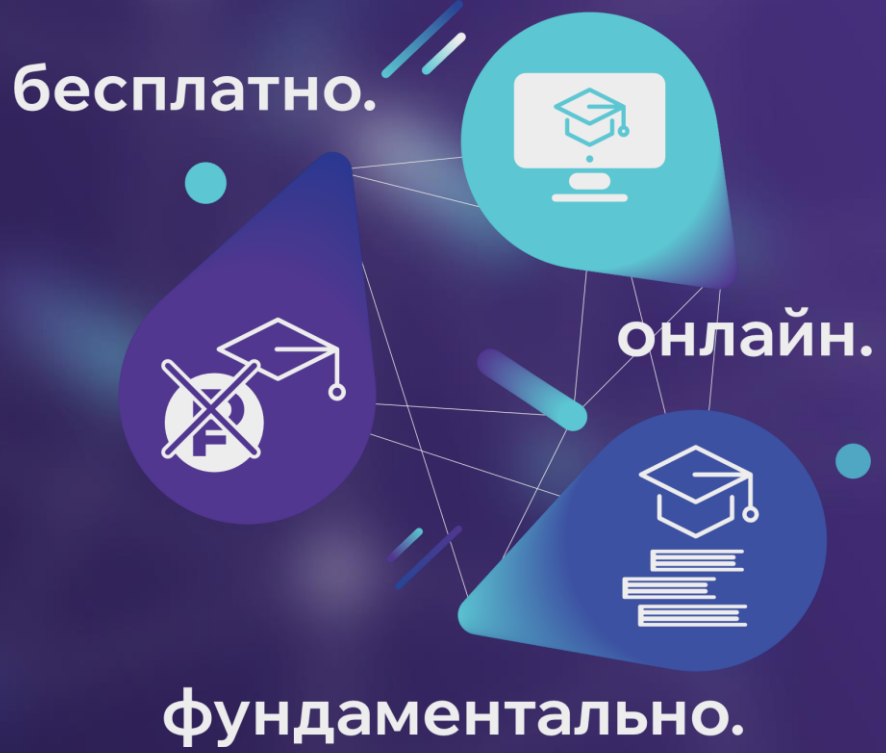
Deep Learning School



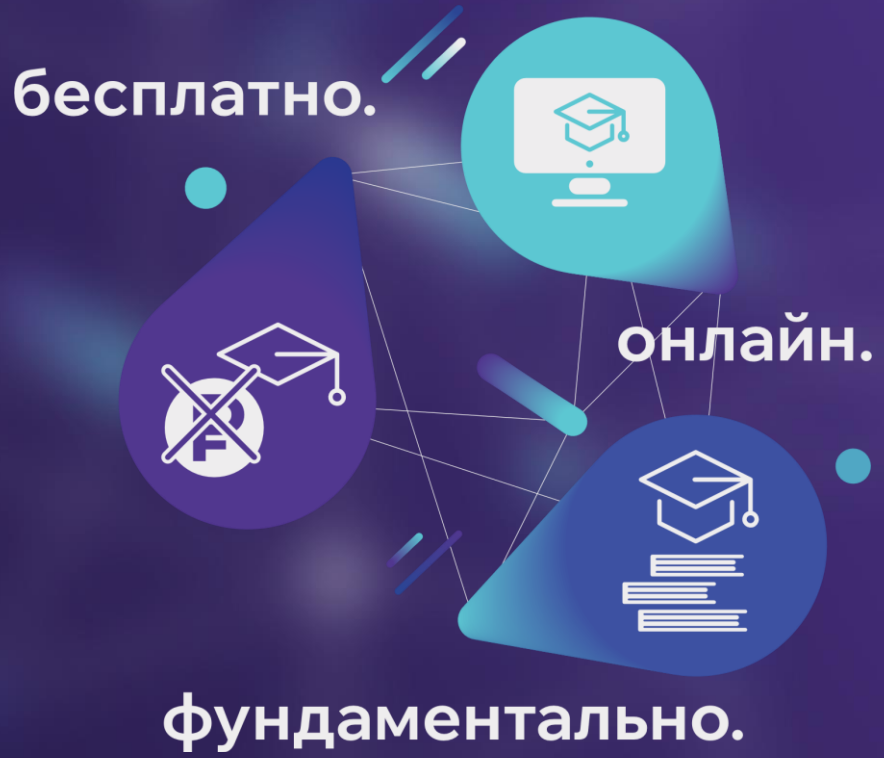
2024


Deep
Learning
School
онлайн-школа по искусственному интеллекту в России


Deep
Learning
School


DLS

Deep Learning School

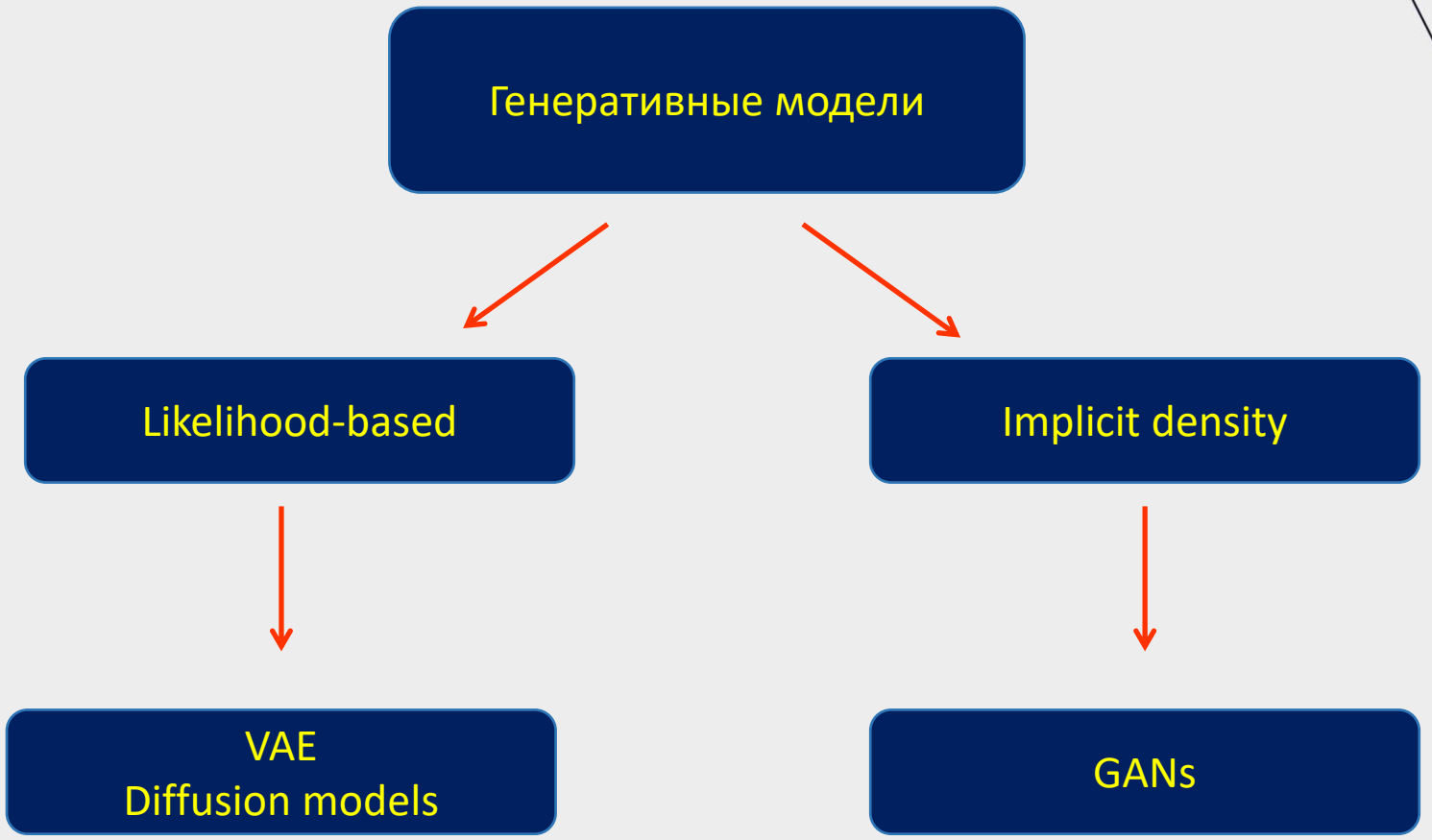


DLS

Коновалова Нина Петровна

Generative Adversarial Networks

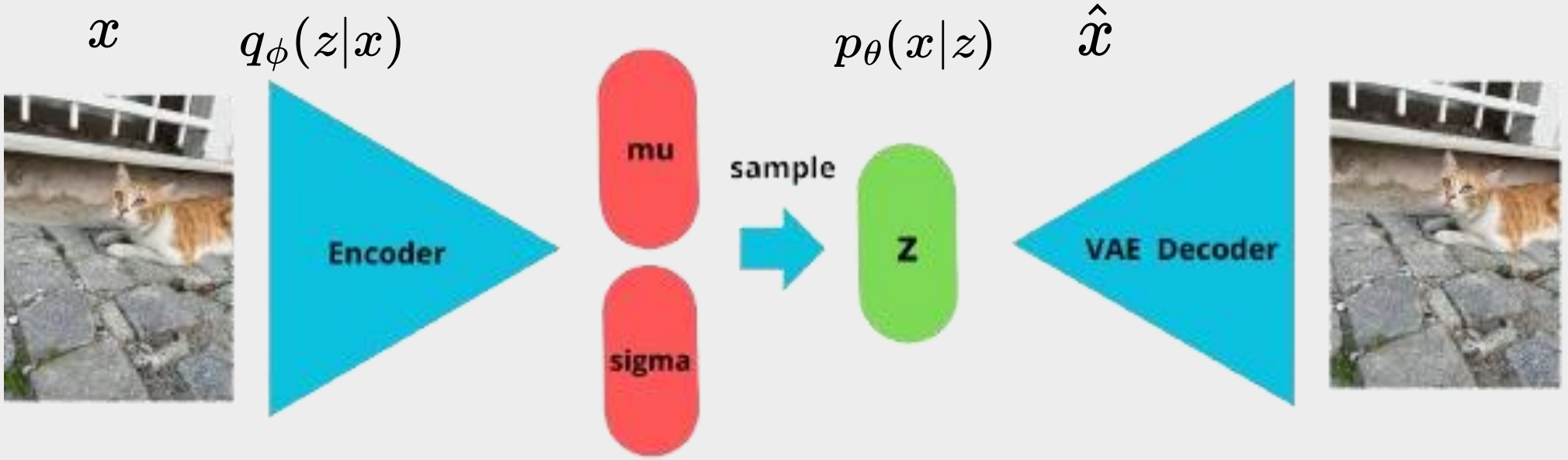
Генеративные модели



VAE

$$\mathcal{L}_{elbo} = \sum_{i=1}^{n} \int q(z_i|x_i,\phi) \log p(x_i|z_i,\theta) dz_i - \sum_{i=1}^{n} D_{KL}(q(z_i|x_i,\phi)|p(z_i))$$

Reconstruction **Regularization**



Распределение

Генерация VAE

Реконструкция

Сэмплирование

Замыленные результаты!

Генерация VAE

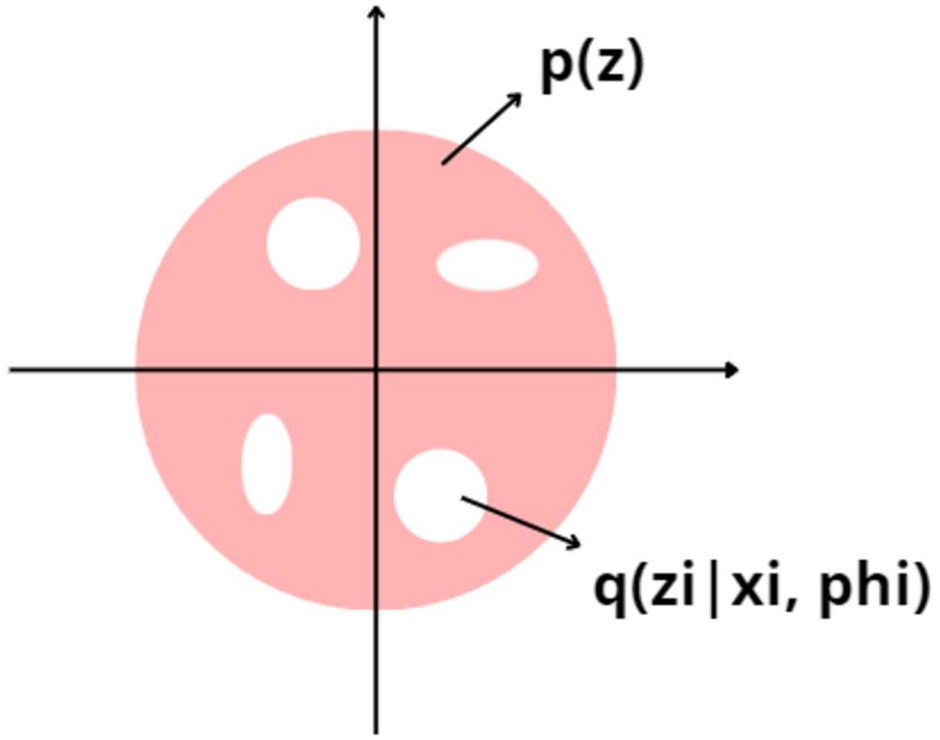
Замыленные результаты!

Проблемы VAE (аппроксимация латентного пространства)

Латентное пространство не полностью учится

↓

проблемы с сэмплированием!



$$\mathcal{L}_{elbo} = \sum_{i=1}^{n} \int q(z_i|x_i,\phi) \log p(x_i|z_i,\theta) dz_i - \sum_{i=1}^{n} D_{KL}(q(z_i|x_i,\phi)|p(z_i))$$

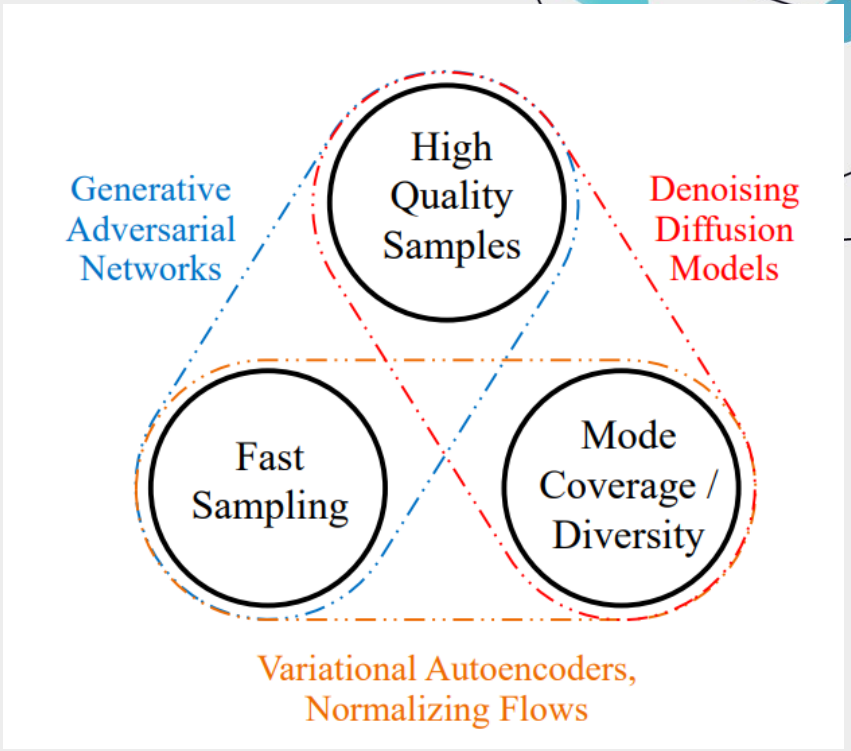
Reconstruction **Regularization**

Генеративная трилемма

Variational Autoencoders

Pros:
- Вариативность генерации
- Высокая скорость

Cons:
- «Замыленные» изображение, низкое качество

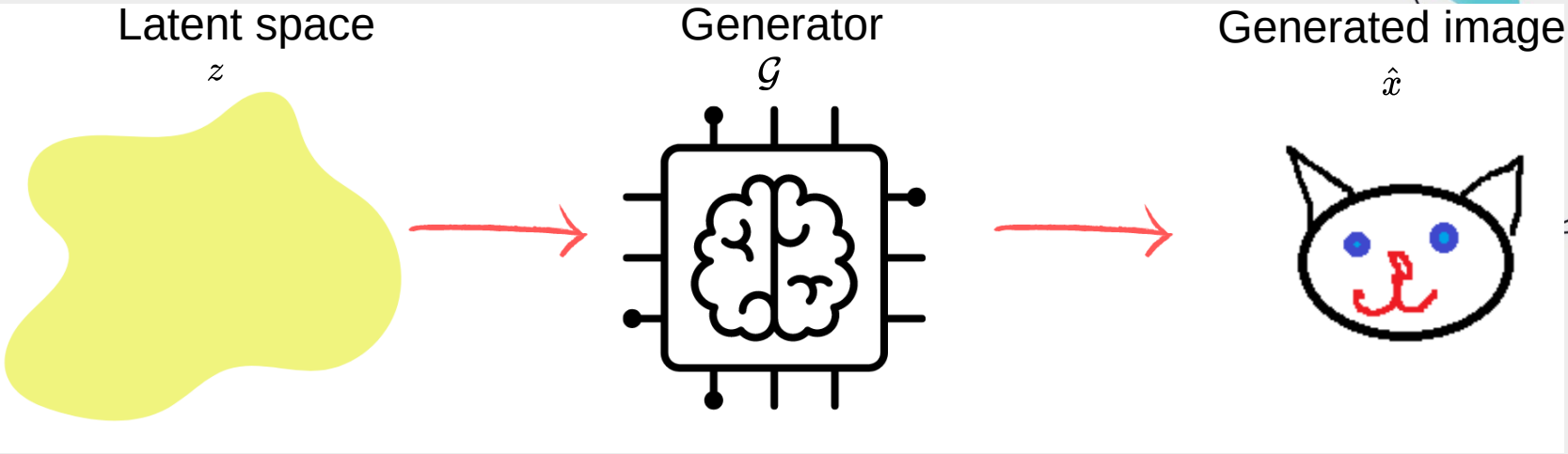


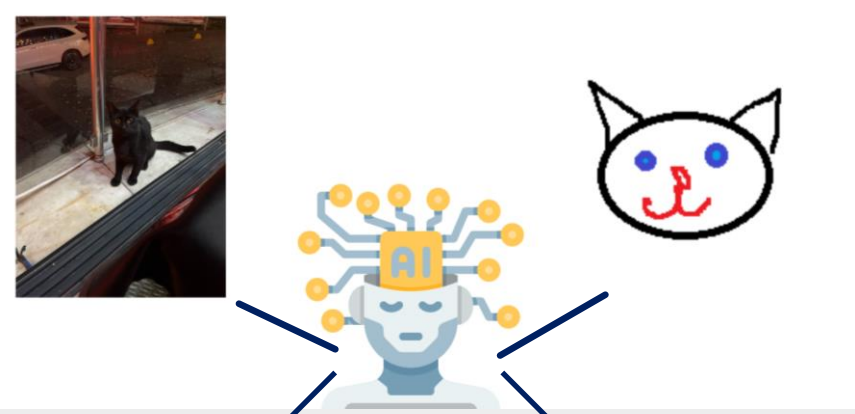
[1] DENOISING DIFFUSION GANS

Generative Adversarial Networks

Изображение сгенерировано с помощью диффузионной модели SDXL Turbo, промт **«two players play minmax game»**

Generative Adversarial Networks





Реальная картинка

Сгенерированная картинка

Generative Adversarial Networks

Генератор: сгенерировать изображение, такое что его нельзя отличить от реального

Дискриминатор: отличить сгенерированное изображение от реального

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x \sim X}[\log \mathcal{D}(x)] + \mathbb{E}_{z \sim Z}[\log(1 - \mathcal{D}(\mathcal{G}(z))]$$

P(«хорошая картинка»)

1 - P(«хорошая картинка»)

Максимизация: параметры дискриминатора

Минимизация: параметры генератора

Generative Adversarial Networks (псевдокод)

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

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

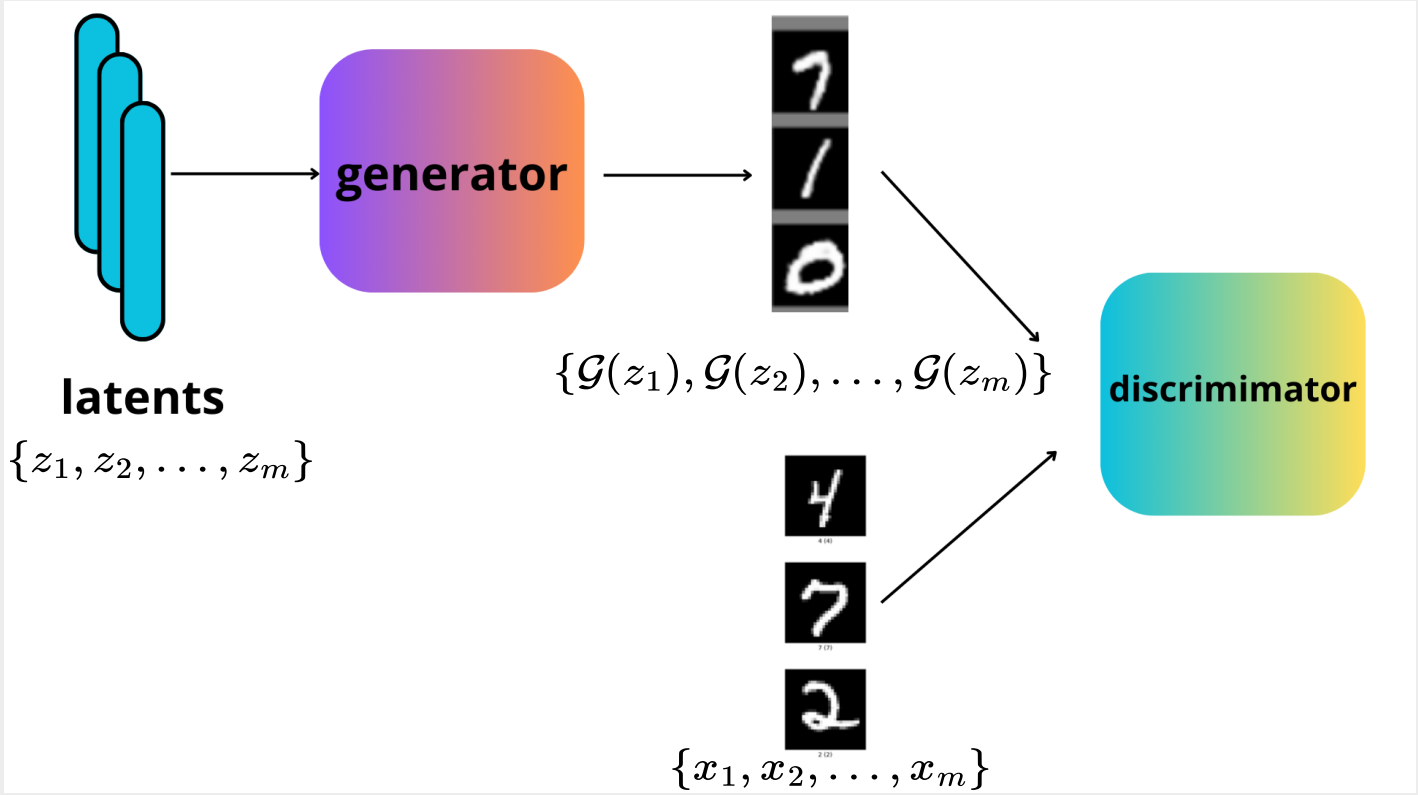
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

