Generative Adversarial Networks

Как заставить нейросеть генерировать новые объекты?

Generative Adversarial Networks

Как заставить нейросеть генерировать новые объекты?

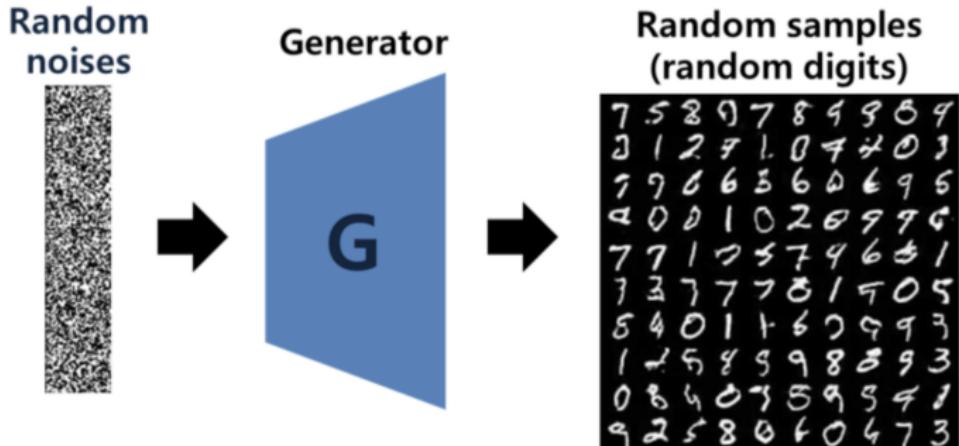
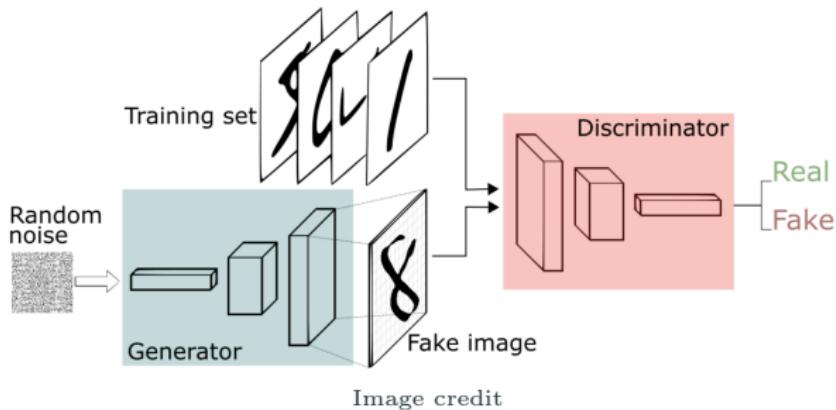


Image credit

- Можно преобразовывать шум в объекты из нужного распределения
- Шум - случайный вектор из нормального распределения

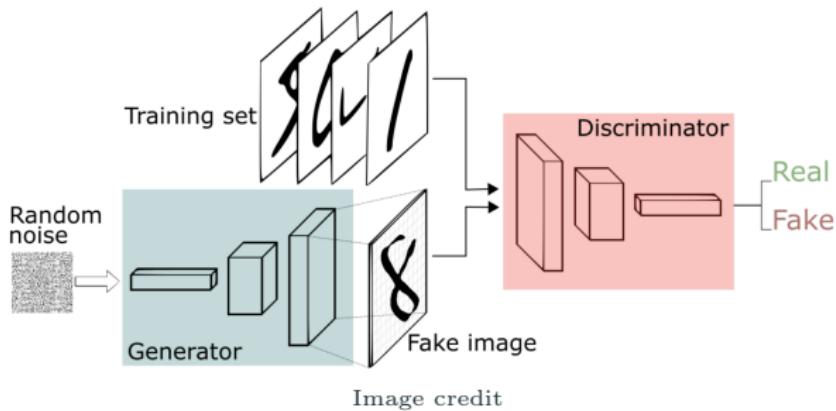
Generative Adversarial Networks

- Generator (G) - сеть, пытающаяся обмануть дискриминатор и синтезировать из шума объекты, не отличимые от реальных
- Discriminator (D) - сеть, отличающая настоящие объекты от синтезированных