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

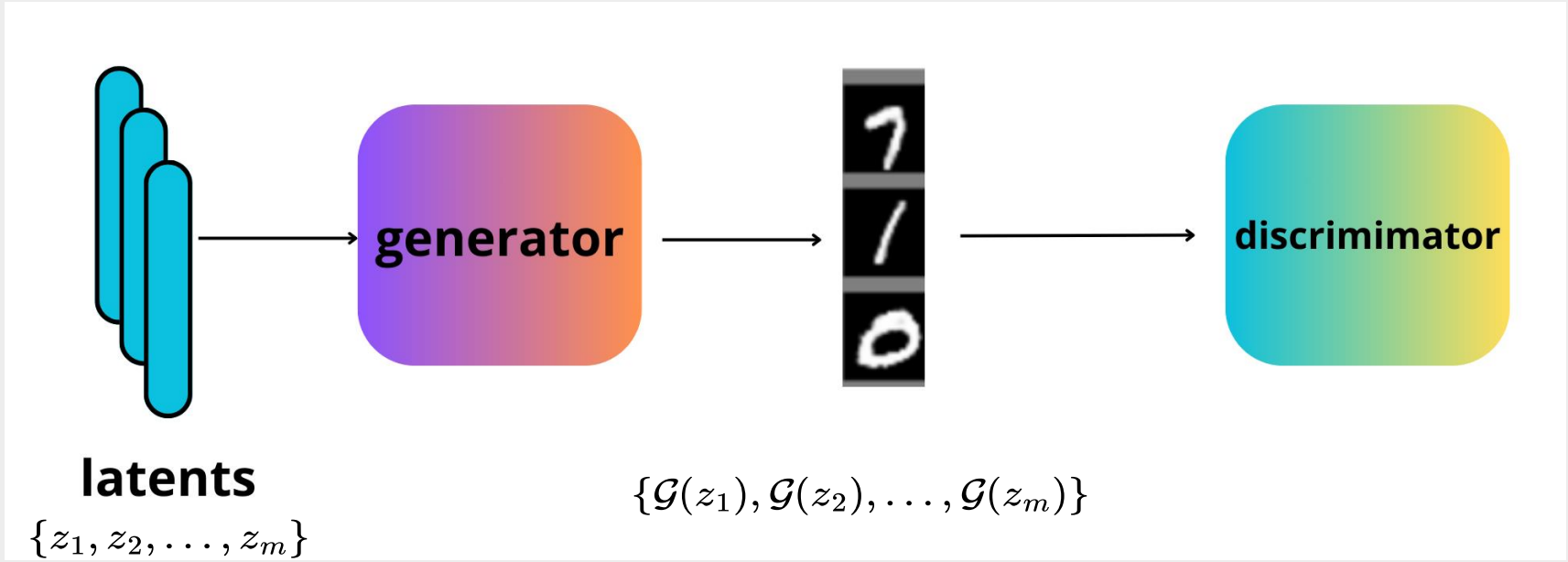
[2] Generative adversarial nets

Generative Adversarial Networks



$$\nabla_{\theta_\mathcal{D}} \frac{1}{m} \sum_{m} [\log \mathcal{D}(x_i) + \log(1 - \mathcal{D}(\mathcal{G}(z_i))]$$

Generative Adversarial Networks



$$\nabla_{\theta_\mathcal{G}} \frac{1}{m} \sum_{m} \log(1 - \mathcal{D}(\mathcal{G}(z_i))$$

Идеальный дискриминатор

Зафиксируем генератор и будем обучать только дискриминатор

$$\max_{\mathcal{D}} V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x \sim X}[\log \mathcal{D}(x)] + \mathbb{E}_{z \sim Z}[\log(1 - \mathcal{D}(\mathcal{G}(z))]$$

$$\frac{\delta V(\mathcal{D}, \mathcal{G})}{\delta \mathcal{D}(x)} = \frac{\delta[p_{data}(x) \log \mathcal{D}(x) + p_{gen}(x) \log\left(1 - \mathcal{D}(\mathcal{G}(z))\right)]}{\delta \mathcal{D}(x)} = \frac{p_{data}(x)}{\mathcal{D}(x)} - \frac{p_{gen}(x)}{1 - \mathcal{D}(x)} = 0$$

$$(1 - \mathcal{D}(x))p_{data}(x) - \mathcal{D}(x)p_{gen}(x) = 0$$

$$\mathcal{D}^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_{gen}(x)}$$

Дискриминатор просто стремится оценить отношение вероятности реальных к сумме вероятностей реальных и сгенерированных данных

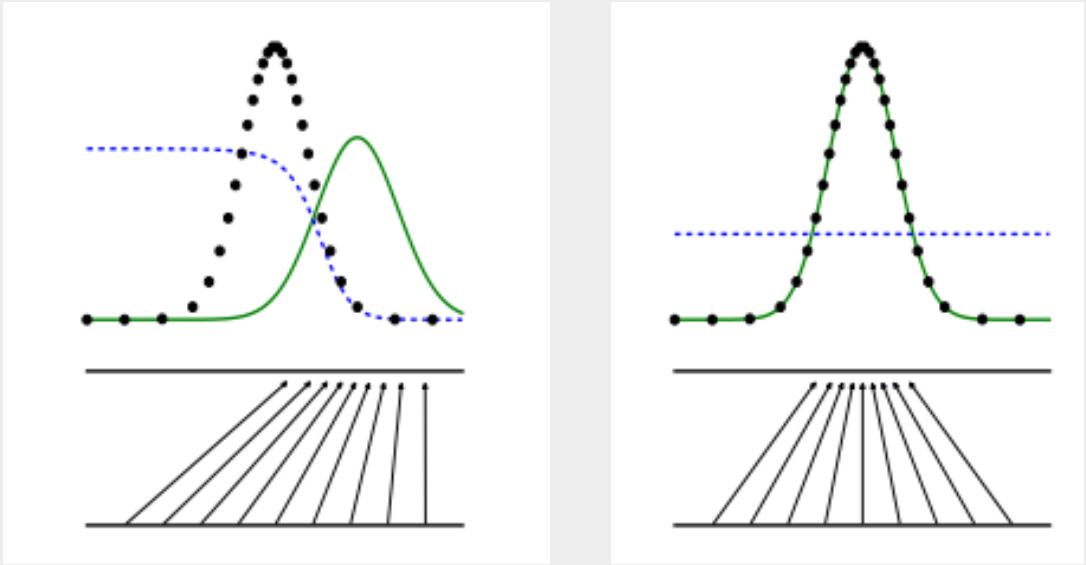
Идеальный дискриминатор

Зафиксируем генератор и будем обучать только дискриминатор

Дискриминатор просто стремится оценить отношение вероятности реальных к сумме вероятностей реальных и сгенерированных данных

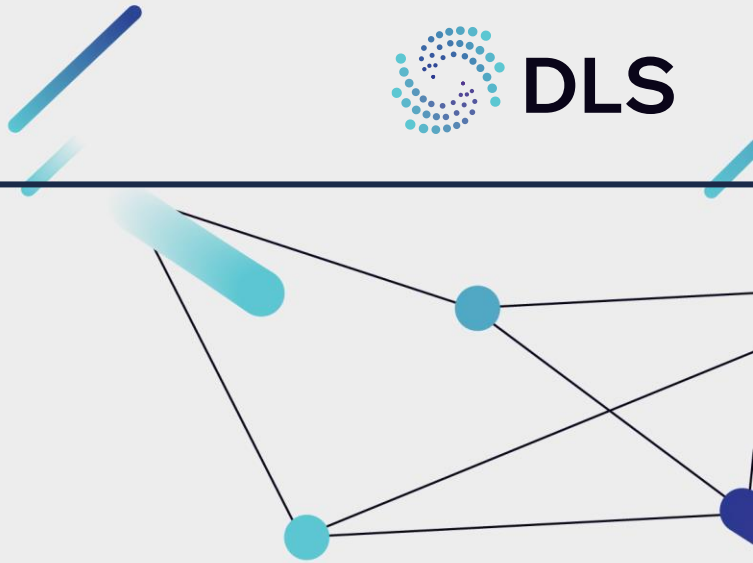
$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x \sim X}[\log \mathcal{D}(x)] + \mathbb{E}_{z \sim Z}[\log - \mathcal{D}(\mathcal{G}(z))]$$

$$\mathcal{D}^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_{gen}(x)}$$



[2] Generative adversarial nets

Идеальный генератор



Зафиксируем дискриминатор и будем обучать только генератор

$$\min_{\mathcal{G}} V(\mathcal{G}) = \mathbb{E}_{x \sim X}[\log \mathcal{D}(x)] + \mathbb{E}_{z \sim Z}[\log(1 - \mathcal{D}(\mathcal{G}(z))]$$

$$\max_{\mathcal{G}} V(\mathcal{G}) = \mathbb{E}_{z \sim Z}[\log(\mathcal{D}(\mathcal{G}(z))]$$

$$\mathcal{G}^*(z) = \arg\max_{x} \mathcal{D}(x)$$

Генератор сойдётся просто в моду дискриминатора и мы получим так называемый **«mode collapse»**

Mode collapse примеры

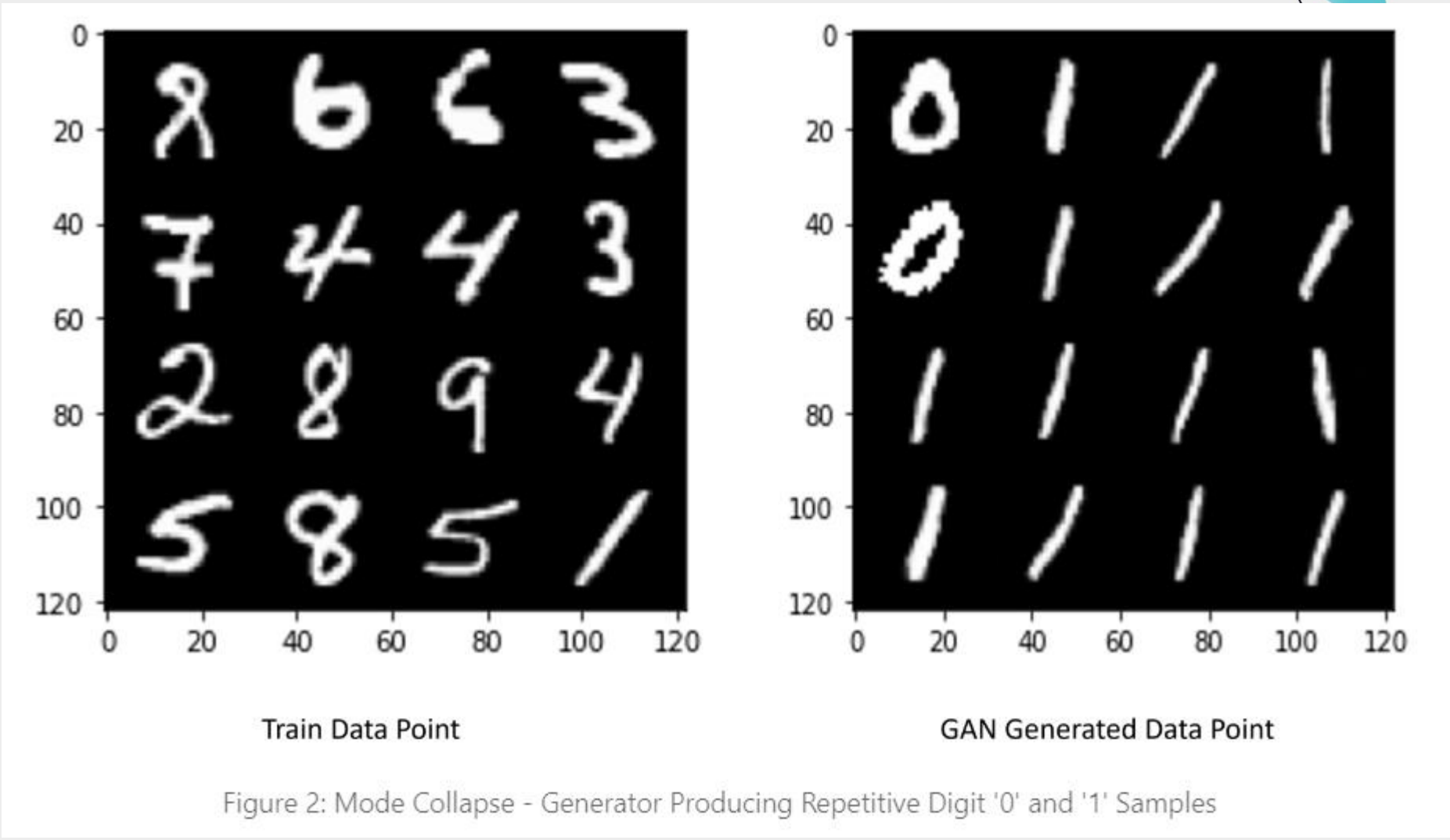


Figure 2: Mode Collapse - Generator Producing Repetitive Digit '0' and '1' Samples

[3] https://www.linkedin.com/pulse/addressing-mode-collapse-generative-adversarial-gans-andrew-jaramillo/

Mode collapse примеры

Figure 10: More face generations from our Face DCGAN.

[4] DCGAN

GAN генерации (проблемы)

Figure 10: More face generations from our Face DCGAN.

[4] DCGAN

1. Есть «странные генерации»

2. Появляются «близнецы» (mode collapse)

3. Теряются некоторые части распределения (например, в обучающей выборке есть люди с очками, а тут ни одного человека).

Проблемы

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x \sim X}[\log \mathcal{D}(x)] + \mathbb{E}_{z \sim Z}[\log(1 - \mathcal{D}(\mathcal{G}(z))]$$

P(«хорошая картинка»)

1 - P(«хорошая картинка»)

Vanishing gradients: если дискриминатор гораздо «умнее» генератора, то у генератора возникает проблема затухающего градиента, так как дискриминатор может легко отличить сгенерированные объекты от объектов обучающей выборки

Проблемы

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x \sim X}[\log \mathcal{D}(x)] + \mathbb{E}_{z \sim Z}[\log(1 - \mathcal{D}(\mathcal{G}(z))]$$

P(«хорошая картинка»)

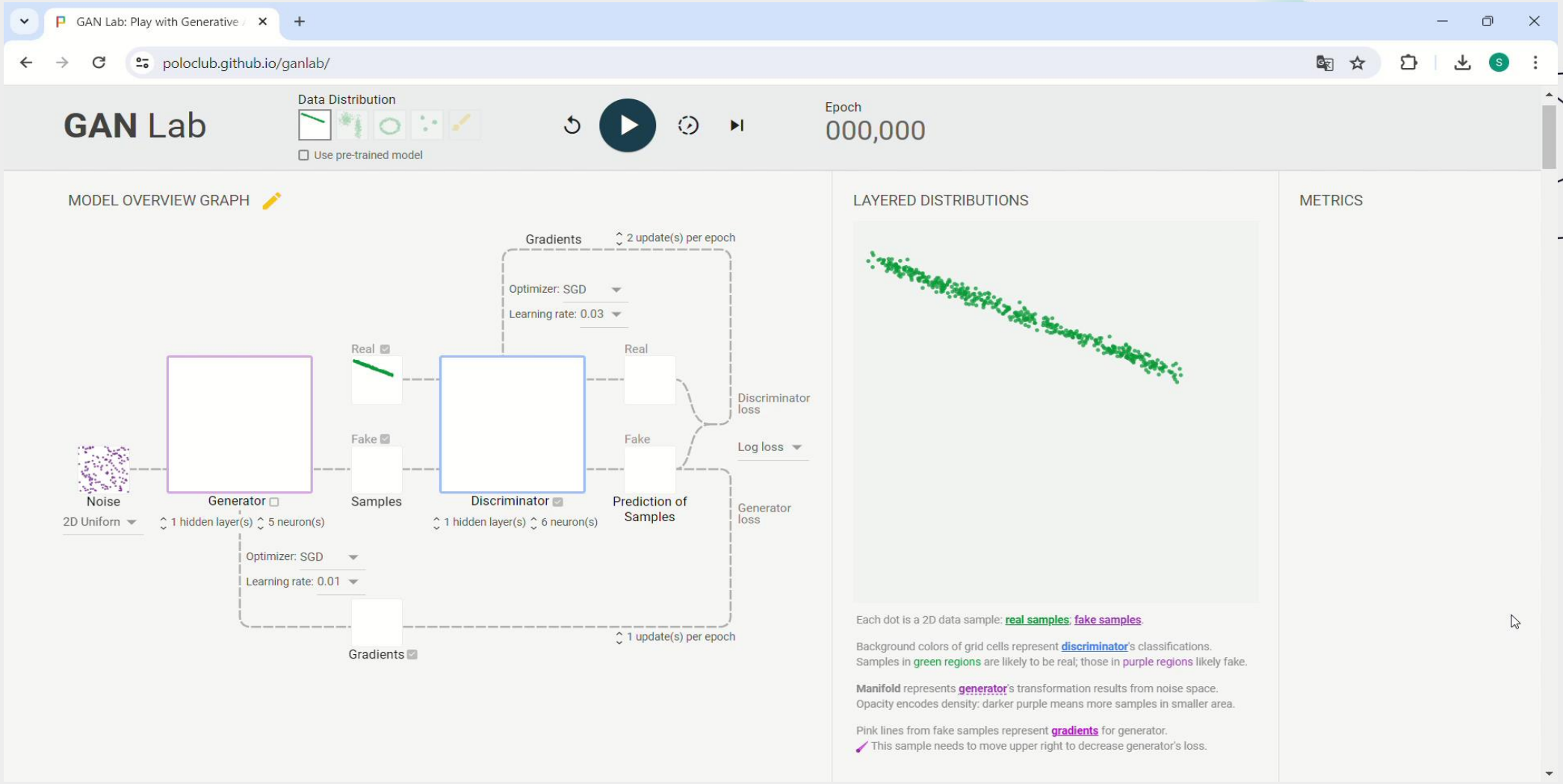
1 - P(«хорошая картинка»)

Решение:

$$1. \max_{\mathcal{D}} V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x \sim X}[\log \mathcal{D}(x)] + \mathbb{E}_{z \sim Z}[\log(1 - \mathcal{D}(\mathcal{G}(z))]$$

$$2. \max_{\mathcal{G}} V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{z \sim Z}[\log(\mathcal{D}(\mathcal{G}(z))]$$

Generative Adversarial Networks



https://poloclub.github.io/ganlab/

Generative Adversarial Networks, результаты

Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and "deconvolutional" generator)

[2] Generative adversarial nets

Какие ещё проблемы?

Распределение фейков и действительных данных не пересекается

↓

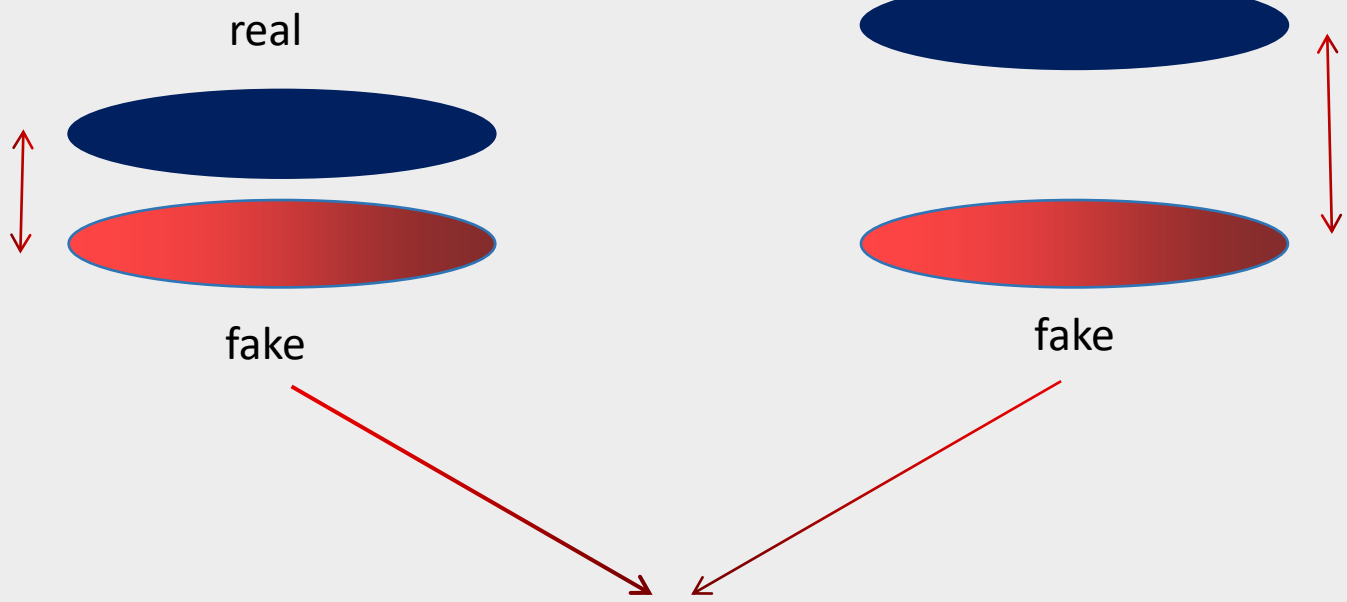
Дискриминатор умеет точно разделять фейки и реальные картинки

↓

Градиент почти нулевой

↓

Хотим: уметь различать близко расположенные и далеко расположенные распределения



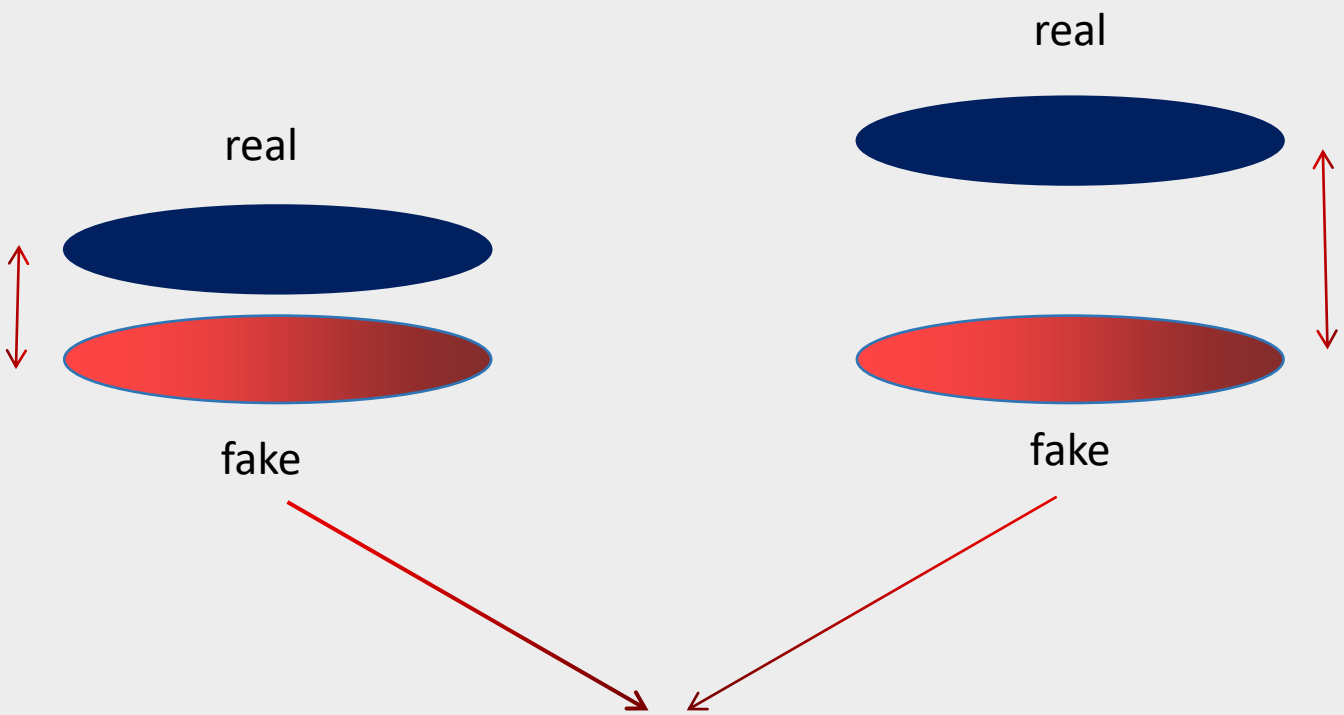
Дискриминатор не различает две ситуации

Earth moved distance

$$\min_{\mathcal{G}} W(X \| \mathcal{G}(Z))$$

$$W(X\|Y) = \inf_{\gamma \in \Pi(X,Y)} \mathbb{E}_{(x,y)\sim\gamma}[\|x-y\|]$$

Earth moved distance



$$\min_{\mathcal{G}} W(X \| \mathcal{G}(Z))$$

$$W(X\|Y) = \inf_{\gamma \in \Pi(X,Y)} \mathbb{E}_{(x,y)\sim\gamma}[\|x-y\|]$$

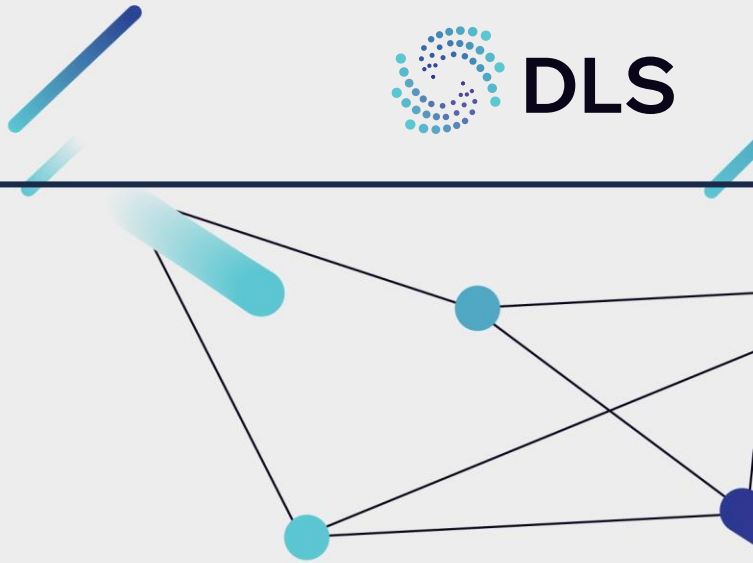
Wasserstein GAN

$$W(X\|Y) = \inf_{\gamma \in \Pi(X,Y)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