Generative Adversarial Networks

- Generator (G) - сеть, пытающаяся обмануть дискриминатор и синтезировать из шума объекты, не отличимые от реальных
- Discriminator (D) - сеть, отличающая настоящие объекты от синтезированных



Функция потерь обучается вместе с данными

Generative Adversarial Networks

Генератор $G(z) : z \rightarrow x$, z - шум, x - объекты

Дискриминатор $D(x) : x \rightarrow [0, 1]$ (бинарная классификация)

Generative Adversarial Networks

Генератор $G(z) : z \rightarrow x$, z - шум, x - объекты

Дискриминатор $D(x) : x \rightarrow [0, 1]$ (бинарная классификация)

Цель дискриминатора - максимизировать величину

$$E_x[\log D(x)] + E_z[\log(1 - D(G(z)))]$$

Generative Adversarial Networks

Генератор $G(z) : z \rightarrow x$, z - шум, x - объекты

Дискриминатор $D(x) : x \rightarrow [0, 1]$ (бинарная классификация)

Цель дискриминатора - максимизировать величину

$$E_x[\log D(x)] + E_z[\log(1 - D(G(z)))]$$

Цель генератора - минимизировать

$$E_z[\log(1 - D(G(z)))]$$

Generative Adversarial Networks

Генератор $G(z) : z \rightarrow x$, z - шум, x - объекты

Дискриминатор $D(x) : x \rightarrow [0, 1]$ (бинарная классификация)

Цель дискриминатора - максимизировать величину

$$E_x[\log D(x)] + E_z[\log(1 - D(G(z)))]$$

Цель генератора - минимизировать

$$E_z[\log(1 - D(G(z)))]$$

Получается minimax game

$$\min_G \max_D E_x[\log D(x)] + E_z[\log(1 - D(G(z)))]$$

Generative Adversarial Networks

Генератор $G(z) : z \rightarrow x$, z - шум, x - объекты

Дискриминатор $D(x) : x \rightarrow [0, 1]$ (бинарная классификация)

Цель дискриминатора - максимизировать величину

$$E_x[\log D(x)] + E_z[\log(1 - D(G(z)))]$$

Цель генератора - минимизировать

$$E_z[\log(1 - D(G(z)))]$$

Получается minimax game

$$\min_G \max_D E_x[\log D(x)] + E_z[\log(1 - D(G(z)))]$$

Из-за затухания градиента в начале обучения (когда $D(G(z)) \approx 0$) вместо $\min_G E_z[\log(1 - D(G(z)))]$ лучше брать $\min_G E_z[-\log D(G(z))]$

Deep Convolutional GAN (DCGAN)

Radford A. et al. (2015) Unsupervised representation learning with deep convolutional generative adversarial networks.