Kantorovich-Rubinstein duality

$$W(X\|Y) = \sup_{\|f\|_L \leq 1} (\mathbb{E}_{x\sim X}[f(x)] - \mathbb{E}_{x\sim Y}[f(x)])$$

Ограниченность по Липшицу



Функция f(x) считается ограниченной по Липшицу, если существует константа L, называемая константой Липшица, такая что для любых двух точек x1 и x2 из области определения функции выполнено неравенство:

$$f(x_1) - f(x_2) \leq L|x_1 - x_2|$$

Wasserstein GAN

Kantorovich-Rubinstein duality

$$W(X\|Y) = \sup_{\|f\|_L \leq 1} (\mathbb{E}_{x\sim X}[f(x)] - \mathbb{E}_{x\sim Y}[f(x)])$$

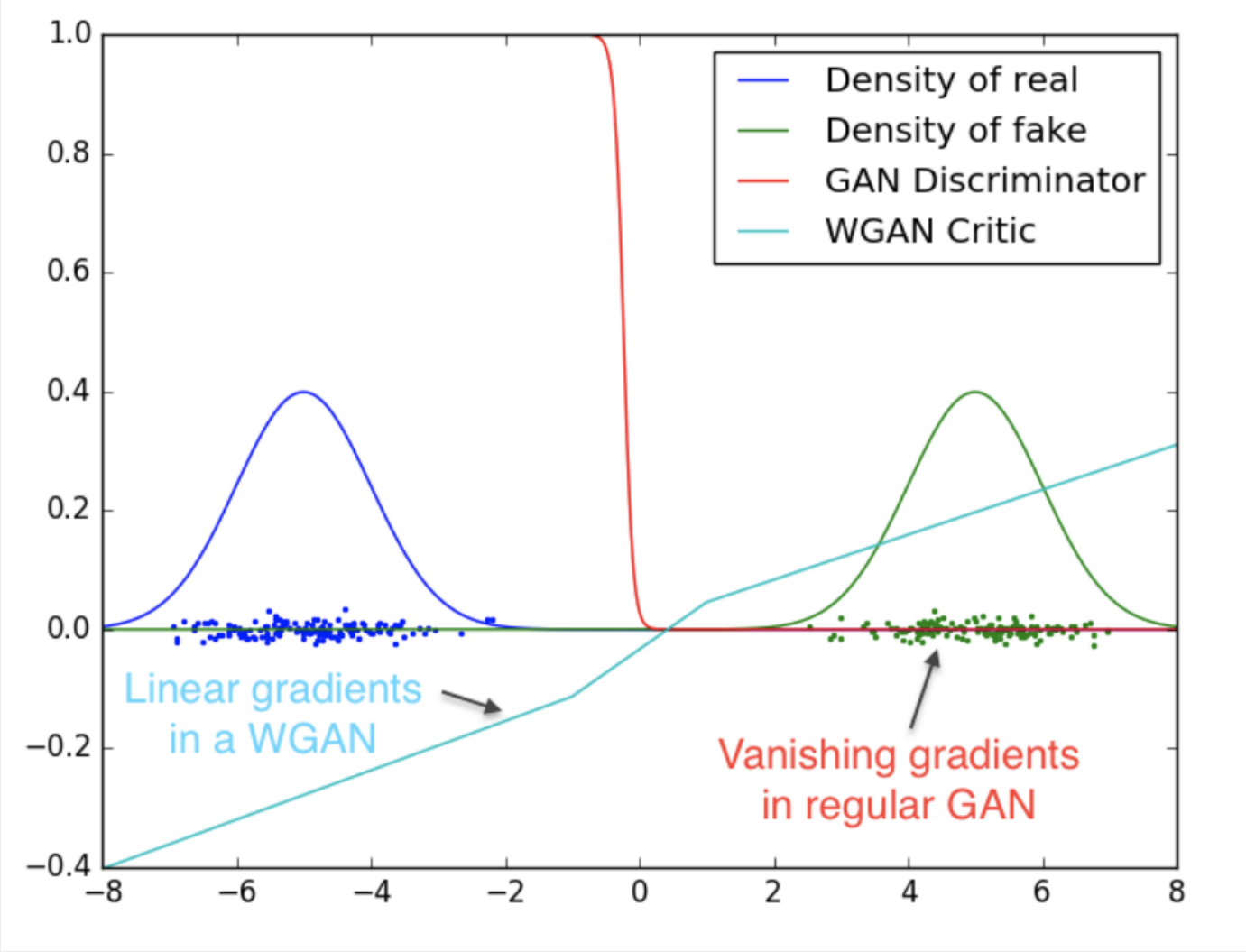
Wasserstein GAN

$$\min_G \max_{\|f\|_L \leq 1} (\mathbb{E}_{x\sim X}[f(x)] - \mathbb{E}_{z\sim Z}[f(g(z))])$$

Usual GAN

$$\min_G \max_D (\mathbb{E}_{x\sim X}[\log d(x)] - \mathbb{E}_{z\sim Z}[1 - d(g(z))])$$

Wasserstein GAN



[5] Wasserstein generative adversarial networks

Wasserstein GAN

$$\min_G \max_{\|f\|_L \leq 1} (\mathbb{E}_{x \sim X}[f(x)] - \mathbb{E}_{z \sim Z}[f(g(z))])$$

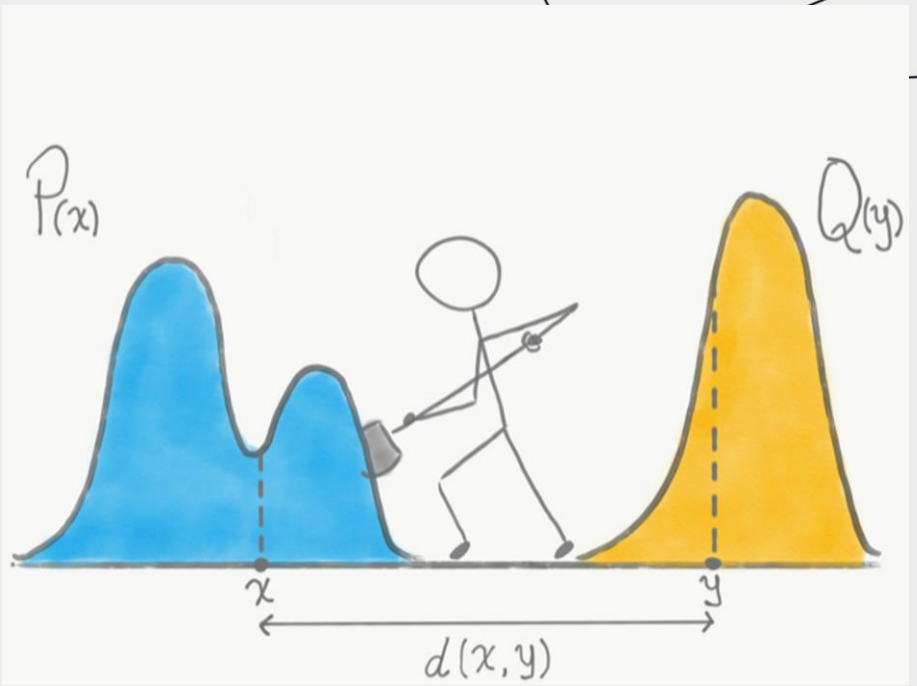
Usual GAN

$$\min_G \max_D (\mathbb{E}_{x \sim X}[\log d(x)] - \mathbb{E}_{z \sim Z}[1 - d(g(z))])$$

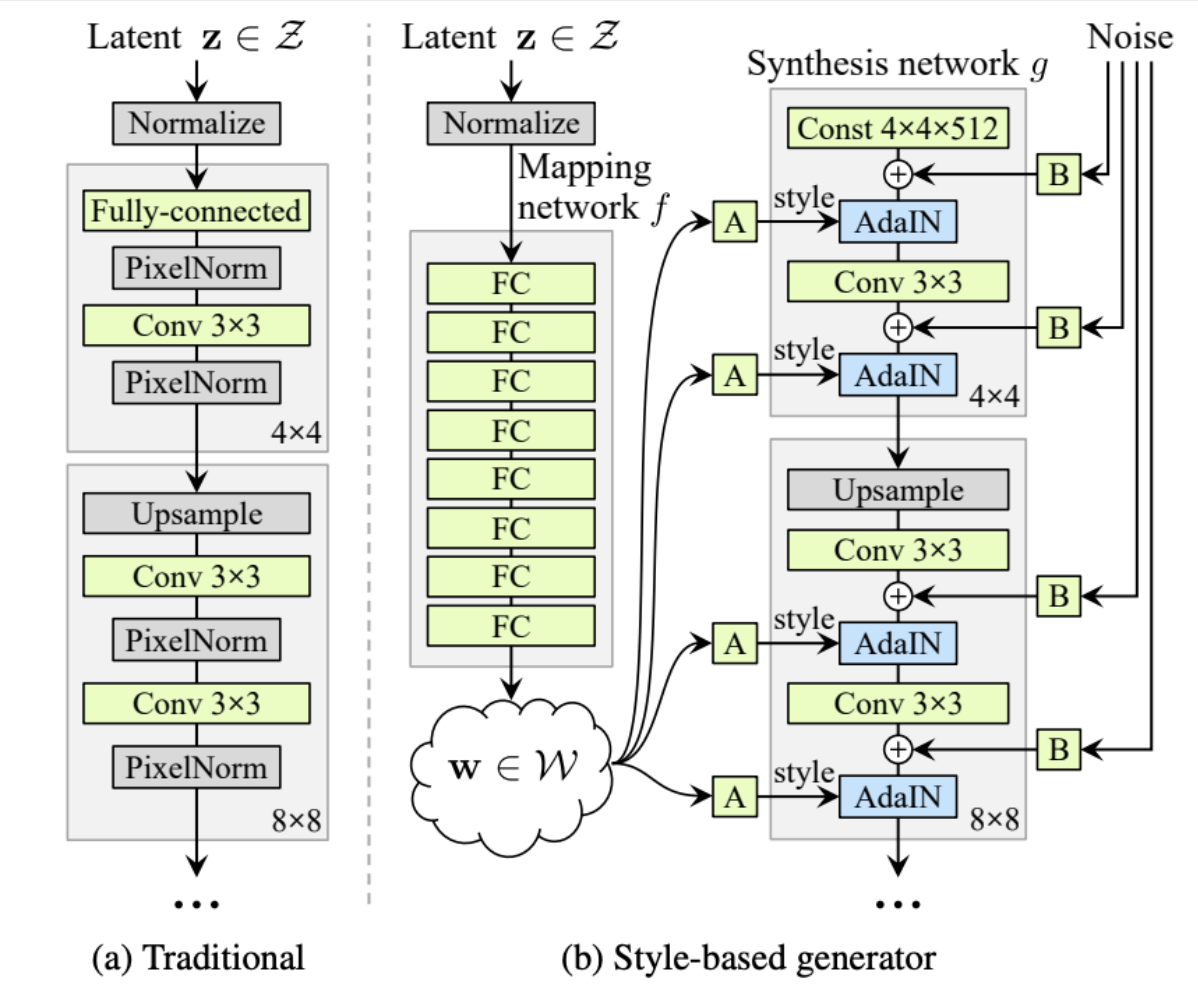
Wasserstein GAN

Основные изменения по сравнению с классическим GANs

- После каждого обновления градиента на функции критика ограничивают веса в небольшом фиксированном диапазоне.
- Используется новая функция потерь, производная от расстояния Вассерштейна, без логарифма.
- Модель "дискриминатора" больше не выступает в роли прямого критика, а служит вспомогательным инструментом для оценки метрики Вассерштейна между реальным и сгенерированным распределениями данных.



StyleGAN



[6]StyleGAN

Основные идеи:

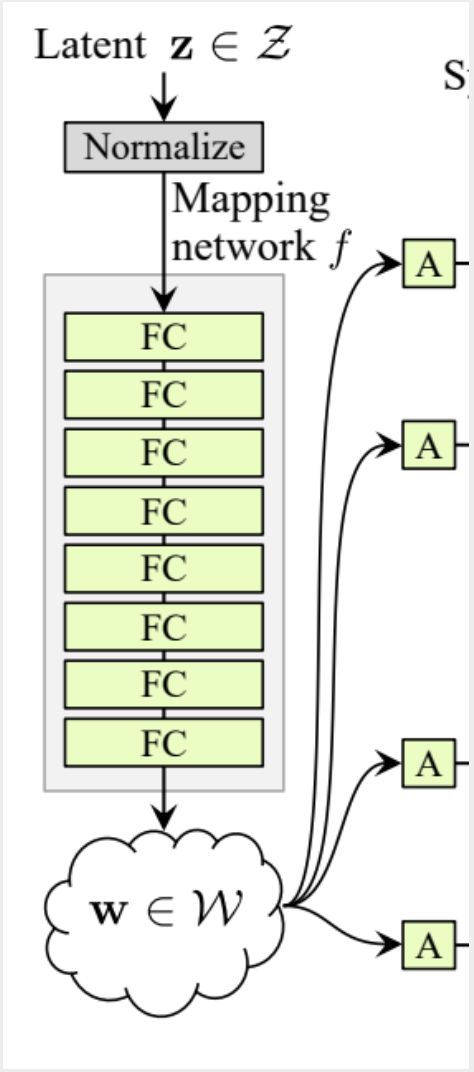
1) Mapping network

2) AdaIN layers

3) Добавление шума после каждой свёртки

4) Добавление обучаемого скейл фактора для шума

StyleGAN - mapping network



[6]StyleGAN

латентный вектор z

8-layers MLP

Affine transform - style vectors

AdaIN

Возможные нормализации:

Batch normalization

$$\mathrm{BN}(x) = \gamma\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \beta$$

$$\mu(x) = \frac{1}{HWN}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nwh}$$

$$\sigma(x) = \frac{1}{HWN}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{nwh} - \mu(x))^2$$

Instance normalization

$$\mathrm{IN}(x) = \gamma\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \beta$$

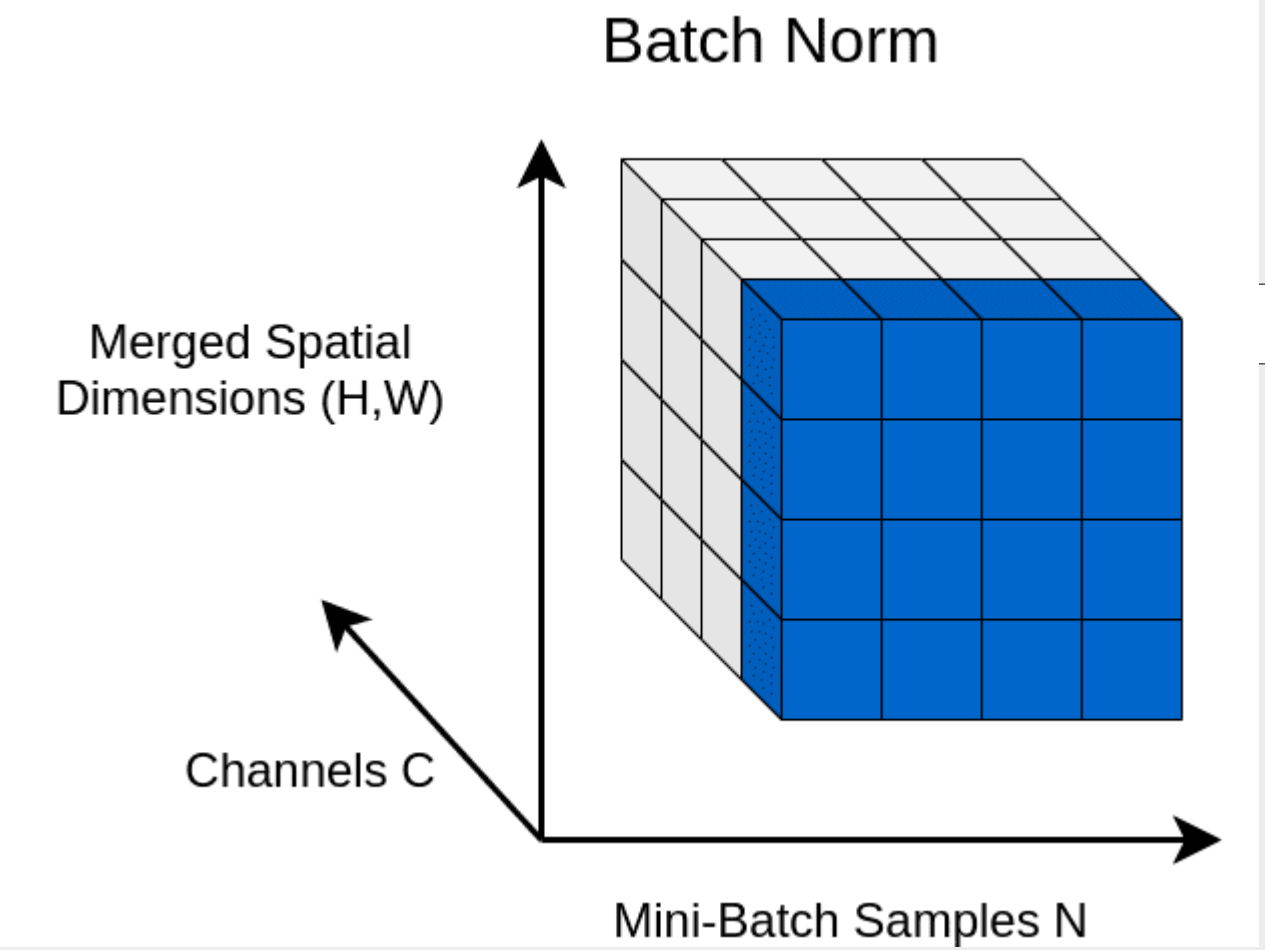
$$\mu(x) = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{wh}$$

$$\sigma(x) = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{wh} - \mu(x))^2$$

Adaptive Instance normalization

$$\mathrm{AdaIN}(x) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)$$

Batch normalization



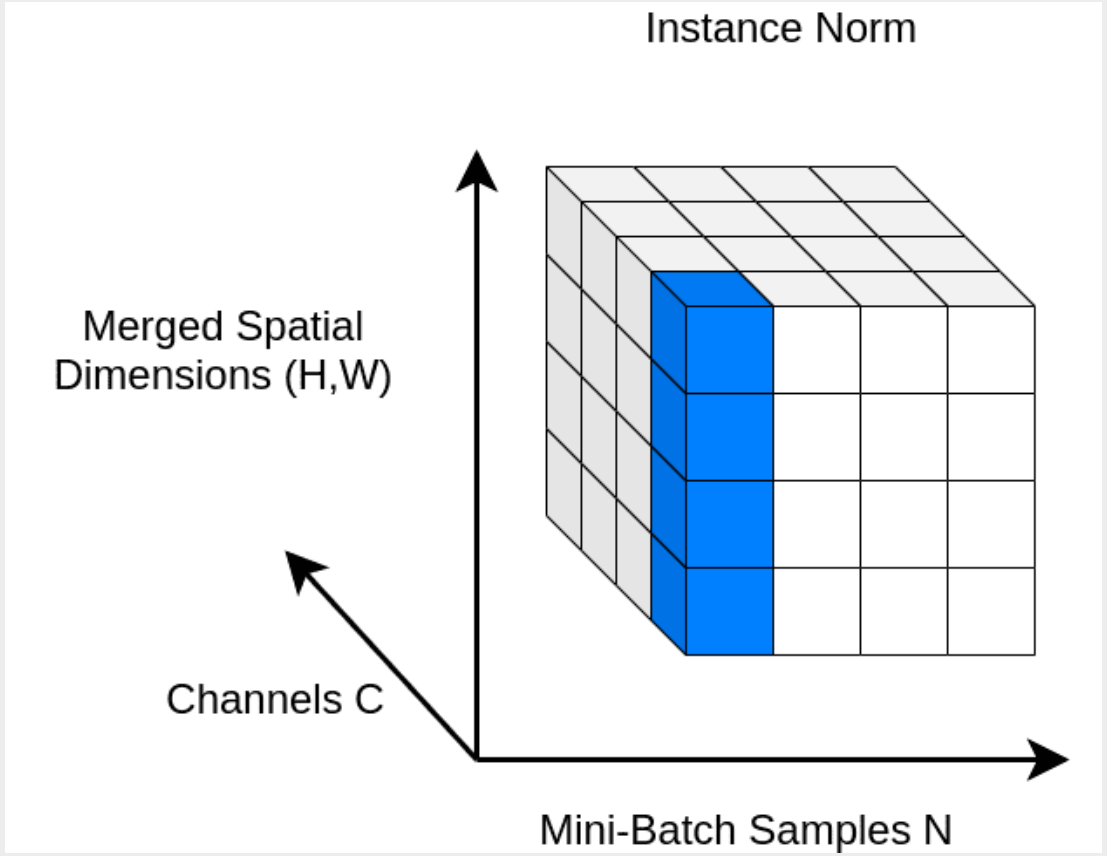
Batch normalization

$$\mathrm{BN}(x) = \gamma\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \beta$$

$$\mu(x) = \frac{1}{HWN}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nwh}$$

$$\sigma(x) = \frac{1}{HWN}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} (x_{nwh} - \mu(x))^2$$

Instance normalization



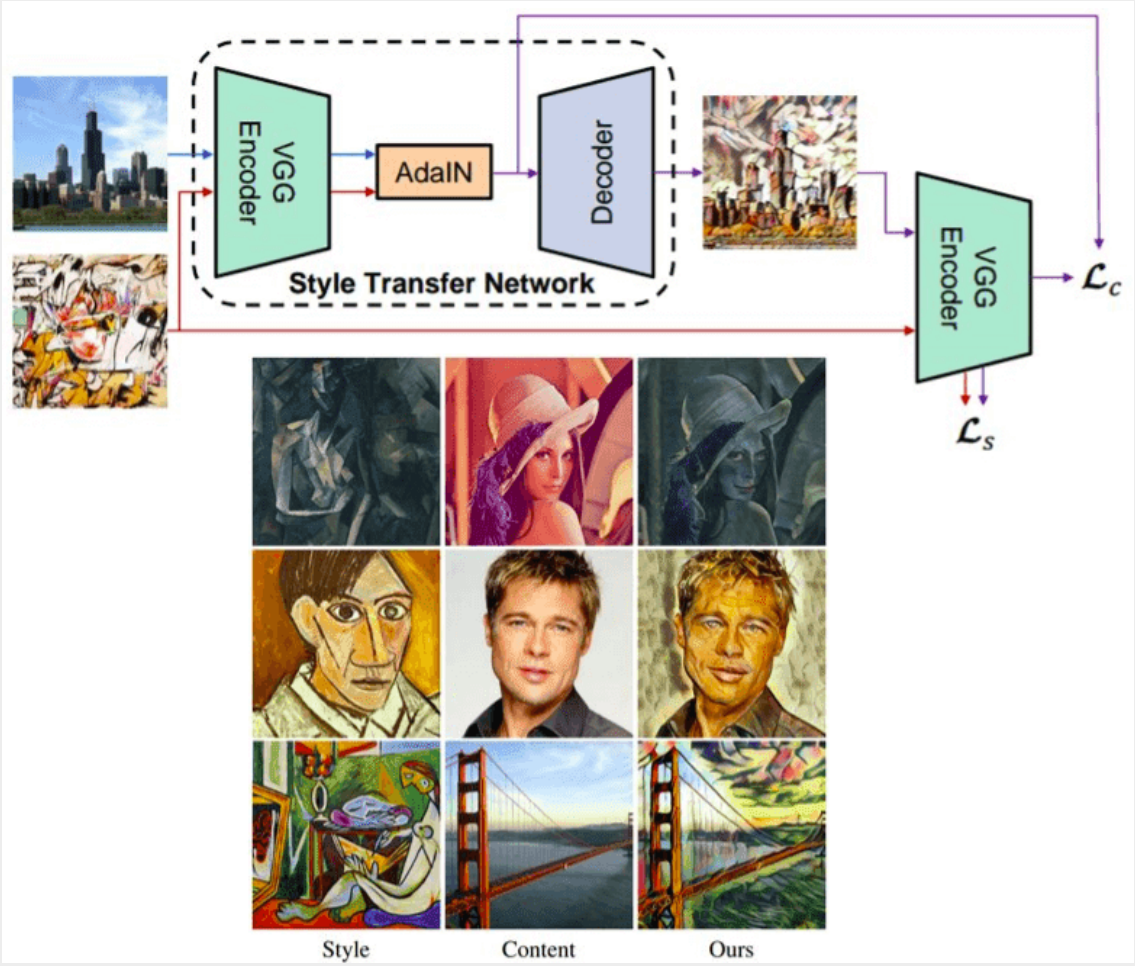
Instance normalization

$$\mathrm{IN}(x) = \gamma\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \beta$$

$$\mu(x) = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{wh}$$

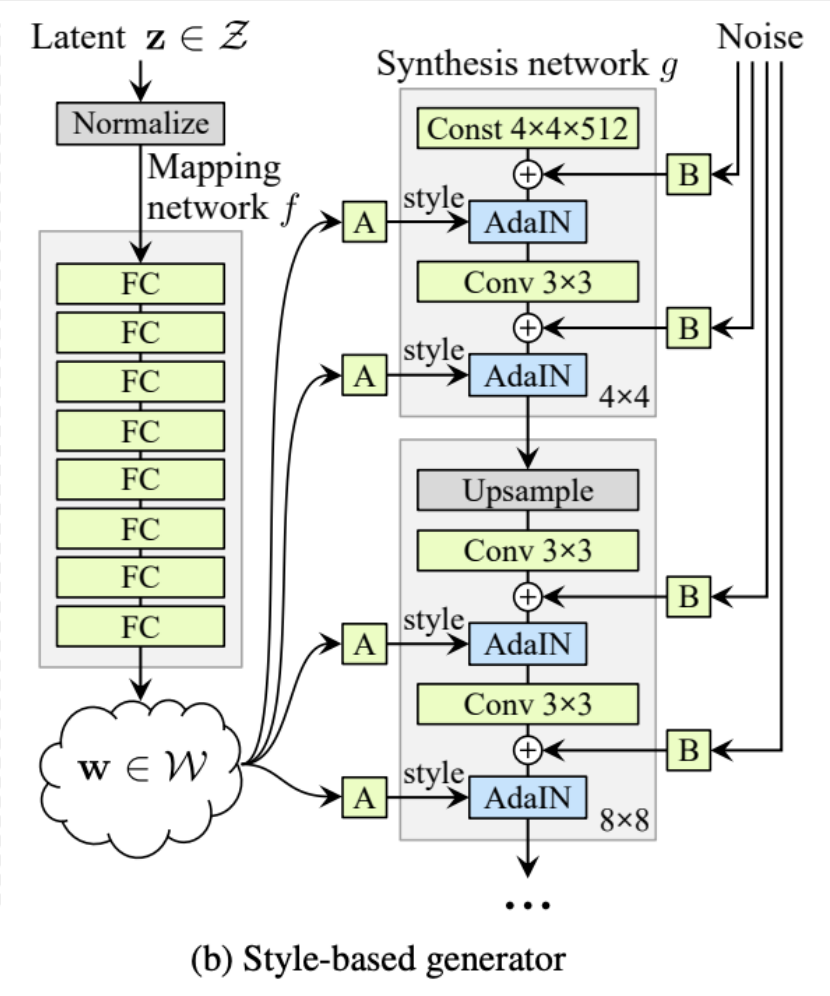
$$\sigma(x) = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{wh} - \mu(x))^2$$

Adaptive Instance normalization

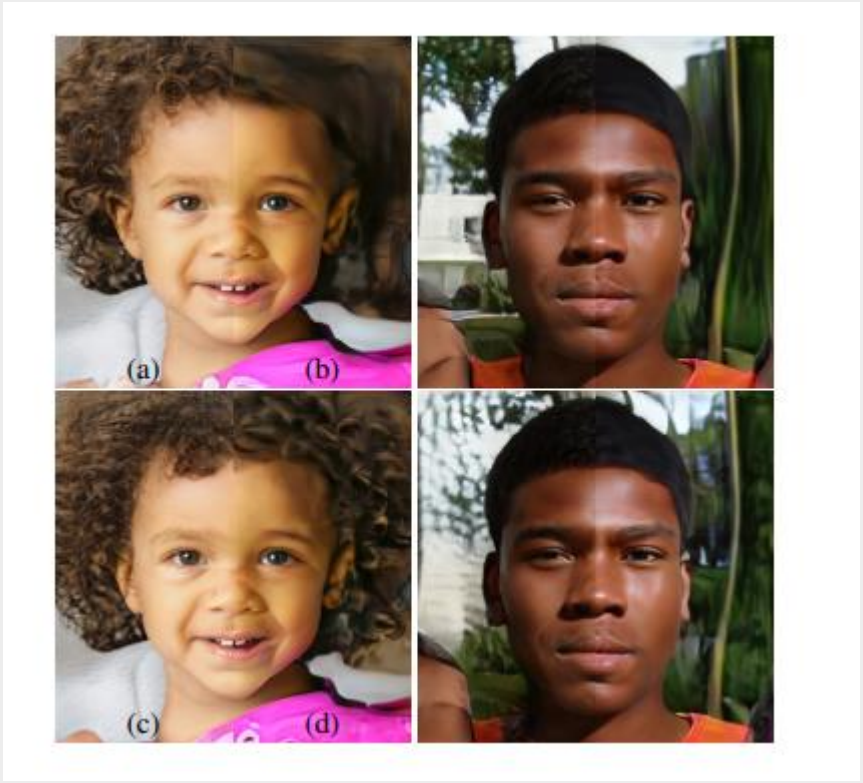


$$\text{AdaIN}(x) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)$$

StyleGAN - generator



(b) Style-based generator

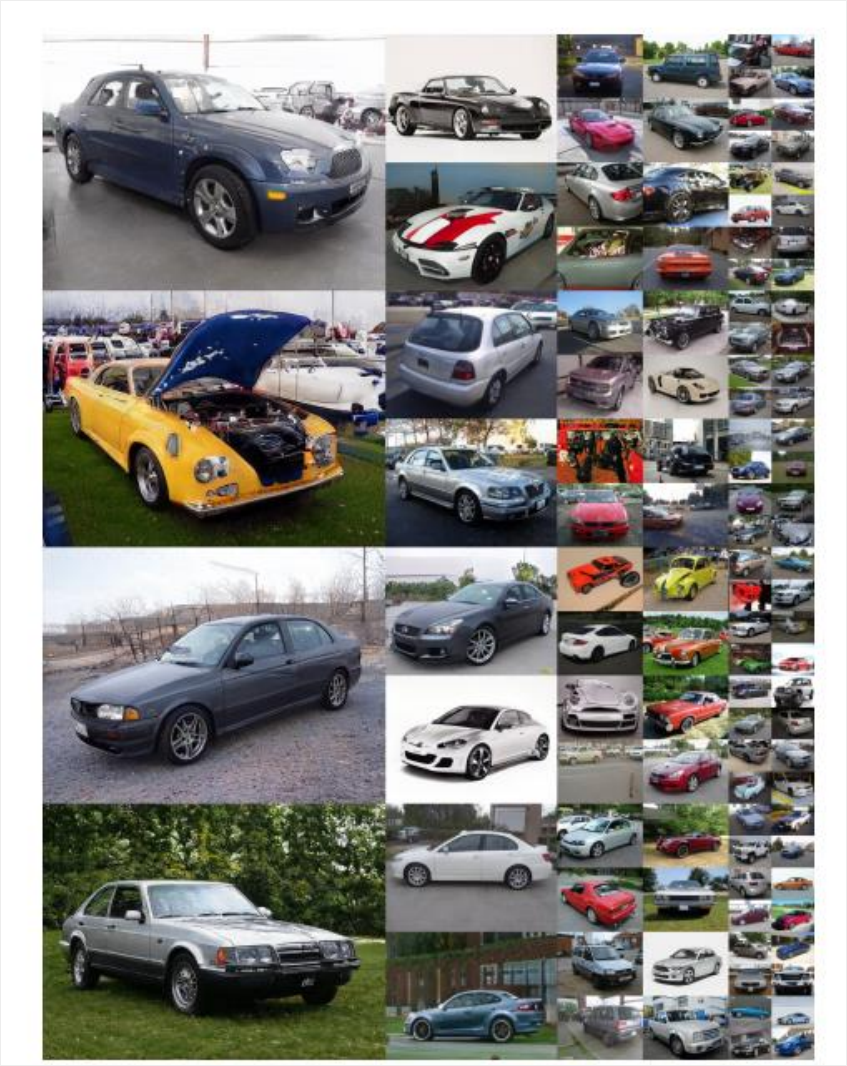


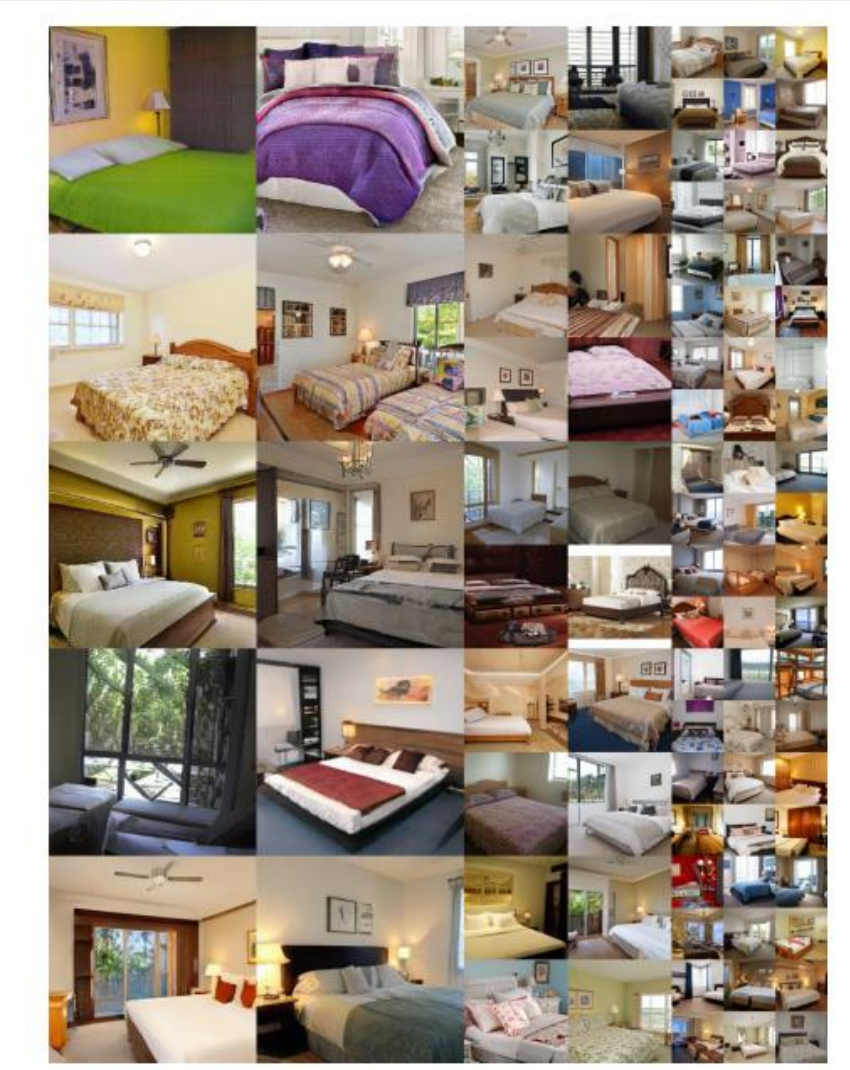
Влияние шума на генерацию. a) шум, добавленный во все слои, b) без добавления шума c) шум на второй половине блоков 64x64 -- 1024x1024, d) шума на первой половине блоков 4x4-32x32

$$\mathrm{AdaIN}(x) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)$$

[6]StyleGAN

StyleGAN - results





[6]StyleGAN

StyleGAN - results

[6]StyleGAN

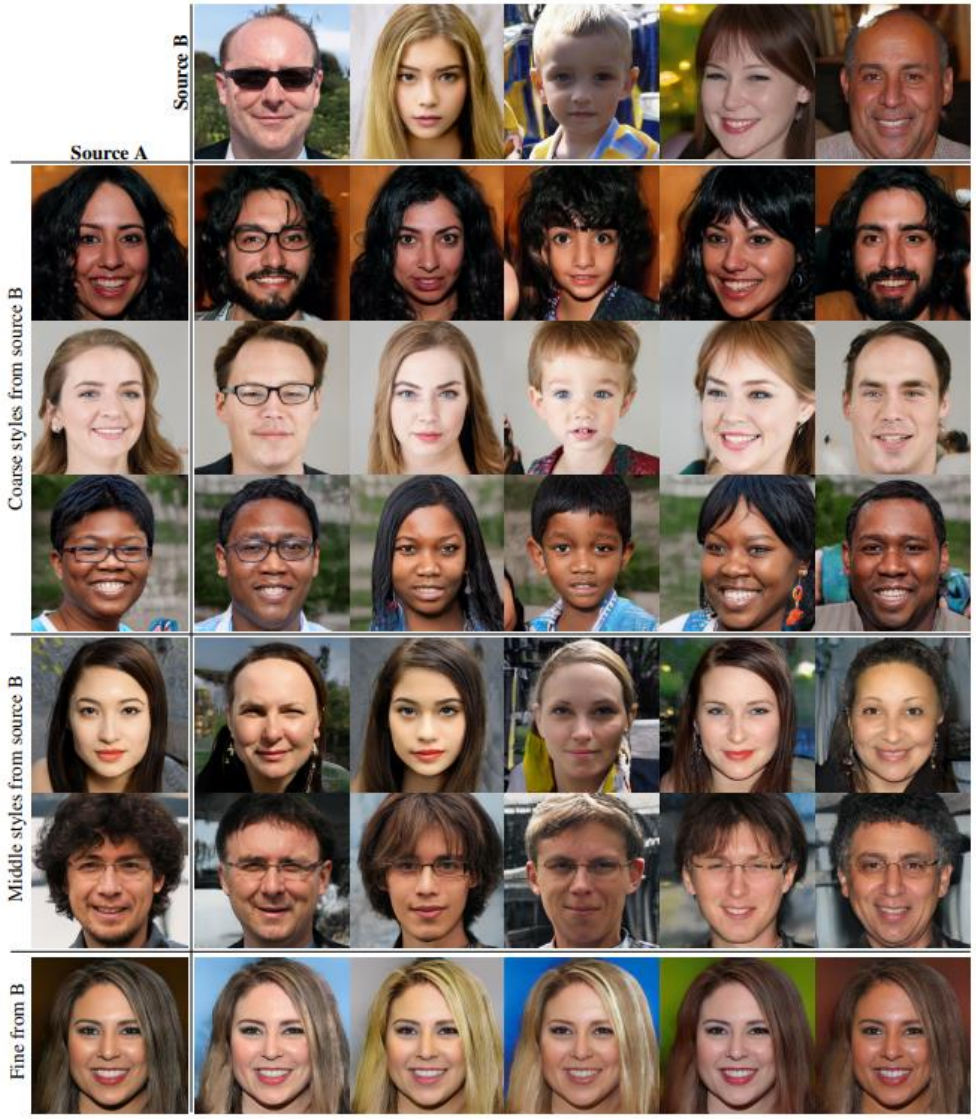
StyleGAN - style mixing

Можем смешивать различные латентные векторы

- если добавить стайл вектора в начальную стадию генерации - изменяются верхнеуровневые атрибуты, такие как поза головы, форма лица, появление очков от источника B, в то время как основные цвета и черты остаются от источника A.

- Если добавлять стайл векторы в середине процесса генерации, то меняются стиль волос, открытые/закрытые глаза и менее существенные особенности лица и сохраняются основная поза, форма головы от источника A.

- Использование стайл векторов источника B на последних разрешениях генерации влияет в основном на цветовую гамму изображения.



[6]StyleGAN

StyleGAN problems

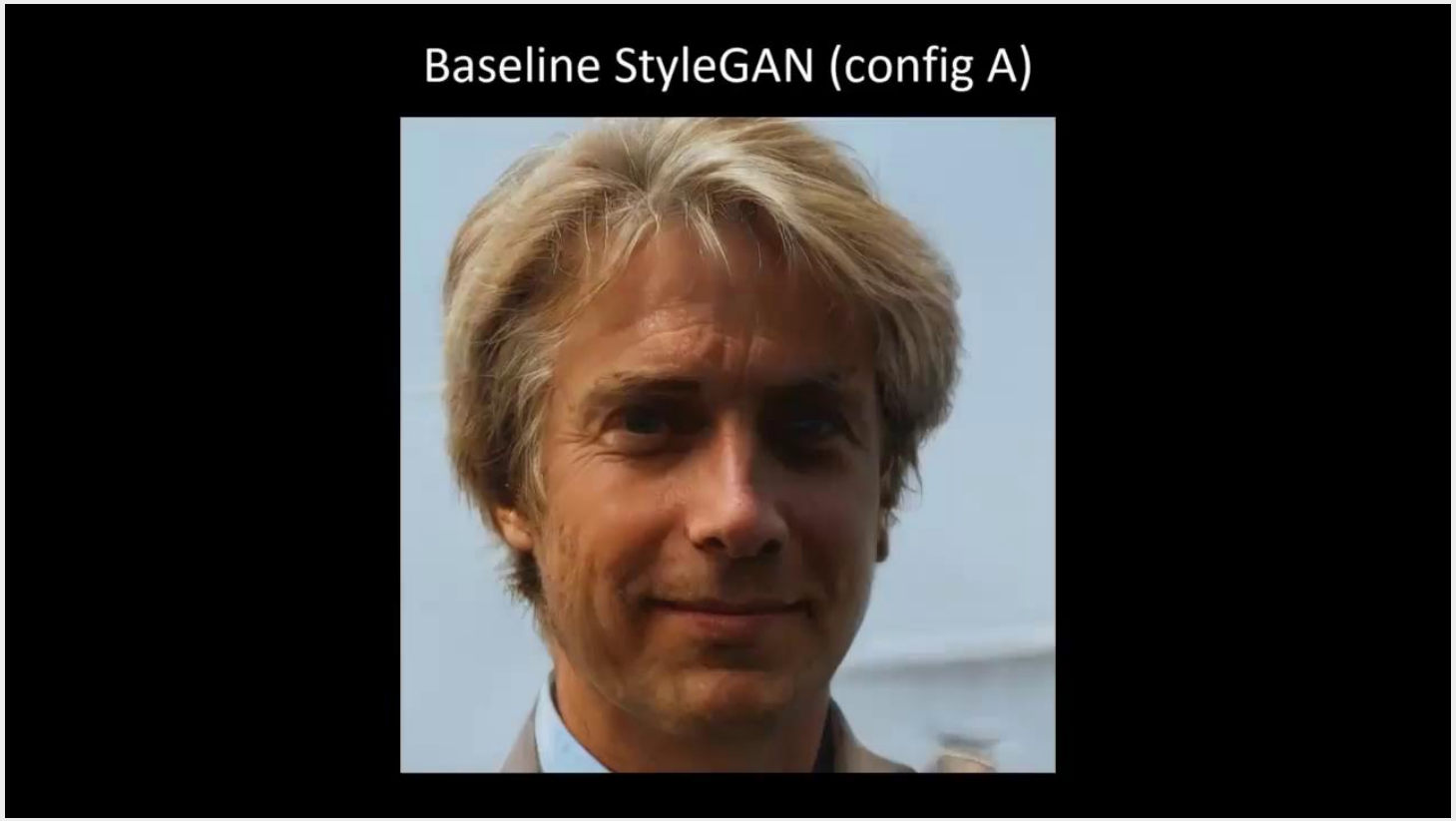
Проблема: появление артефактов из-за AdaIN нормализации

$$\mathrm{AdaIN}(x) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)$$

Figure 1. Instance normalization causes water droplet -like artifacts in StyleGAN images. These are not always obvious in the generated images, but if we look at the activations inside the generator network, the problem is always there, in all feature maps starting from the 64x64 resolution. It is a systemic problem that plagues all StyleGAN images.