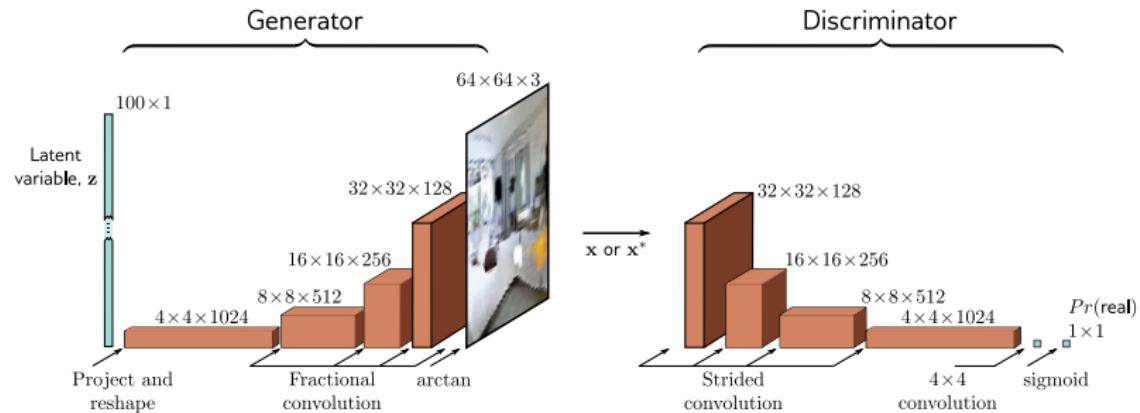


Image credit

Проблемы обучения GAN

- Нестабильное обучение

Сходимость зависит от мелких деталей, высокая
чувствительность к гиперпараметрам

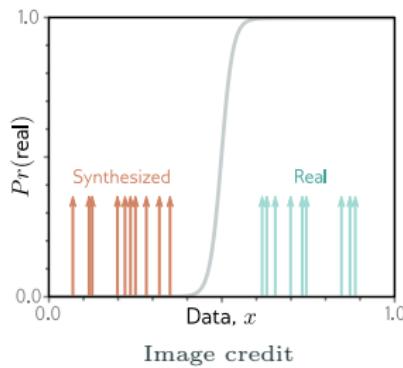
Проблемы обучения GAN

- Нестабильное обучение

Сходимость зависит от мелких деталей, высокая чувствительность к гиперпараметрам

- Vanishing gradients

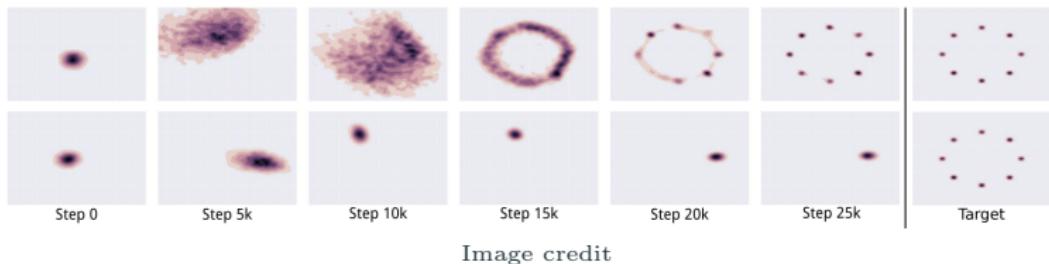
Если дискриминатор гораздо умнее генератора, то сигмоида на выходе дискриминатора насыщается



Проблемы обучения GAN

- Mode collapse

Генератор выдаёт реалистично выглядящие объекты, но они покрывают только небольшую часть распределения данных $p(x)$



Генератор обучается обманывать дискrimинатор, а не воспроизводить исходное распределение

Truncation

Сэмплирование шума из распределения с обрезанными хвостами

- Сэмплирование тех примеров из латентного пространства, которые генератор чаще всего видел при обучении
- Улучшение качества генерации за счет уменьшения разнообразия

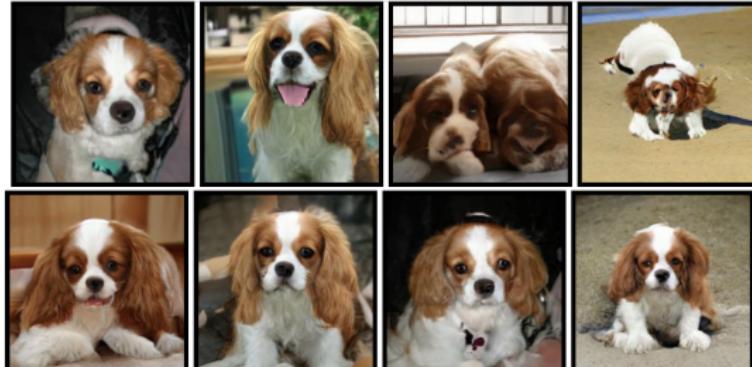
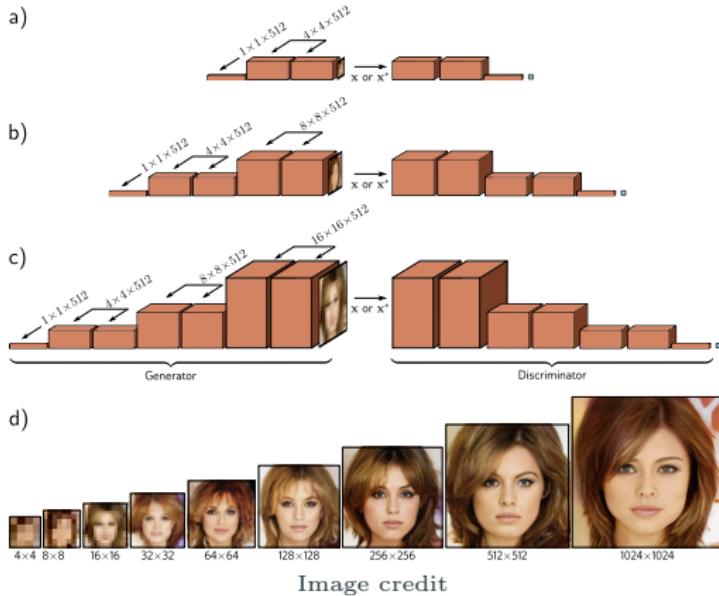