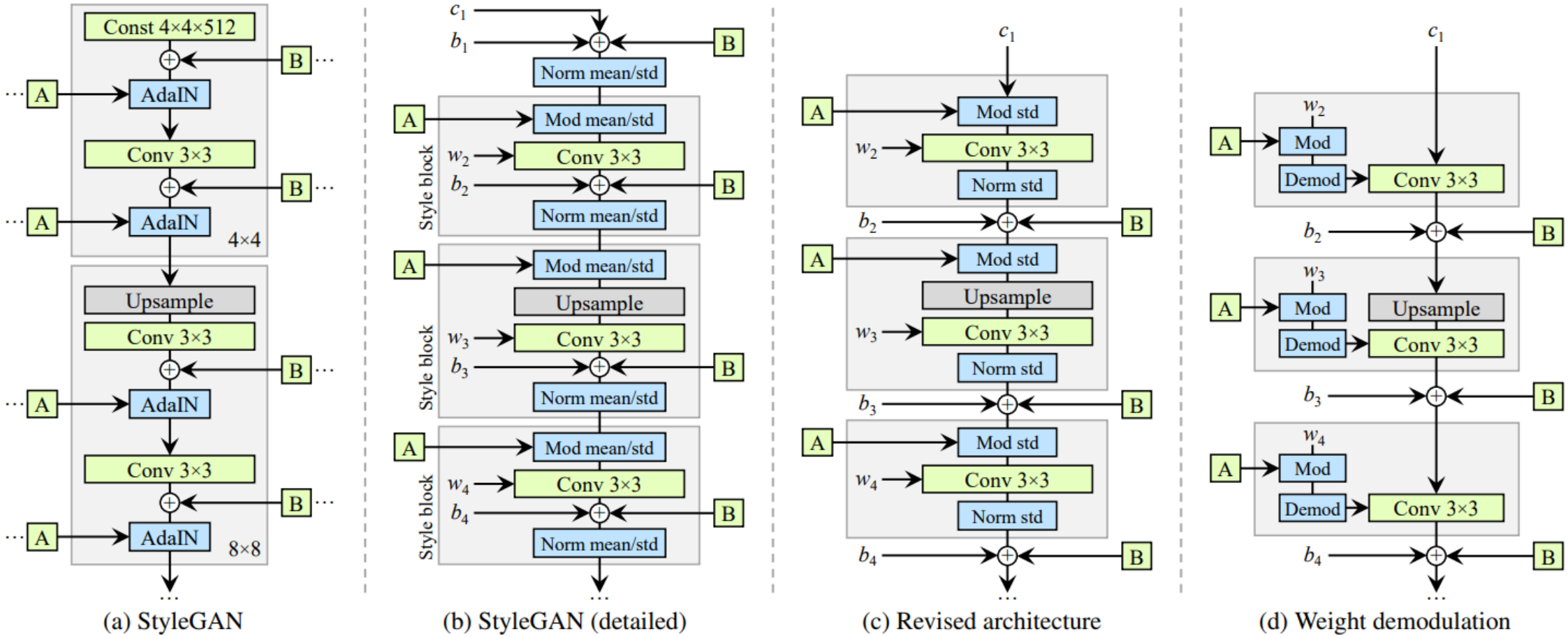
[7] StyleGAN2

StyleGAN problems



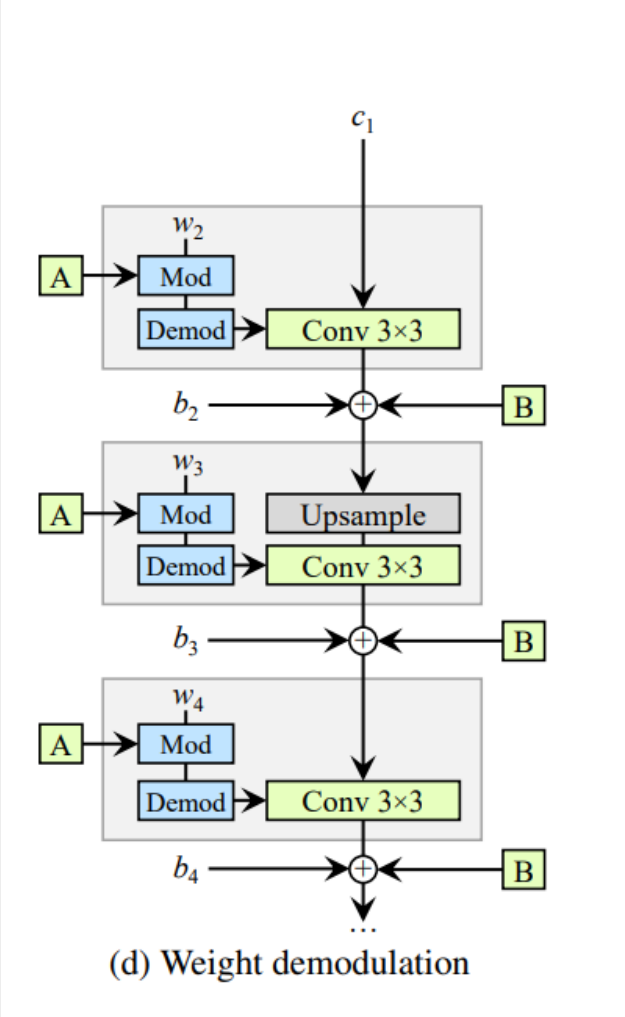
[7] StyleGAN2

StyleGAN2



[7] StyleGAN2

StyleGAN2 - weight modulation



(d) Weight demodulation

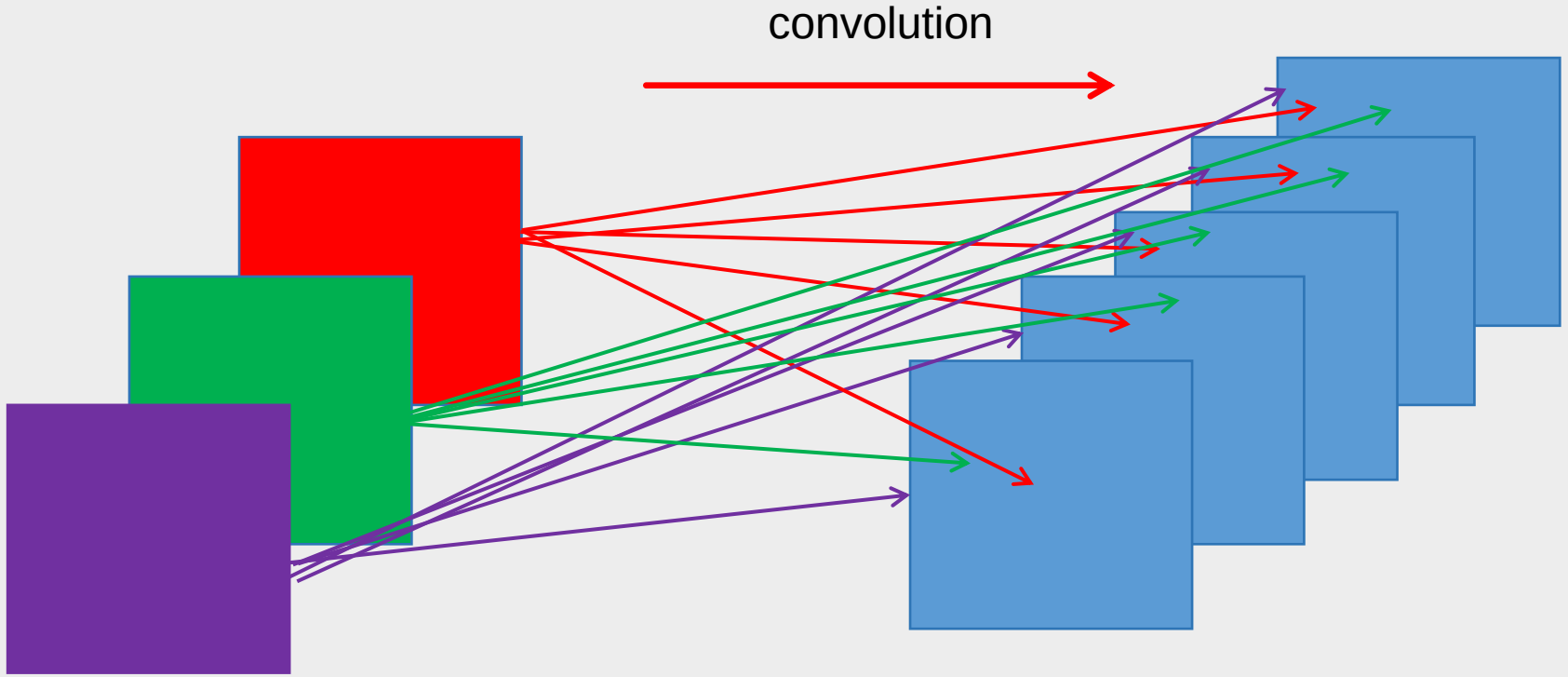
[7] StyleGAN2

Вместо AdaIN нормализации теперь используется модуляция и демодуляция весов генератора

Модуляция: $$w'_{i,j,k} = s_i w_{i,j,k}$$

Демодуляция: $$w'_{i,j,k} = \frac{w'_{i,j,k}}{\sqrt{\sum_{j,k} w'^2_{i,j,k} + \epsilon}}$$

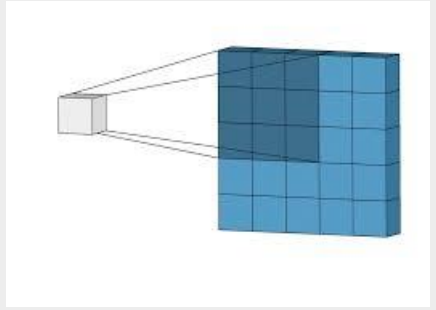
StyleGAN2 - weight modulation



стрелка -
convolutional kernel

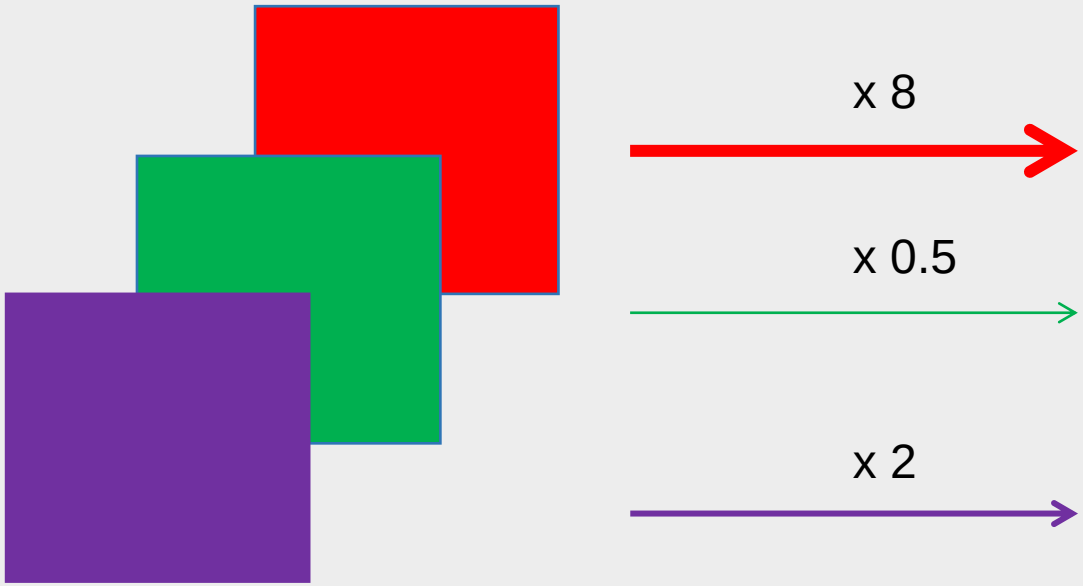
feature map

next feature map



Модуляция: $$w'_{i,j,k} = s_i w_{i,j,k}$$

StyleGAN2 - weight modulation

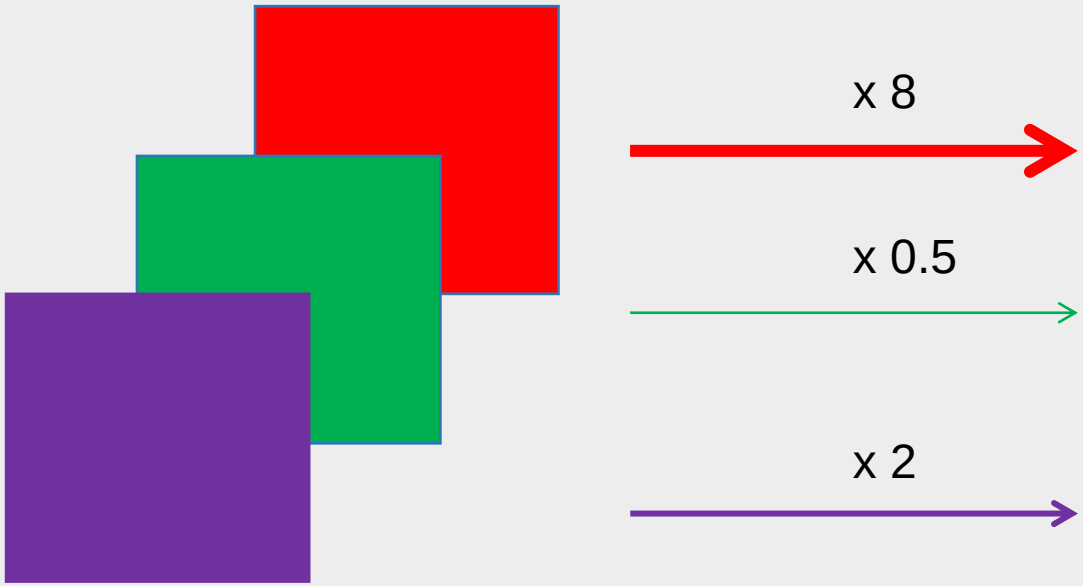


feature map

Вместо того, чтобы нормировать и умножать на скейл каждый канал - мы умножаем соответствующий ему convolution kernel

Модуляция: $$w'_{i,j,k} = s_i w_{i,j,k}$$

StyleGAN2 - weight demodulation



feature map

Демодуляция - нормируем, чтобы стандартное отклонение стало равным 1

Демодуляция:

$$w'_{i,j,k} = \frac{w'_{i,j,k}}{\sqrt{\sum_{j,k} w'^2_{i,j,k} + \epsilon}}$$

StyleGAN2 - results

Больше нет таких артефактов, как у StyleGAN!

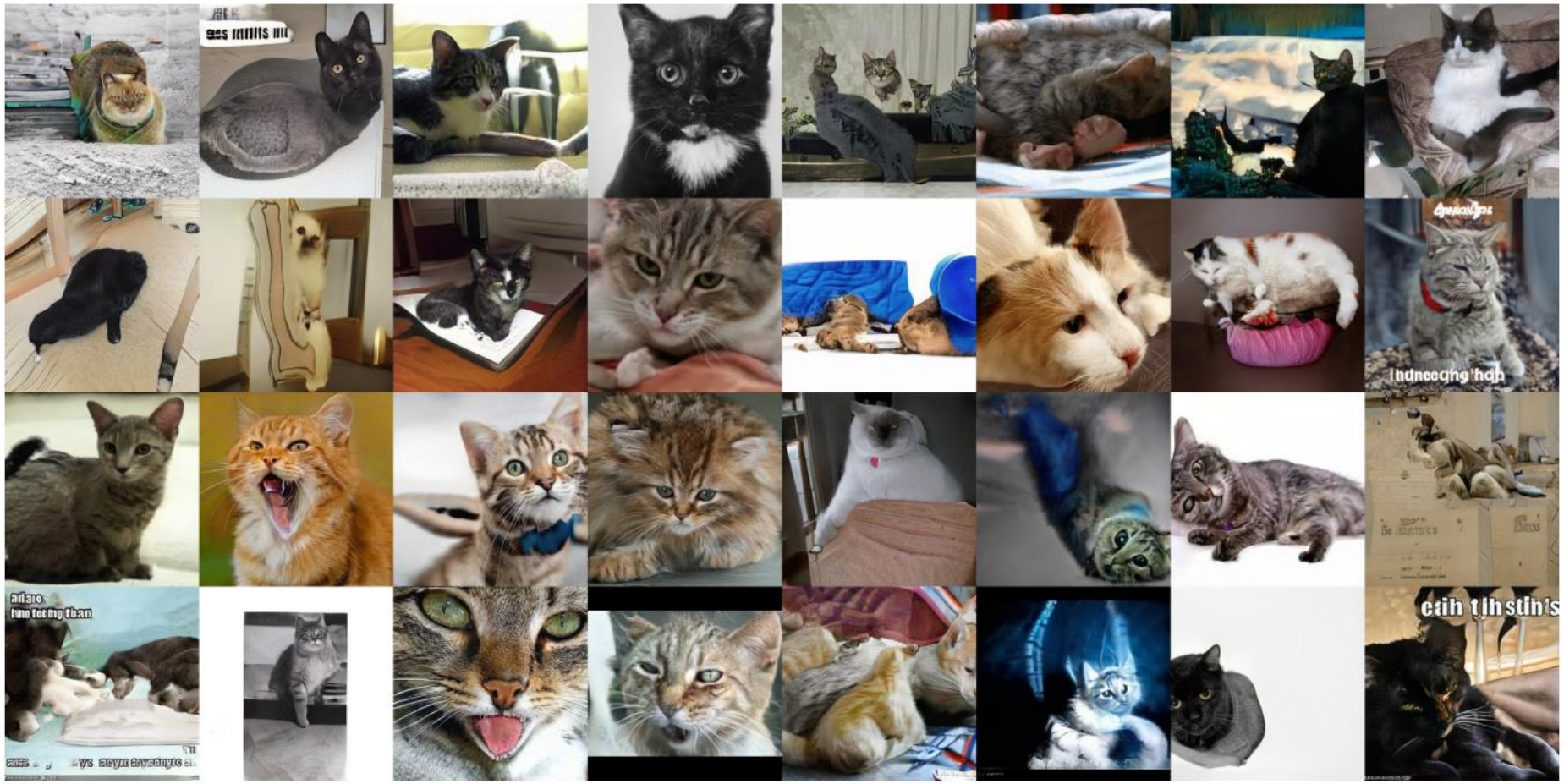
[7] StyleGAN2

StyleGAN2 - results

StyleGAN — generated images | StyleGAN2 — generated images | StyleGAN2 — real images

[7] StyleGAN2

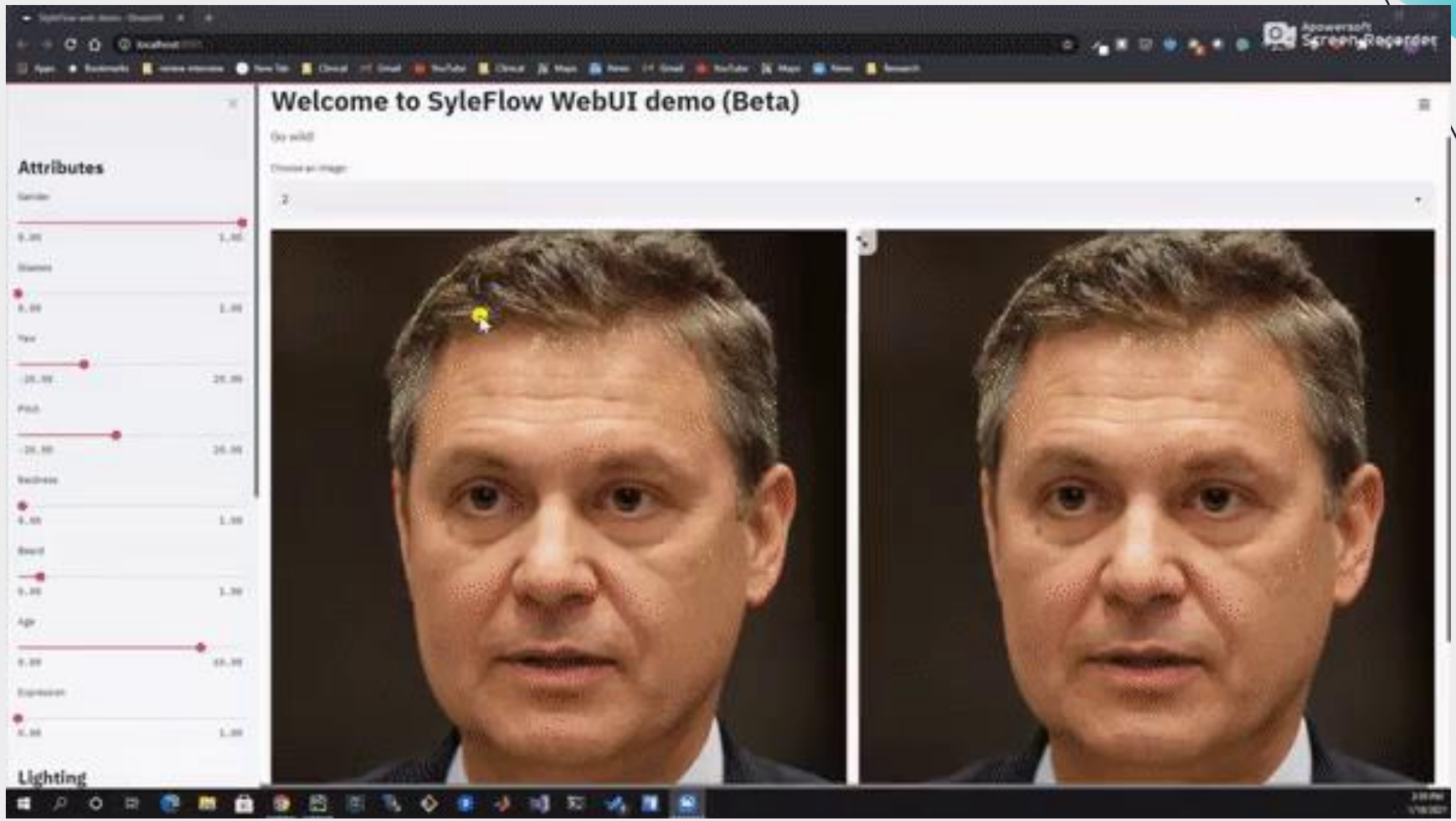
StyleGAN2 - results



Model 2: FID = 8.53, P = 0.62, R = 0.29, PPL = 387

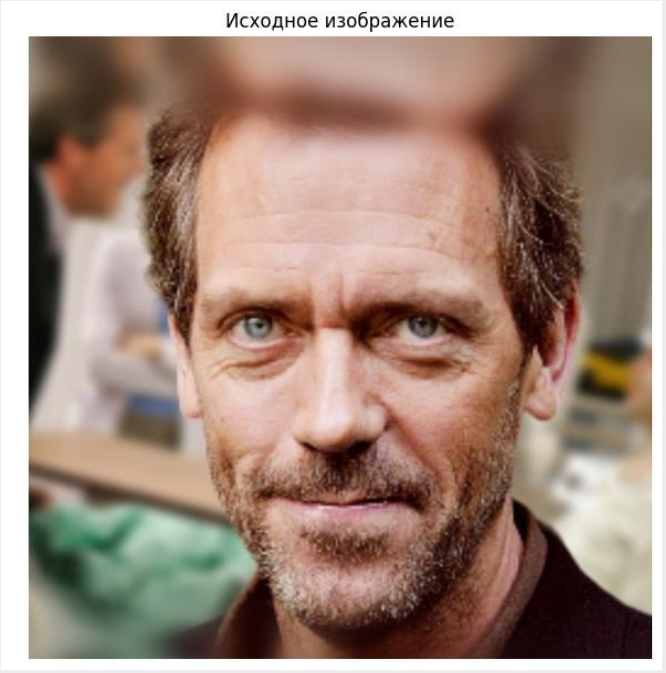
[7] StyleGAN2

Image editing using StyleGAN



[8] Styleflow

А что с реальными картинками?



$$z \in \mathcal{Z}(1 \times 512)$$

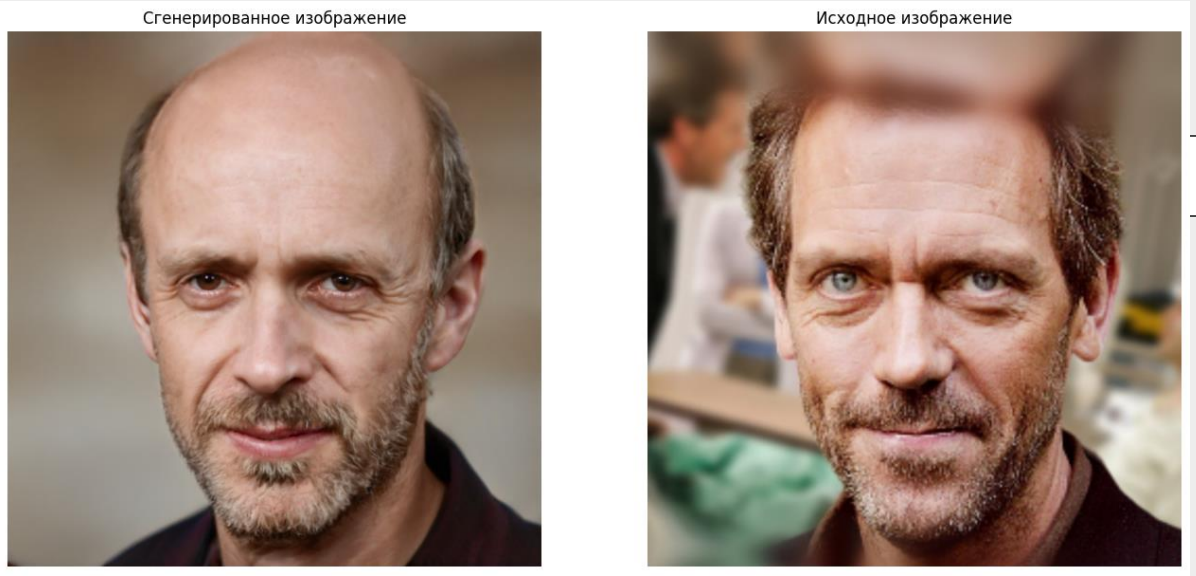
Оптимизация латентных векторов

$z \in \mathcal{Z}$ → StyleGAN →

Update

Loss function

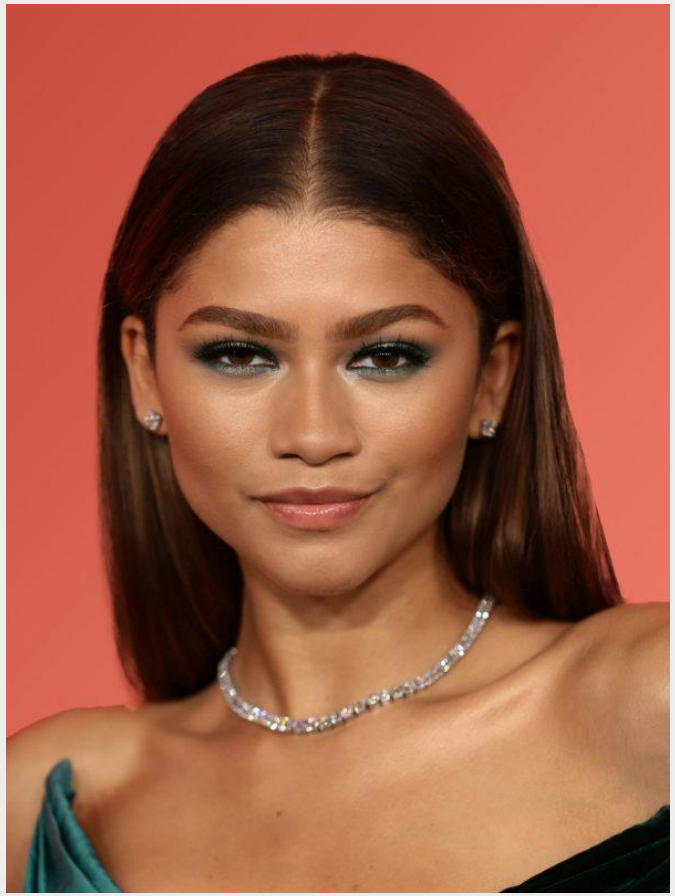
Invert to z



Real world image

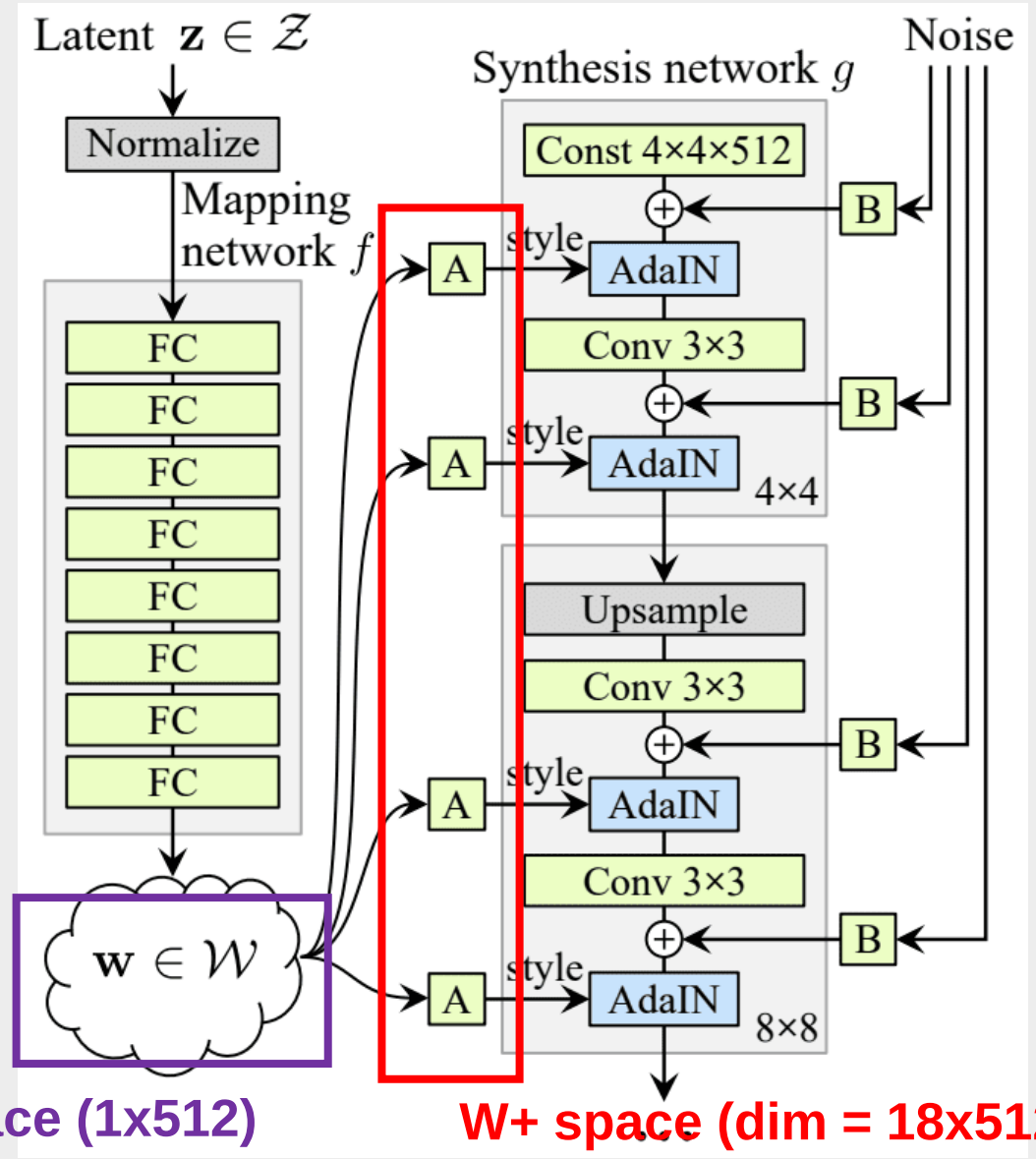
Generated image

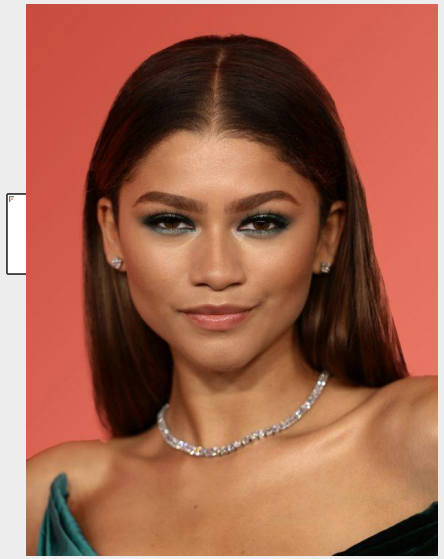
Invert to w



$$w \in \mathcal{W}(1 \times 512)$$

W+ пространство





$$w^+ \in \mathcal{W}+(18 \times 512)$$

W space (1x512)

W+ space (dim = 18x512)

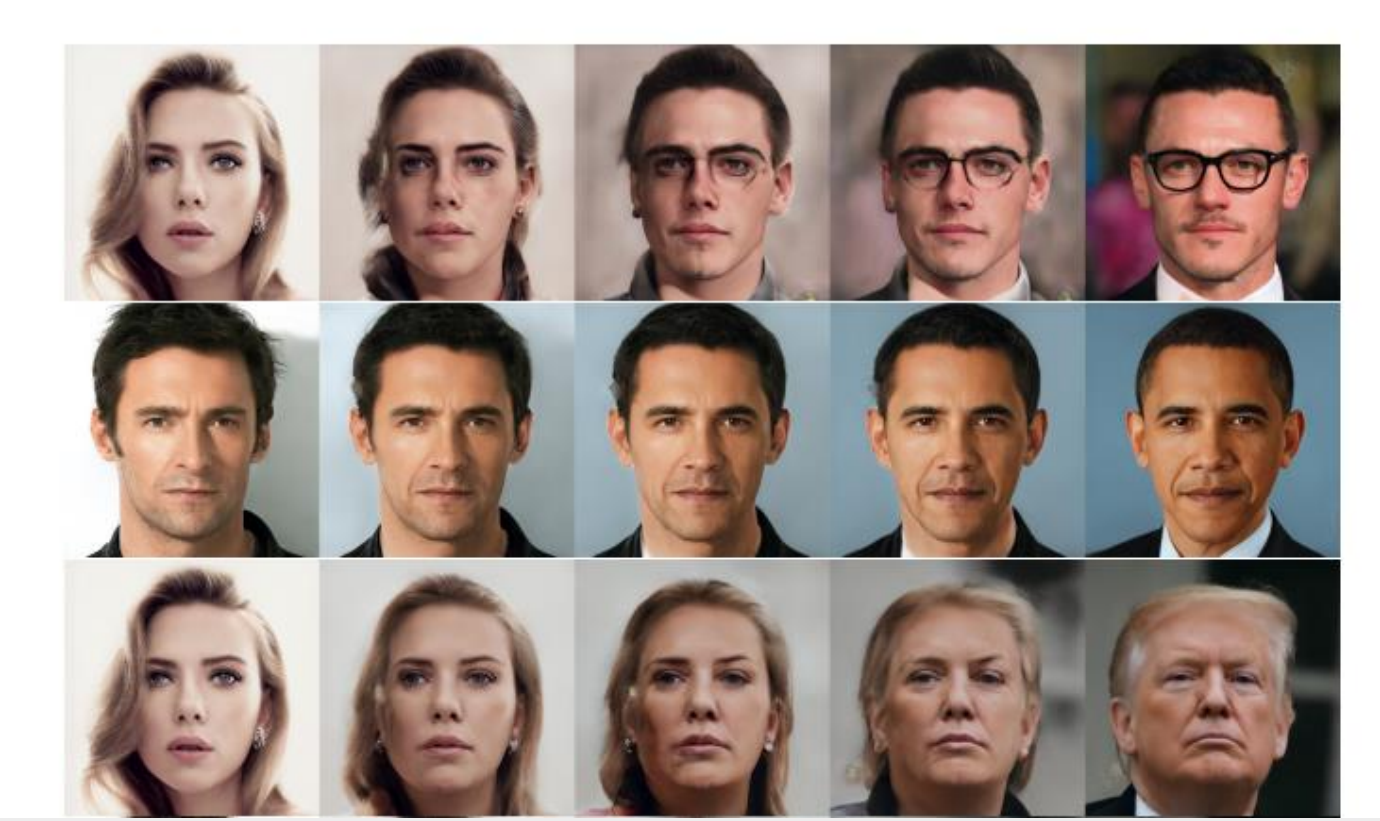
Image to StyleGAN

Figure 1: Top row: input images. Bottom row: results of embedding the images into the StyleGAN latent space.

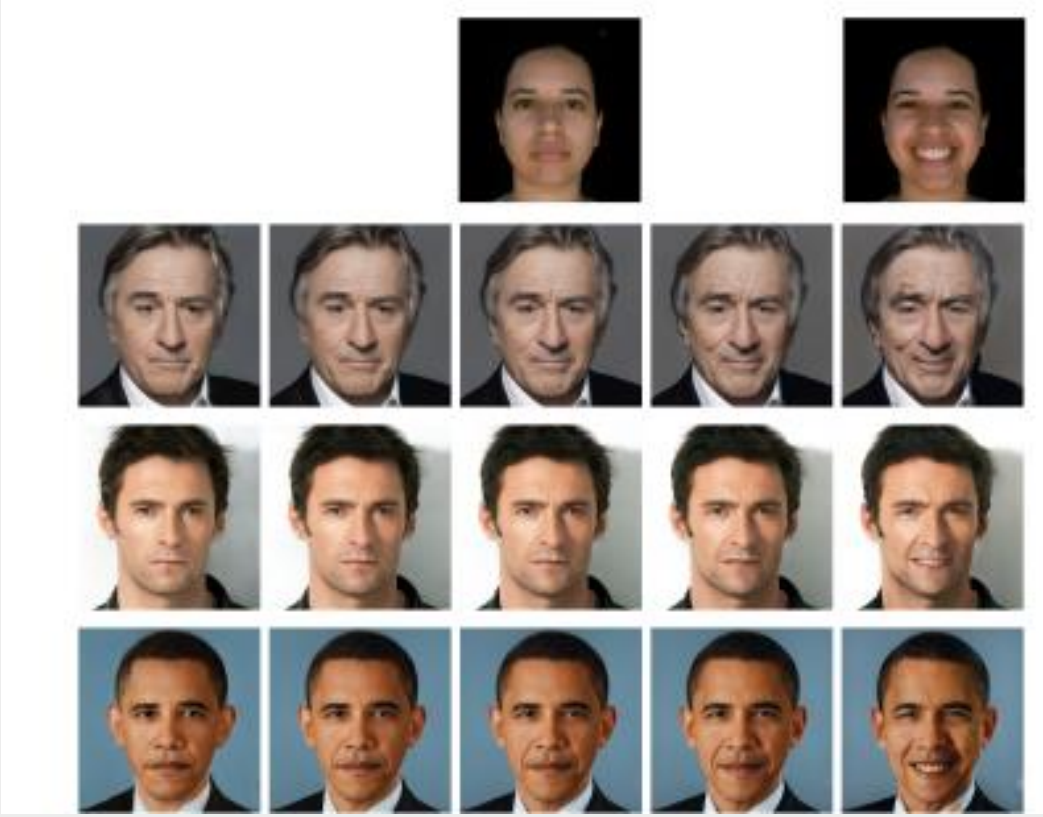
Найдите w, который воспроизводит изображение, выполняя градиент спуск в W+

[9] Image2stylegan

Image to StyleGAN



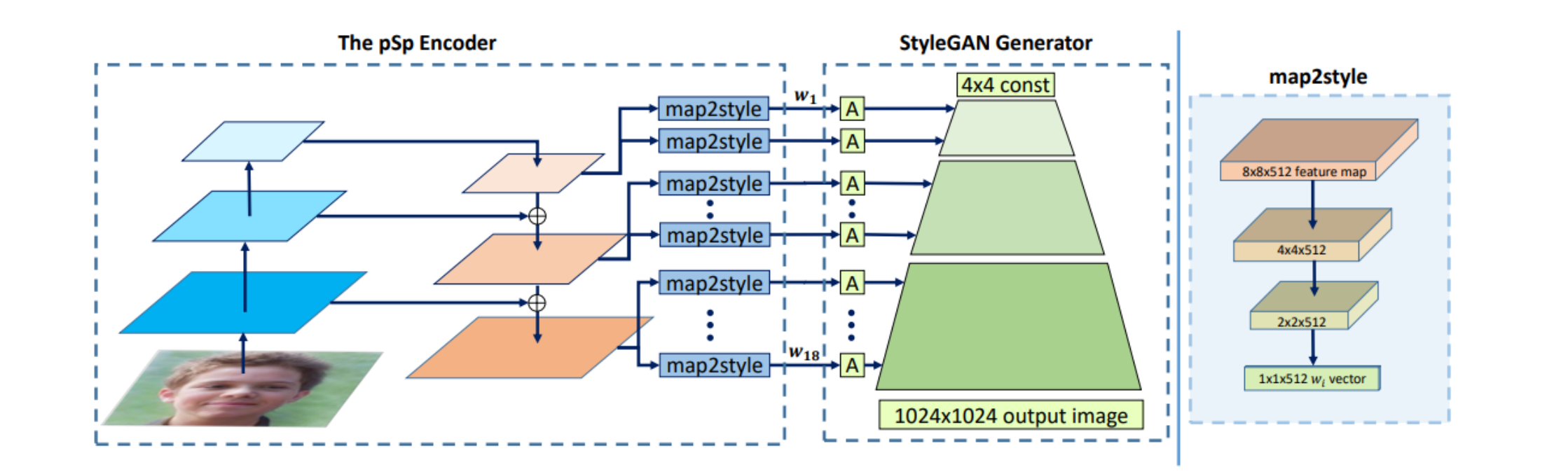
Интерполяция



Expression transfer

[9] Image2stylegan

Encoding in Style



Кодирование с помощью сетки вместо оптимизации

[10] Encoding in style

Encoding in Style

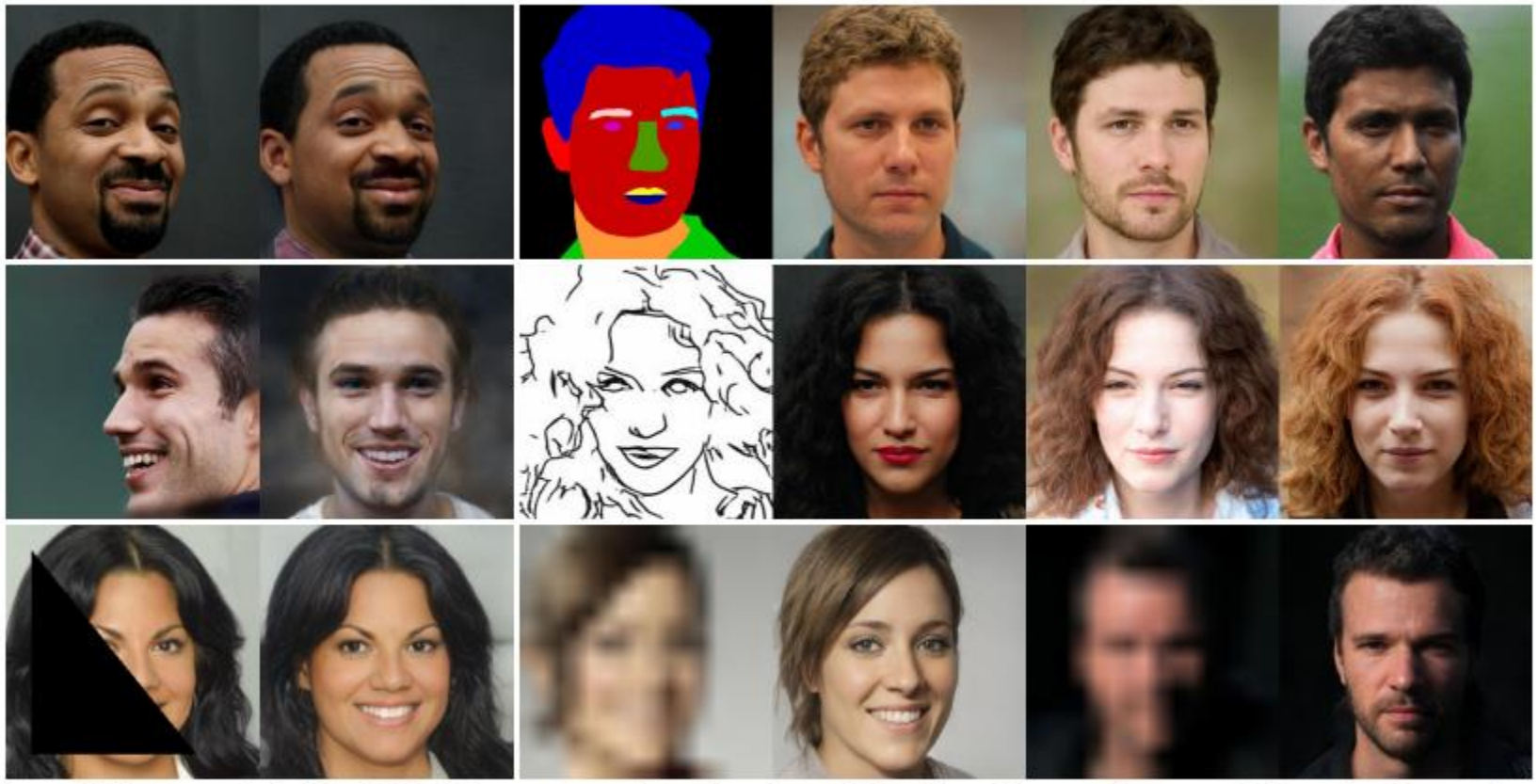
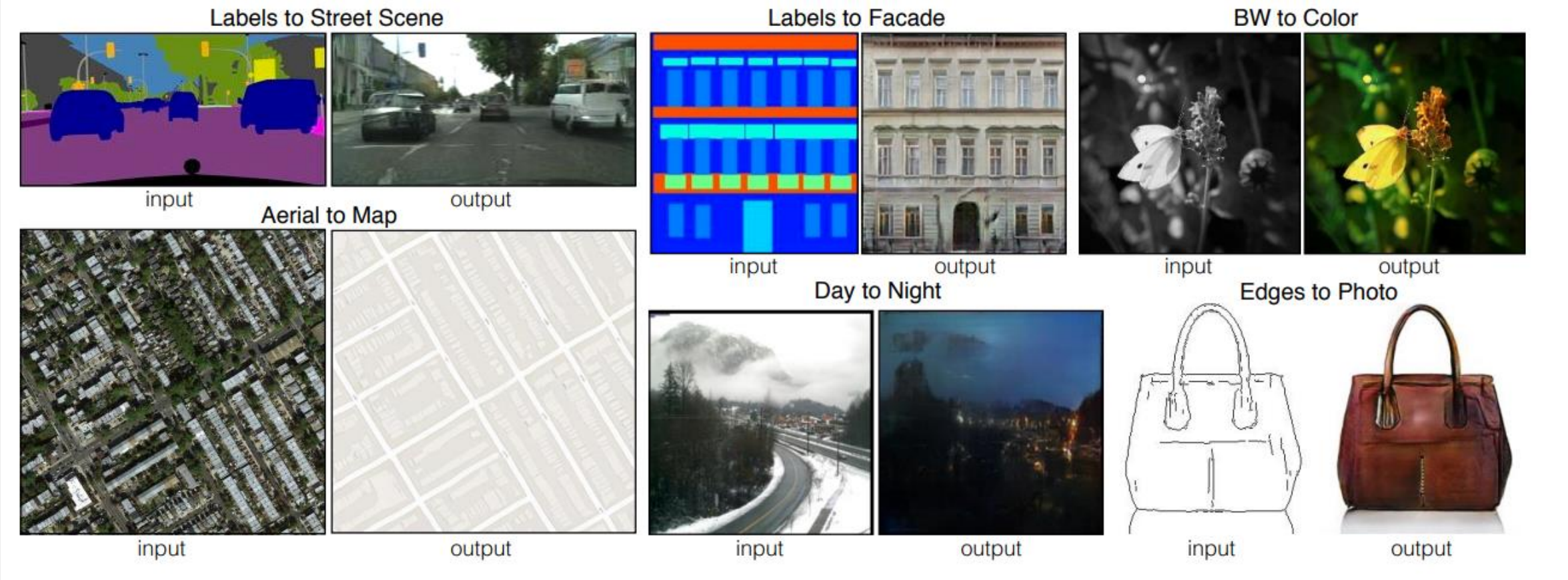


Figure 1. The proposed pixel2style2pixel framework can be used to solve a wide variety of image-to-image translation tasks. Here we show results of pSp on StyleGAN inversion, multi-modal conditional image synthesis, facial frontalization, inpainting and super-resolution.

[10] Encoding in style

Pix2pix



[11] Pix2Pix

Pix2pix

- Задача Image translation
- Модель принимает на вход изображение и на выходе тоже даёт изображение
- Обучается на парных данных

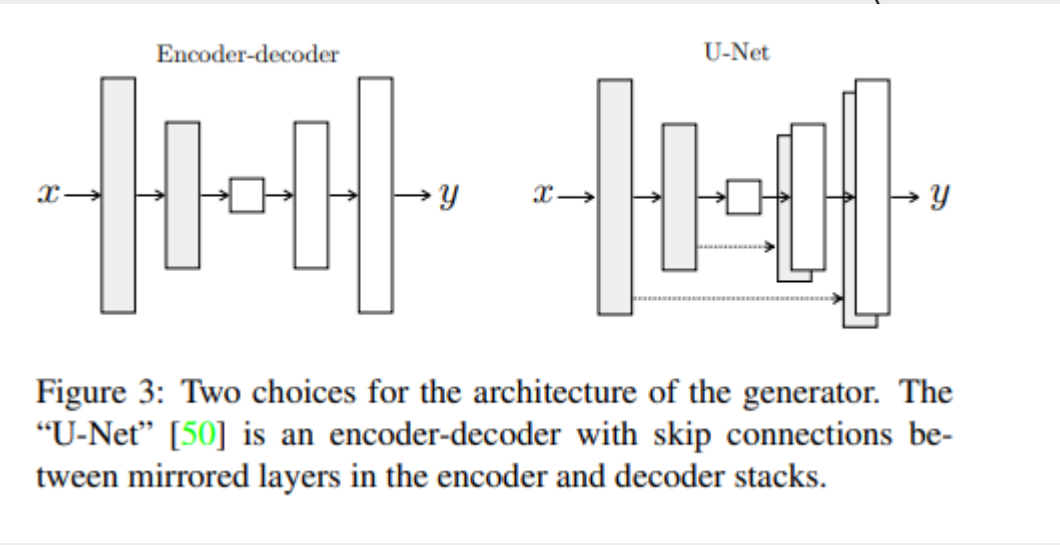
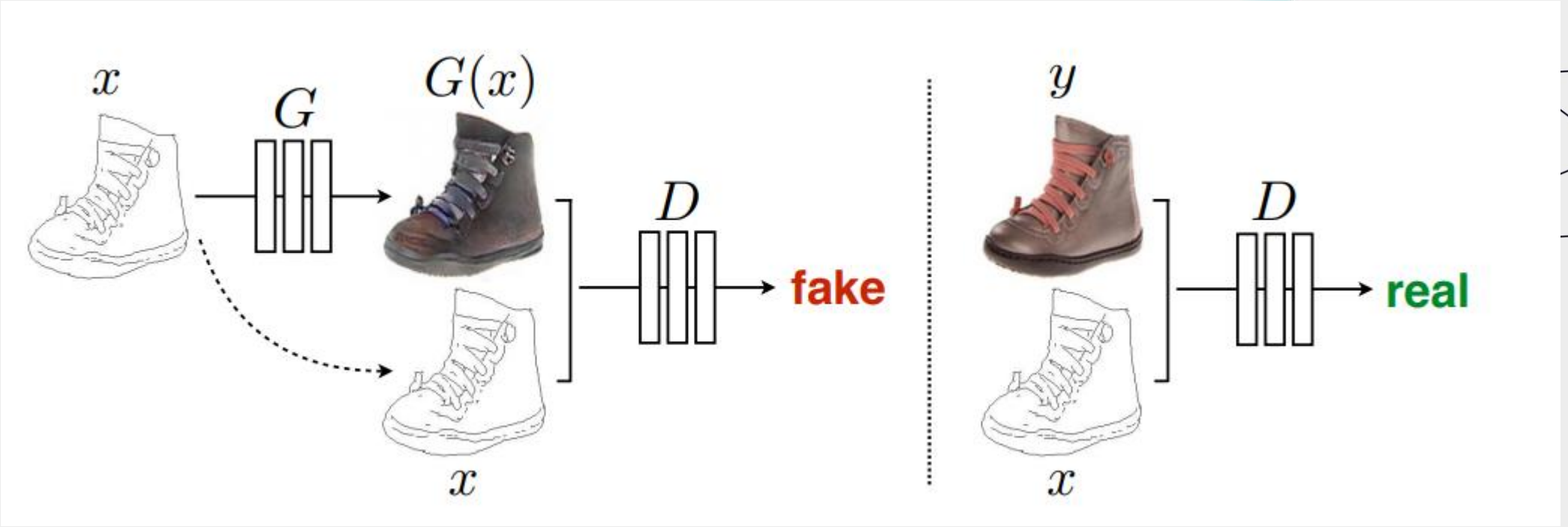


Figure 3: Two choices for the architecture of the generator. The "U-Net" [50] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