Image credit

Progressive growing

- Сперва обучаем первые слои генератора создавать изображение очень низкого разрешения
- Постепенно добавляем следующие слои и дообучаем под увеличенное разрешение



Измерение качества генерации

Frechet Inception Distance (FID)

- Хотим измерить похожесть распределений реальных данных и синтетических
- Идея - использовать признаки с последнего слоя предобученной модели (Inception)
- Считаем расстояние между этими признаками для синтетических и реальных данных, моделируя их многомерным распределением Гаусса

$$FID = |\mu_1 - \mu_2|^2 + \text{tr} \left(\Sigma_1 + \Sigma_2 - 2(\Sigma_1 \Sigma_2)^{1/2} \right)$$

μ_1, μ_2 - средние, Σ_1, Σ_2 - матрицы ковариации двух распределений.

Conditional GAN

Условная генерация - дополнительные данные y являются условием для генератора и дискриминатора.

$$\min_G \max_D E_x[\log D(x|y)] + E_z[\log(1 - D(G(z|y)))]$$

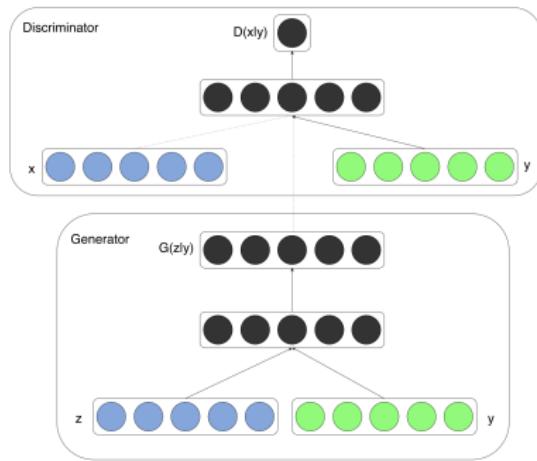


Image credit

y - например, метки классов

Image-to-image translation

Отображение изображения в изображение, например:

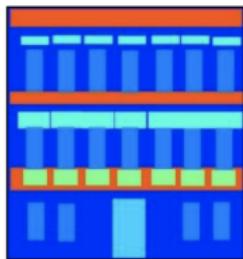
- Colorization
- Super-resolution
- Style Transfer
- Image inpainting

Pix2Pix

Isola P. et al. (2016) Image-to-image translation with conditional adversarial networks.

Conditional GAN with conditioning on input image

Labels to Facade

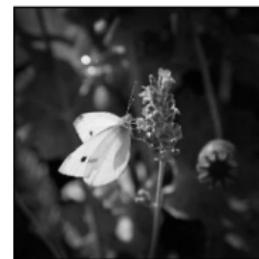


input



output

BW to Color



input



output

Day to Night



input



output

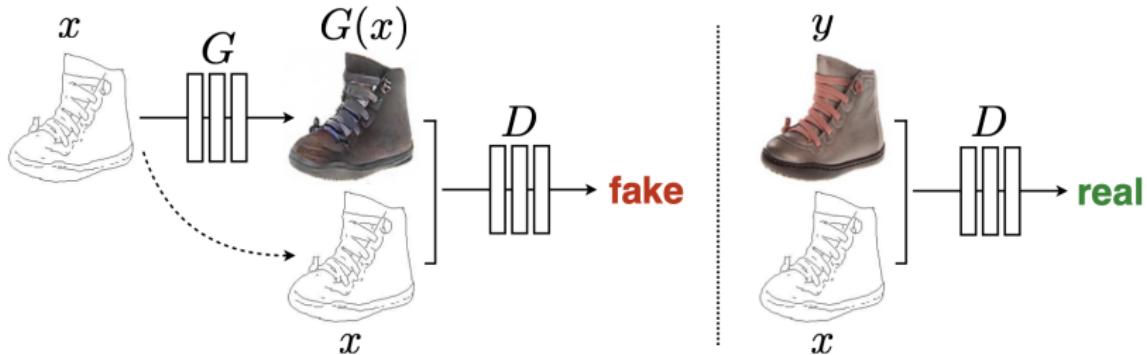
Edges to Photo



input



output



Генератор $G : \{x, z\} \rightarrow y$

x - input image, y - output image, z - random noise

Дискриминатор D отличает настоящие пары $\{x, y\}$ от сгенерированных $\{x, G(x)\}$.

Составная функция потерь: GAN loss + Reconstruction loss

$$L_{cGAN}(G, D) = E_{x,y}[\log D(x, y)] + E_{x,z}[\log(1 - D(x, G(x, z)))]$$

$$L_{L1}(G) = E_{x,y,z}[\|y - G(x, z)\|_1]$$

$$G^* = \arg \min_G \max_D [L_{cGAN}(G, D) + \lambda L_{L1}(G)]$$

Составная функция потерь: GAN loss + Reconstruction loss

$$L_{cGAN}(G, D) = E_{x,y}[\log D(x, y)] + E_{x,z}[\log(1 - D(x, G(x, z)))]$$

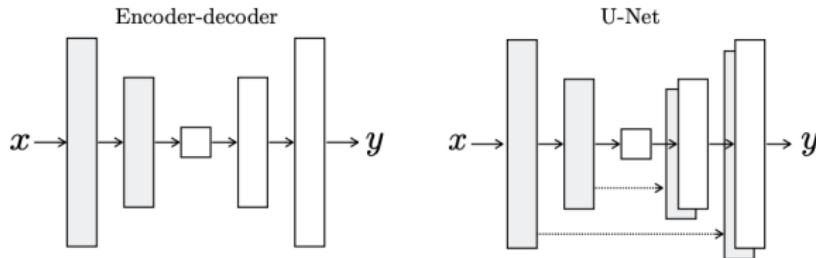
$$L_{L1}(G) = E_{x,y,z}[\|y - G(x, z)\|_1]$$

$$G^* = \arg \min_G \max_D [L_{cGAN}(G, D) + \lambda L_{L1}(G)]$$

Если обучать только с reconstruction loss, то получаются размытые изображения

- reconstruction loss отвечает за общую структуру
- GAN loss отвечает за мелкие детали