[11] Pix2Pix

Pix2pix

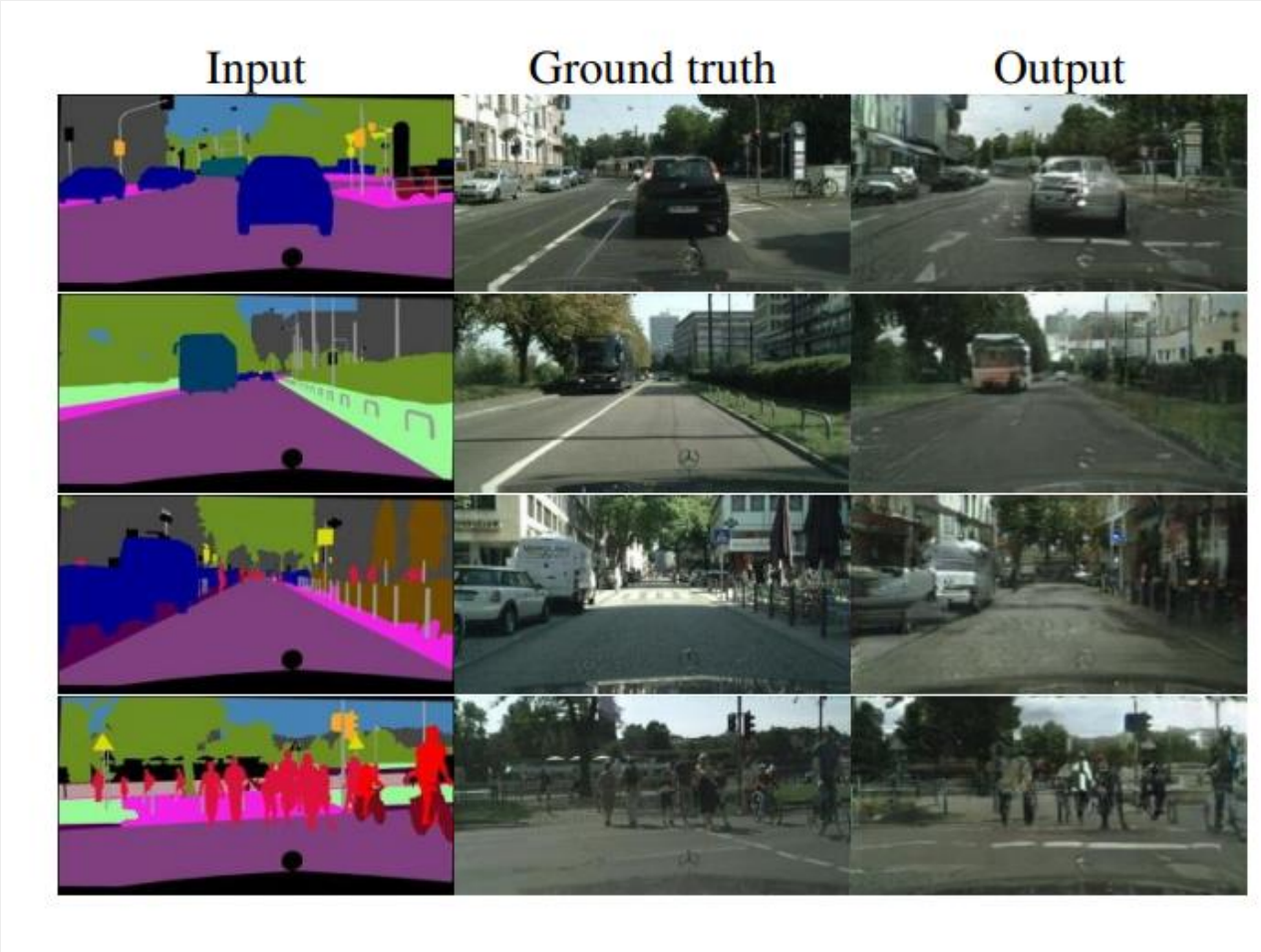


Дискриминатор работает с парой изображений, учится разделять реальные и сгенерированные картинки, генератор в итоге обучается благодаря следующей оптимизации:

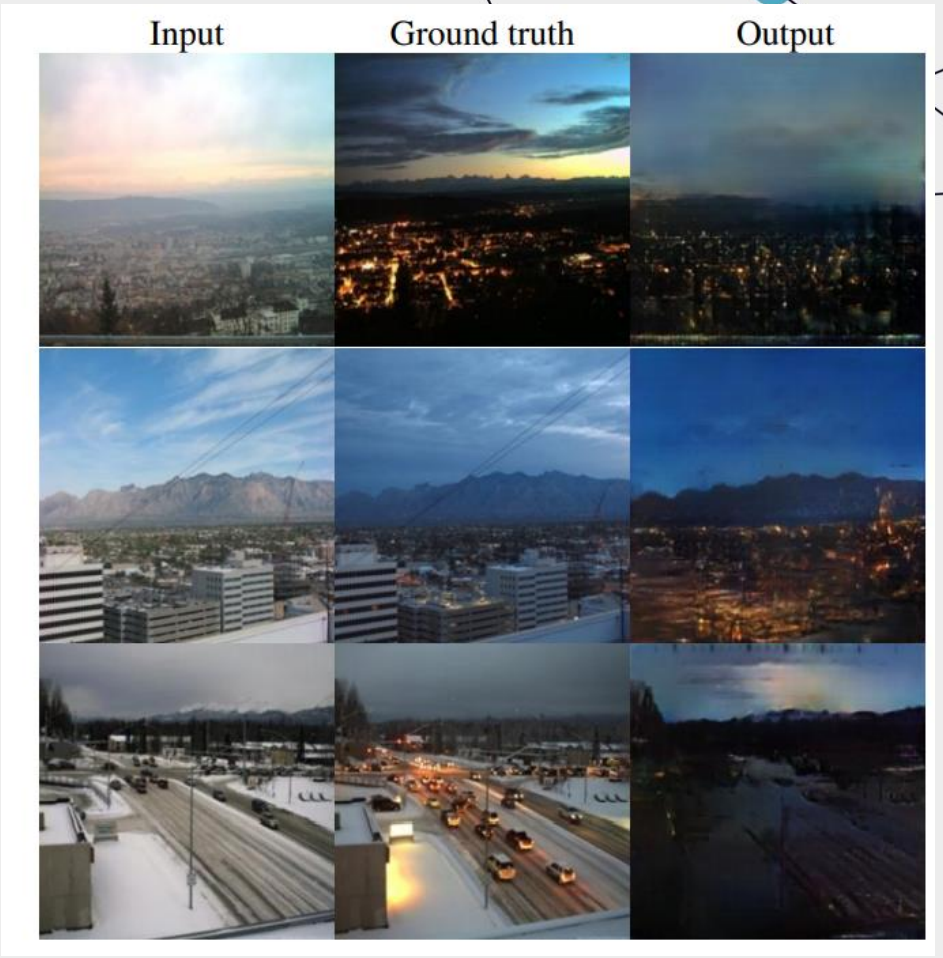
$$G^* = \arg\min_G \max_D \mathcal{L}_{GAN}(G, D) + \lambda \mathcal{L}_1(G)$$

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))]$$

Pix2pix



сегментационная маска - изображение

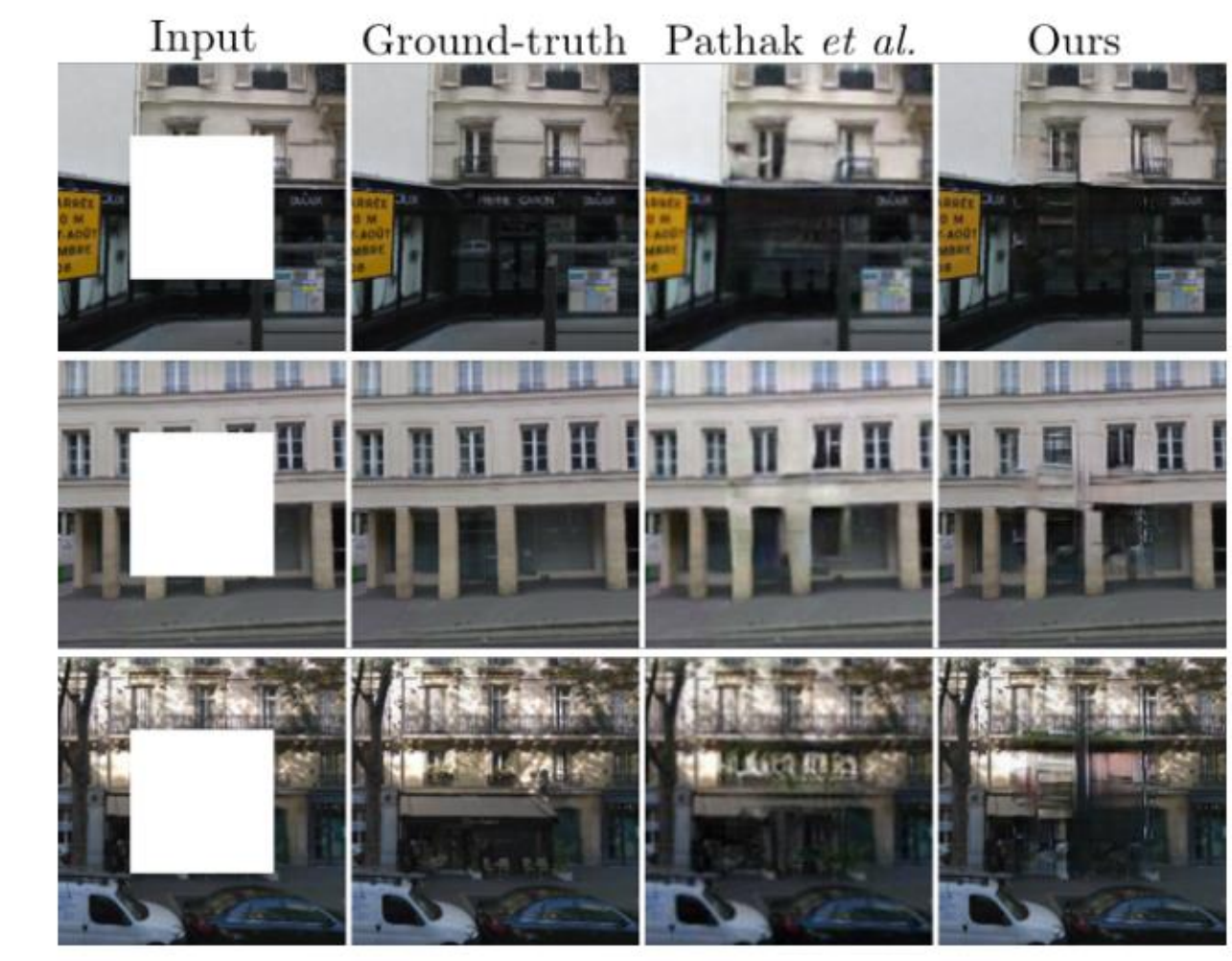


смена времени суток

[11] Pix2Pix

Pix2pix

скетч - изображение



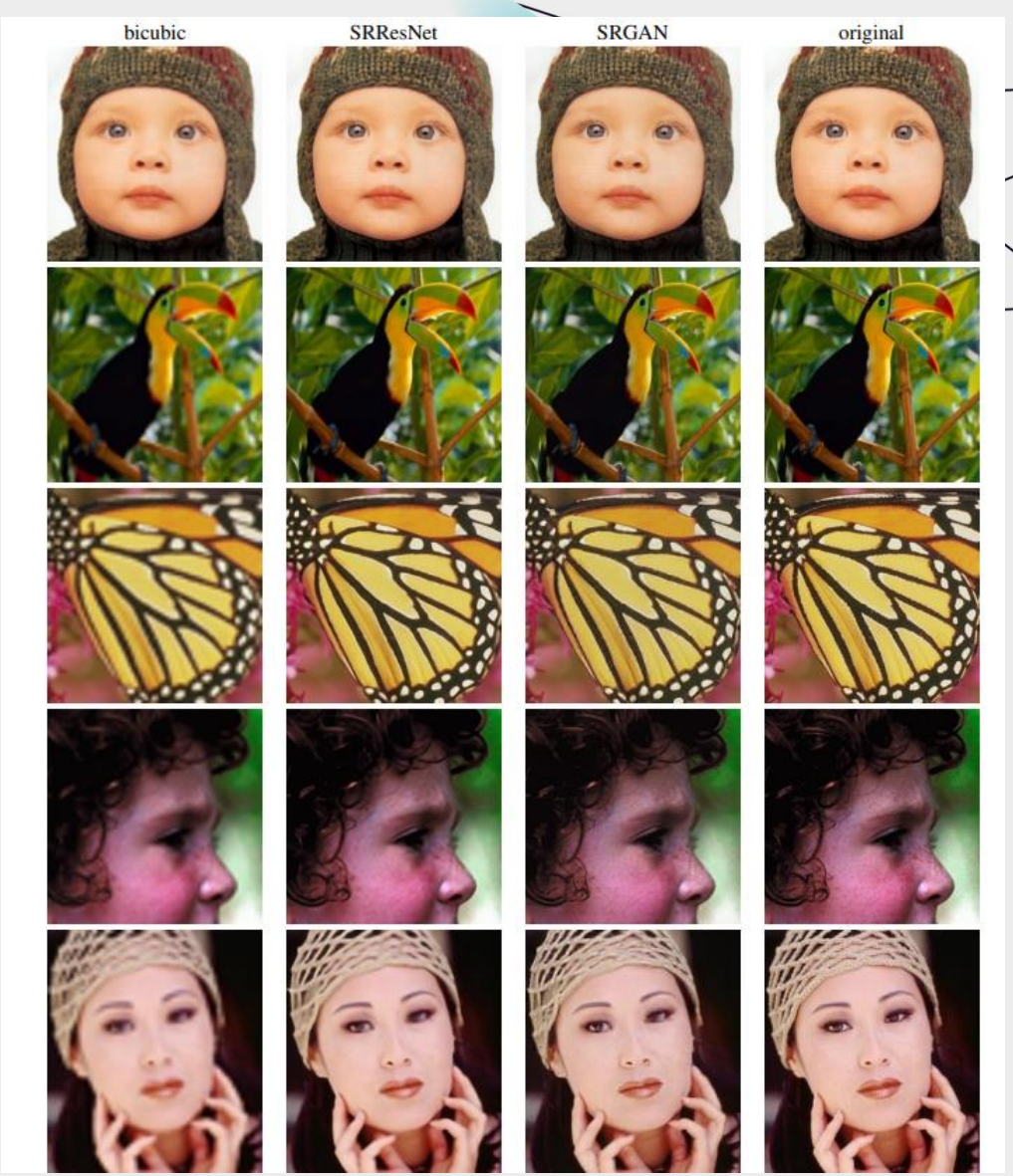
инпейнтинг

[11] Pix2Pix

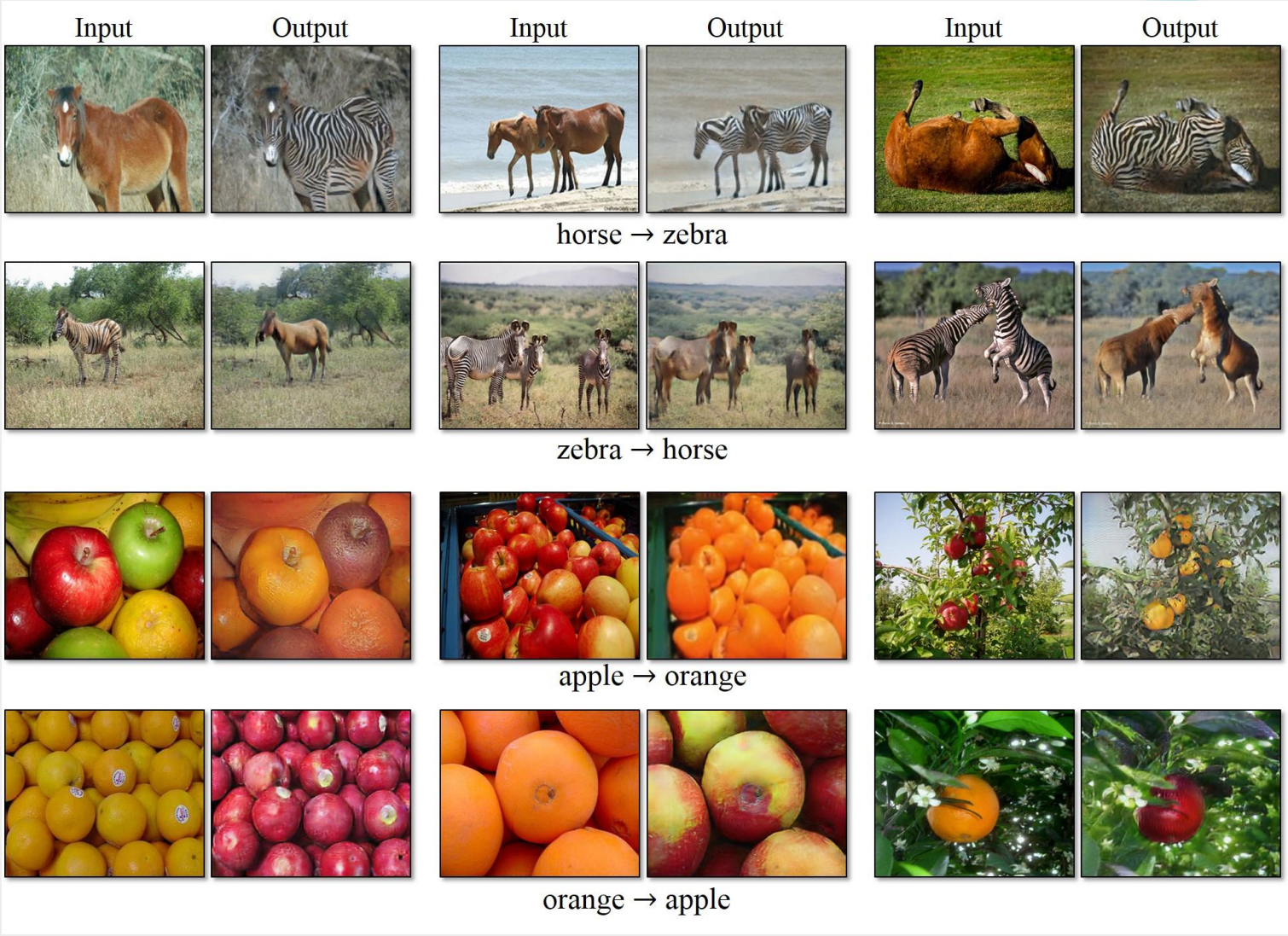
Superresolution GAN - SRGAN

Figure 14: Results for **Set14** using bicubic interpolation, SRResNet and SRGAN. [4× upscaling]

[12]SRGAN

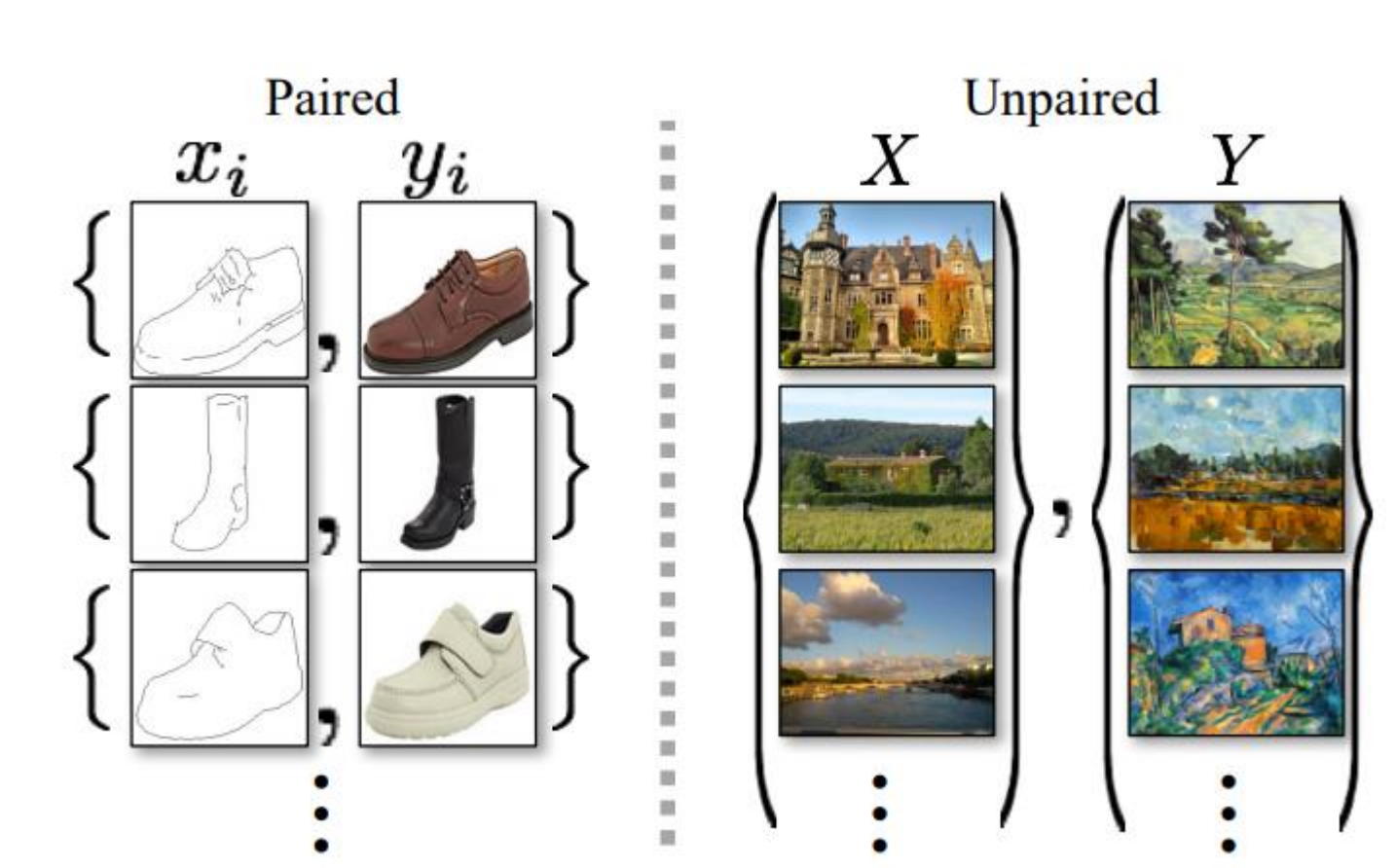


CycleGAN



[13]CycleGAN

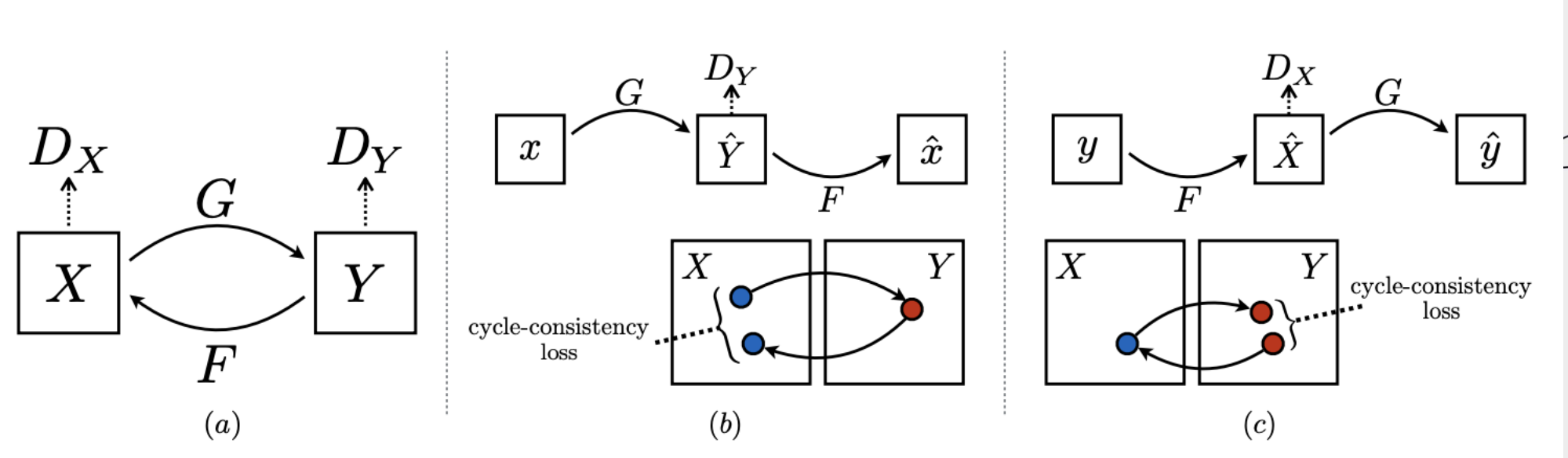
CycleGAN



[13]CycleGAN

- Задача Image translation
- Нет парных данных для обучения

CycleGAN



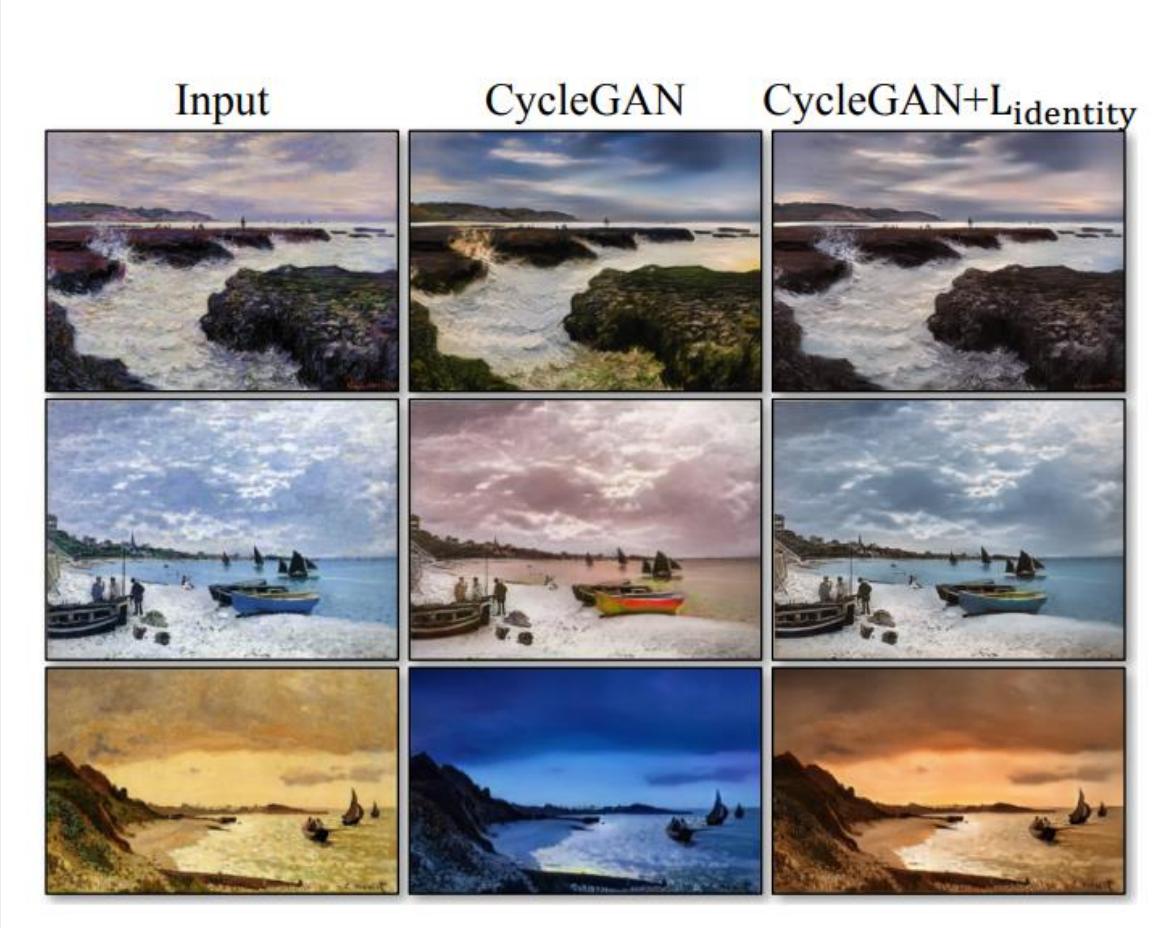
$$\mathcal{L}_{GAN_G}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]$$

$$\mathcal{L}_{GAN_F}(F, D_X, X, Y) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D_X(x)] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log(1 - D_X(F(y))]$$