Generator



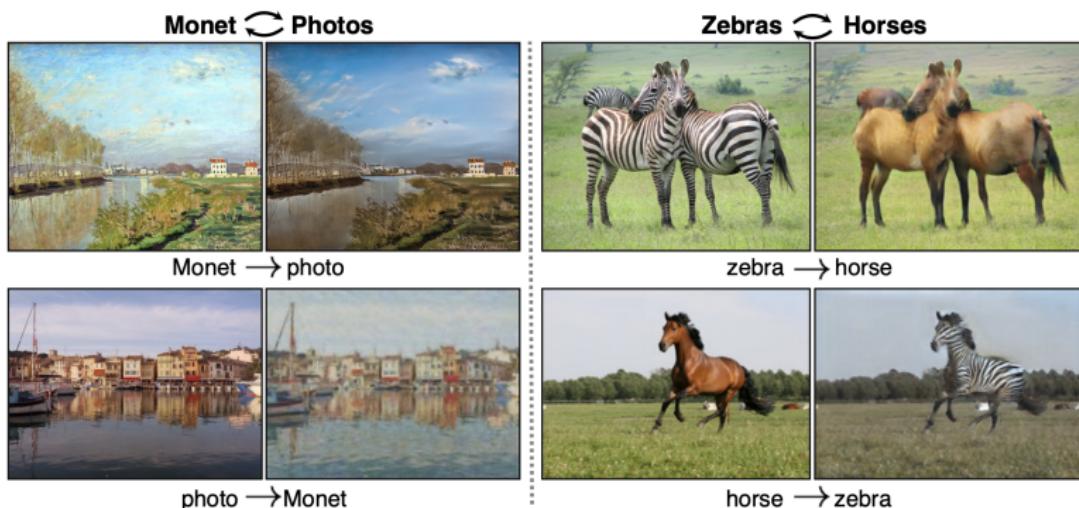
Discriminator - PatchGAN

- Классифицируем каждый патч размера $N \times N$ по отдельности
- Собираем патчи со всего изображения и усредняем
- Отвечает за реалистичность мелких деталей

CycleGAN

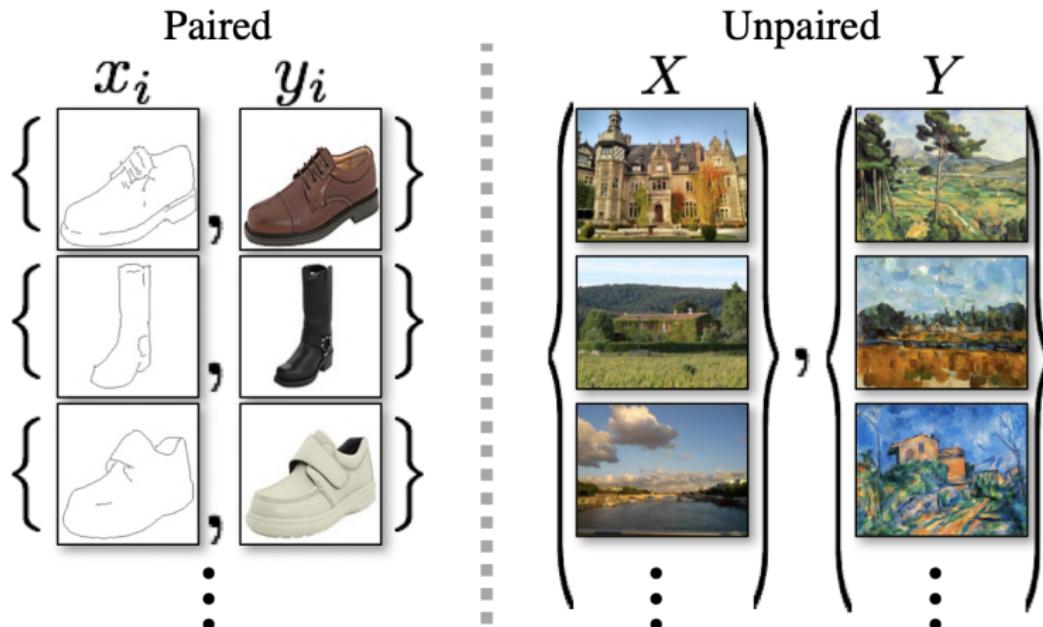
Zhu J. Y. et al. (2017) *Unpaired image-to-image translation using cycle-consistent adversarial networks.*

В отличие от pix2pix не нужны размеченные пары изображений!

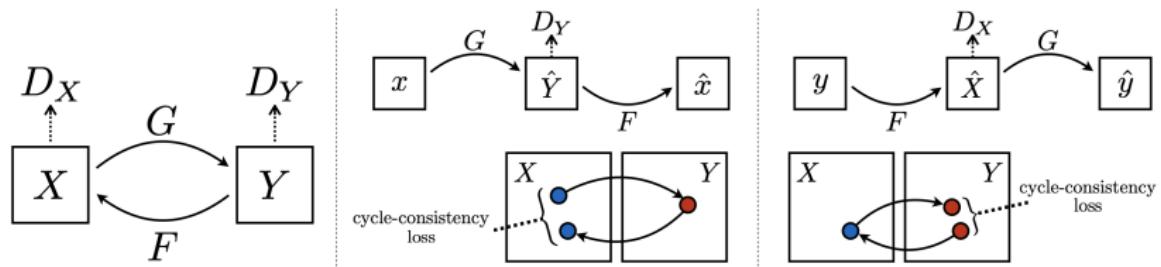


CycleGAN

Paired vs Unpaired data



CycleGAN



Два генератора - $G : x \rightarrow y$ и $F : y \rightarrow x$

Два дискриминатора - D_X и D_Y

Дополнительно требуем, чтобы $F(G(x)) \approx x$ и $G(F(y)) \approx y$

CycleGAN

Функция потерь

$$L_{GAN}(G, D_Y) = E_y[\log D_Y(y)] + E_x[\log(1 - D_Y G(x))]$$

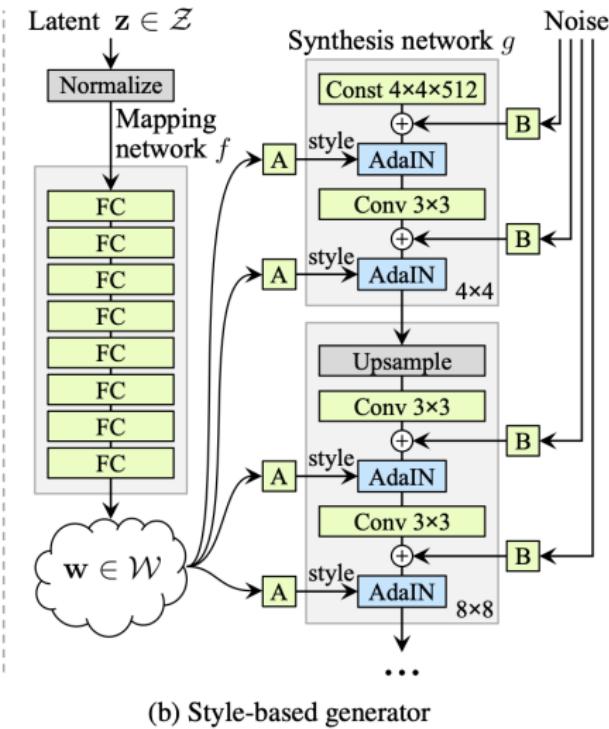
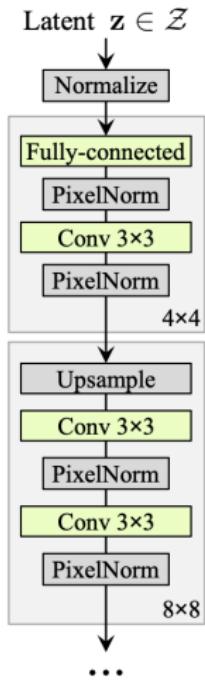
Cycle consistency loss

$$L_{cyc}(G, F) = E_x[\|F(G(x)) - x\|_1] + E_y[\|G(F(y)) - y\|_1]$$

$$G^* = \arg \min_{G,F} \max_{D_X,D_Y} [L_{GAN}(G, D_Y) + L_{GAN}(F, D_X) + \lambda L_{cyc}(G, F)]$$

StyleGAN

Karras T. et al. (2019) A style-based generator architecture for generative adversarial networks.



- Progressive growing
- Truncation trick
- Подача вектора шума не в начале генератора, а много раз на промежуточных слоях через слой AdaIN - Adaptive Instance Normalization
- Латентные векторы пропускаются через многослойный перцептрон перед подачей в генератор, чтобы получить «распутанные» представления

Long (short) way of GAN



В следующий раз

- Variational Autoencoders
- Diffusion models
- Text-to-image (DALL-E, DALL-E 2)