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]$$

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN_G}(G, D_Y, X, Y) + \mathcal{L}_{GAN_F}(F, D_X, X, Y) + \lambda \mathcal{L}_{cyc}(G, F)$$

CycleGAN

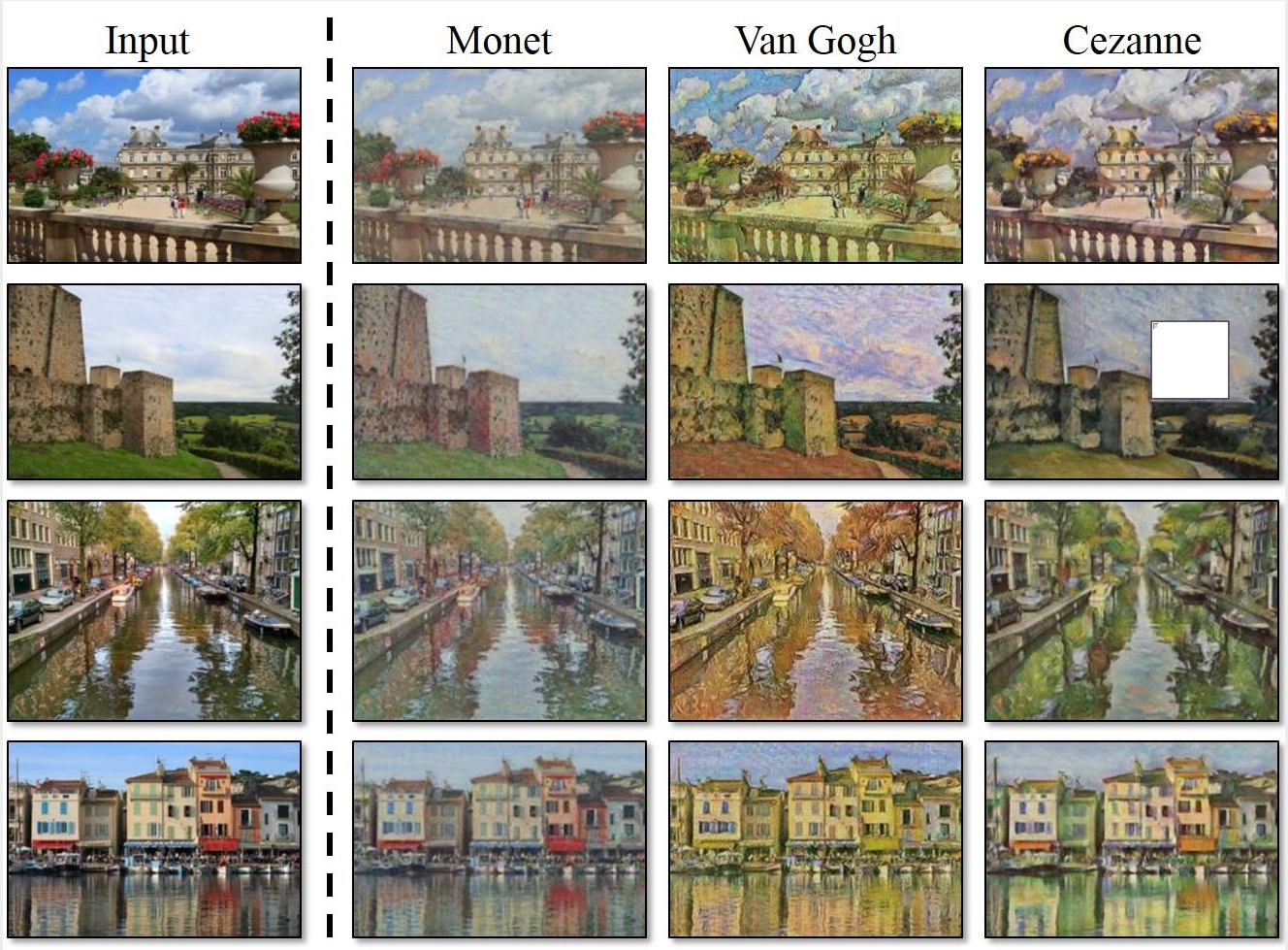


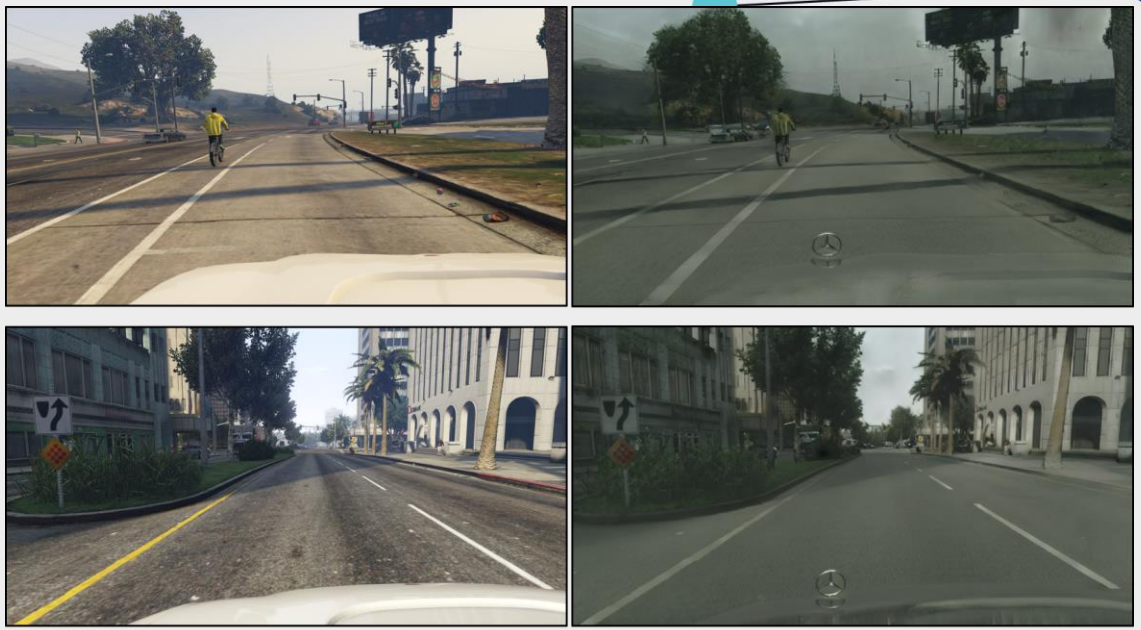
Сохраняет цвет!

$$\mathcal{L}_{identity}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(x) - x\|^2] + \mathbb{E}_{y \sim p_{\text{data}(y)}}[\|G(y) - y\|^2]$$

[13]CycleGAN

CycleGAN

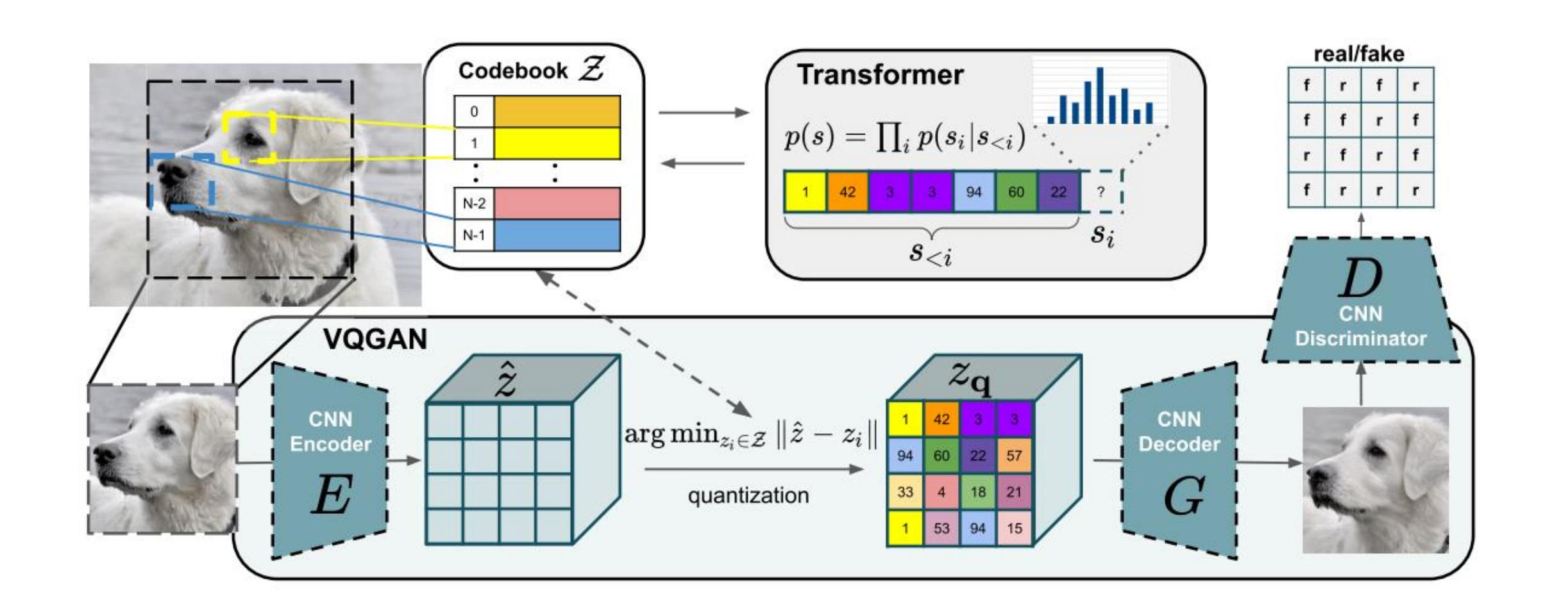




GTA → Cityscapes

[13]CycleGAN

VQGAN



[14] VQGAN

VQGAN

VQVAE Loss

$$\mathcal{L}_{VQ}(\mathcal{E}, \mathcal{G}, \mathcal{Z}) = \underbrace{\|x - \hat{x}\|^2}_{\text{reconstruction loss}} +$$

$$+\|sg[\mathcal{E}(x)] - z_q\|_2^2 + \underbrace{\|sg[z_q] - \mathcal{E}(x)\|_2^2}_{\text{commitment loss}}$$

VQGAN Loss

$$\mathcal{L}_{VQGAN} = \left[\mathcal{L}_{VQ}(\mathcal{E}, \mathcal{G}, \mathcal{Z}) + \lambda \mathcal{L}_{GAN}(\mathcal{E}, \mathcal{G}, \mathcal{Z}, D)\right]$$

$$\mathcal{L}_{GAN} = \mathcal{L}_{LPIPS} + \mathcal{L}_{D}$$

VQGAN

Figure 8. Samples from our class-conditional ImageNet model trained on 256×256 images.

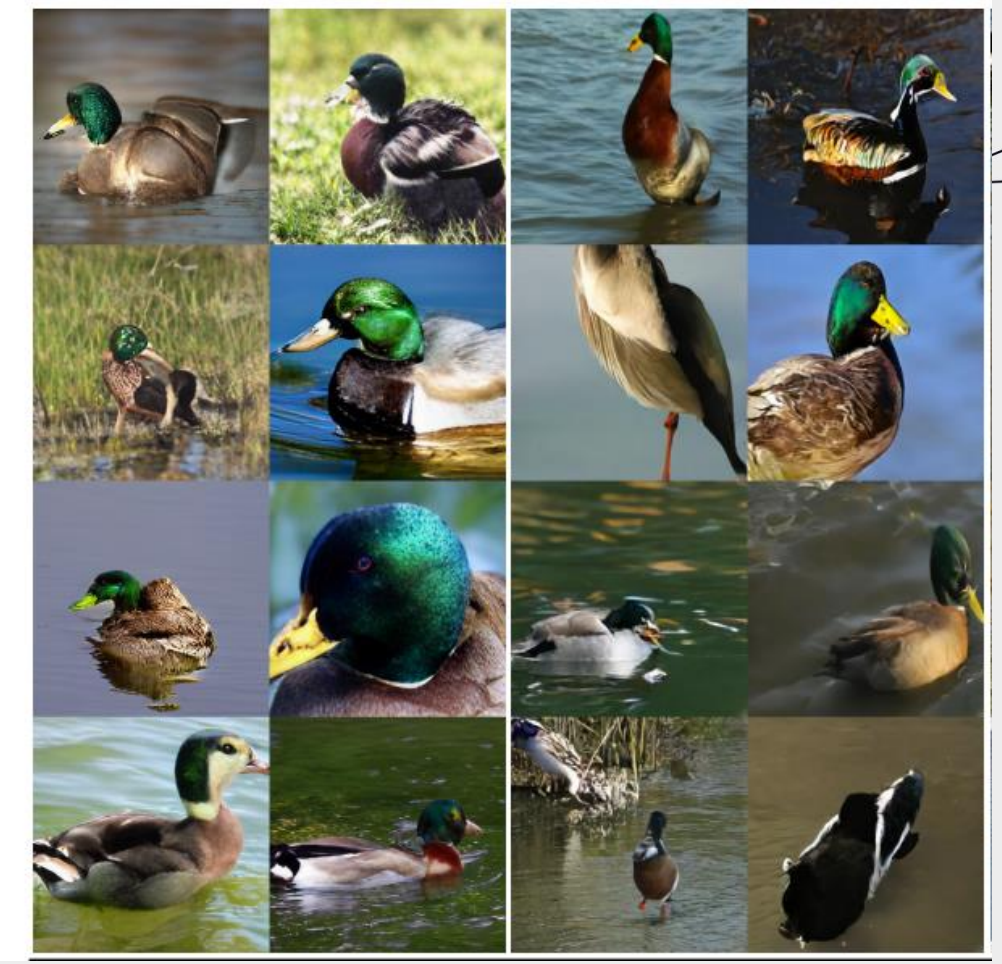
[14] VQGAN

VQGAN

Figure 12. Comparing reconstruction capabilities between VQVAEs and *VQGAN*s. Numbers in parentheses denote compression factor and codebook size. With the same compression factor and codebook size, *VQGAN*s produce more realistic reconstructions compared to blurry reconstructions of VQVAEs. This enables increased compression rates for *VQGAN* while retaining realistic reconstructions. See Sec. C.

[14] VQGAN

VQGAN

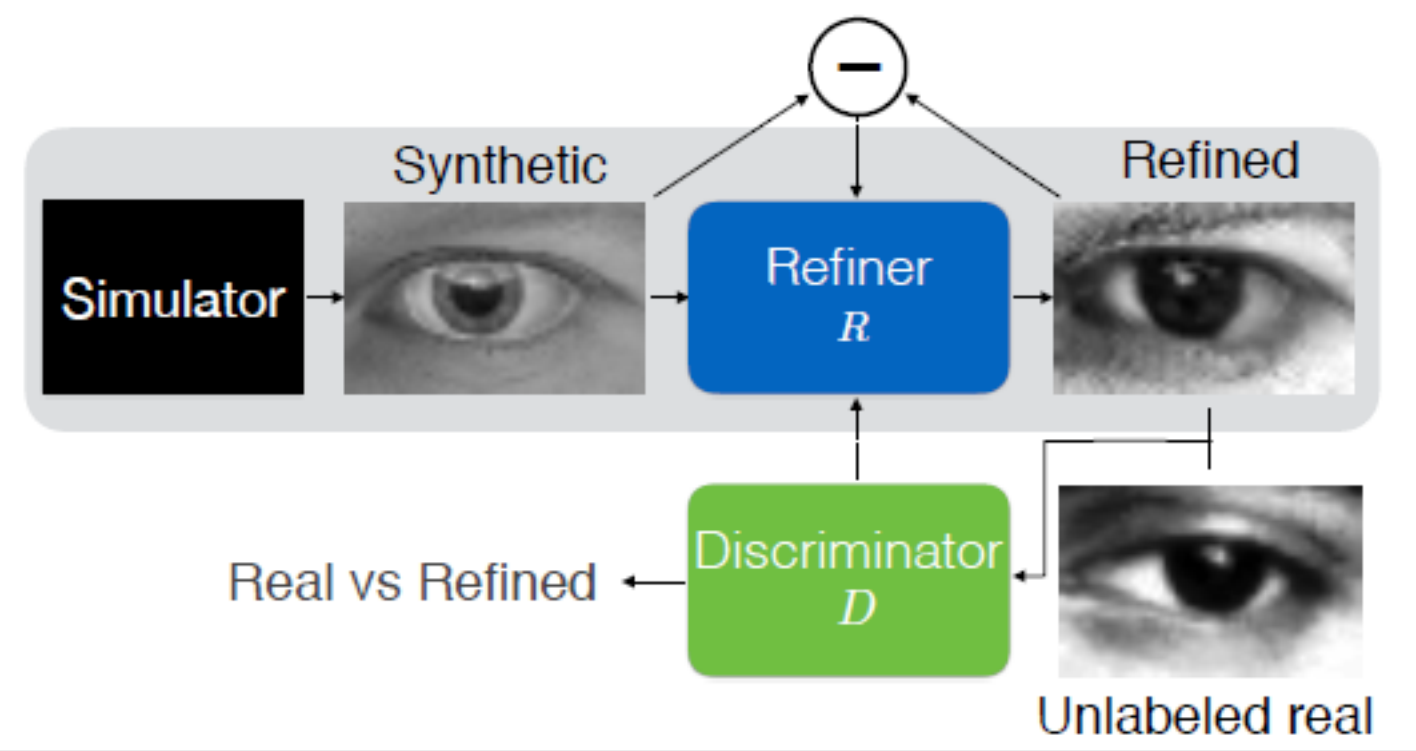


[14] VQGAN

SimGAN

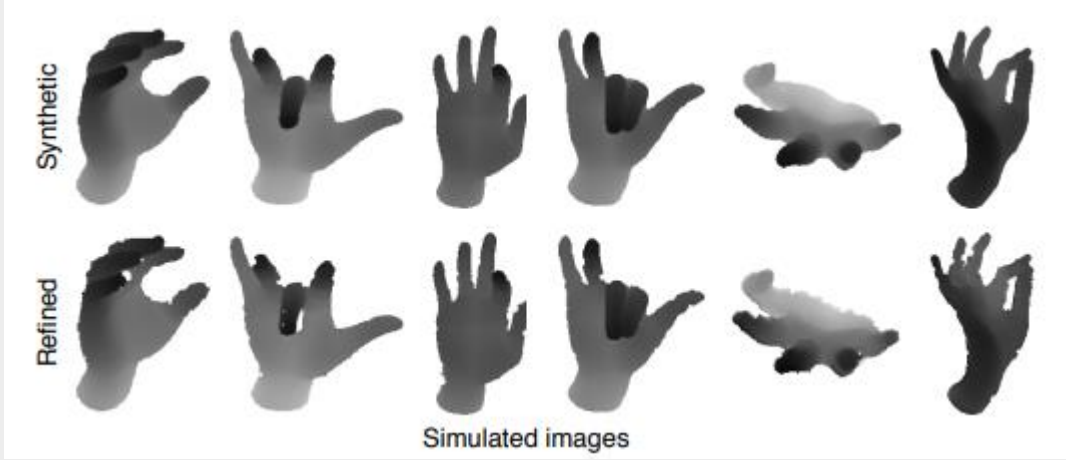
Проблема:

- Не хватает реальных данных для обучения - хотим использовать синтетически сгенерированные
- Сгенерированные данные могут отличаться от реальных и сетки, хорошо работающие на синтетике будут плохо показывать себя на реальных
- Возможное решение - SimGAN - добавление специального Refiner для получения более похожих на реальность данных.



[15] SimGAN

SimGAN - синтетика



[15] SimGAN

Bibliography

[1] Xiao Z., Kreis K., Vahdat A. Tackling the generative learning trilemma with denoising diffusion gans //arXiv preprint arXiv:2112.07804. – 2021.

[2] Goodfellow I. et al. Generative adversarial nets //Advances in neural information processing systems. – 2014. – T. 27.

[3] https://www.linkedin.com/pulse/addressing-mode-collapse-generative-adversarial-gans-andrew-jaramillo/

[4] Radford A., Metz L., Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks //arXiv preprint arXiv:1511.06434. – 2015.

[5] Arjovsky M., Chintala S., Bottou L. Wasserstein generative adversarial networks //International conference on machine learning. – PMLR, 2017. – С. 214-223.

Bibliography

[6] Karras T., Laine S., Aila T. A style-based generator architecture for generative adversarial networks //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2019. – С. 4401-4410..

[7] Karras T. et al. Analyzing and improving the image quality of stylegan //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2020. – С. 8110-8119.

[8] Abdal R. et al. Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows //ACM Transactions on Graphics (ToG). – 2021. – Т. 40. – №. 3. – С. 1-21.

[9] Abdal R., Qin Y., Wonka P. Image2stylegan: How to embed images into the stylegan latent space? //Proceedings of the IEEE/CVF international conference on computer vision. – 2019. – С. 4432-4441.

[10] Richardson E. et al. Encoding in style: a stylegan encoder for image-to-image translation //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2021. – С. 2287-2296.

Bibliography

[11] Isola P. et al. Image-to-image translation with conditional adversarial networks //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2017. – С. 1125-1134.

[12] Nagano Y., Kikuta Y. SRGAN for super-resolving low-resolution food images //Proceedings of the Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management. – 2018. – С. 33-37.

[13] Zhu J. Y. et al. Unpaired image-to-image translation using cycle-consistent adversarial networks //Proceedings of the IEEE international conference on computer vision. – 2017. – С. 2223-2232.

[14] Yu J. et al. Vector-quantized image modeling with improved vqgan //arXiv preprint arXiv:2110.04627. – 2021.

[15] Shrivastava A. et al. Learning from simulated and unsupervised images through adversarial training //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2017. – С. 2107-2116.

DLS

Коновалова Нина

Спасибо
за внимание!

@AfeliaN

konovalova.np@phystech.edu

Deep Learning School