Generative adversarial networks (GANs)

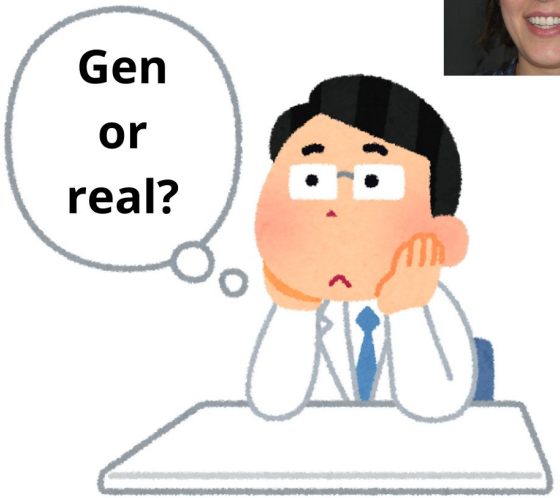
Real vs Generated

Real vs Generated

GANs
Main idea

Generator

Gen Real



Discriminator

GANs. Основная идея

Генератор: принимает на вход вектор из случайного шума $z \sim \mathcal{N}(0, I)$ и из него генерирует изображение $\mathcal{G}(z)$.

Дискриминатор: получает на вход реальные данные и сгенерированные и каждому сэмлпу ставит в соответствие вероятность принадлежности к реальным данным $\mathcal{D}(x)$.

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

P(“хорошая картинка”)

1 - P(“хорошая картинка”)

Идеальный дискриминатор

Зафиксируем генератор и будем обучать только дискриминатор:

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

.

Идеальный дискриминатор

Зафиксируем генератор и будем обучать только дискриминатор:

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

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

Идеальный дискриминатор

Зафиксируем генератор и будем обучать только дискриминатор:

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

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

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

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

Идеальный генератор

Зафиксируем дискриминатор и будем обучать только генератор

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{L}(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \left[\log \mathcal{D}(x)\right] + \mathbb{E}_{z \sim p_z(z)} \left[\log \left(1 - \mathcal{D}(\mathcal{G}(z))\right)\right] =$$

$$= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{data}(x)}{p_{data}(x) + p_{gen}(x)}] + \mathbb{E}_{z \sim p_z(z)}[\log \frac{p_{gen}(x)}{p_{data}(x) + p_{gen}(x)}] =$$

Идеальный генератор

Зафиксируем дискриминатор и будем обучать только генератор

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{L}(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \left[\log \mathcal{D}(x)\right] + \mathbb{E}_{z \sim p_z(z)} \left[\log \left(1 - \mathcal{D}(\mathcal{G}(z))\right)\right] =$$

$$= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{data}(x)}{p_{data}(x) + p_{gen}(x)}] + \mathbb{E}_{z \sim p_z(z)}[\log \frac{p_{gen}(x)}{p_{data}(x) + p_{gen}(x)}] =$$

$$= -\log 4 + D_{KL}(p_{data} \| \frac{p_{data}(x) + p_{gen}(x)}{2}) + D_{KL}(p_{gen} \| \frac{p_{data}(x) + p_{gen}(x)}{2})$$

$$D_{\text{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$$

Идеальный генератор

Зафиксируем дискриминатор и будем обучать только генератор

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{L}(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \left[\log \mathcal{D}(x)\right] + \mathbb{E}_{z \sim p_z(z)} \left[\log \left(1 - \mathcal{D}(\mathcal{G}(z))\right)\right] =$$

$$= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{data}(x)}{p_{data}(x) + p_{gen}(x)}] + \mathbb{E}_{z \sim p_z(z)}[\log \frac{p_{gen}(x)}{p_{data}(x) + p_{gen}(x)}] =$$

$$= -\log 4 + \boxed{D_{KL}(p_{data} \| \frac{p_{data}(x) + p_{gen}(x)}{2}) + D_{KL}(p_{gen} \| \frac{p_{data}(x) + p_{gen}(x)}{2})}$$

$$2 \cdot JSD(p \mid\mid q) \geq 0$$

$$D_{\text{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$$

Алгоритм обучения

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.

[Источник]

Проблемы обучения

Figure 10: More face generations from our Face DCGAN.

1. Есть «странные генерации»

2. Может возникнуть mode collapse

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

[Источник]

Исчезающие градиенты

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

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

P(“хорошая картинка”)

1 - P(“хорошая картинка”)

Исчезающие градиенты

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

P(“хорошая картинка”)

1 - P(“хорошая картинка”)

Для генератора

$$\mathbb{E}_{z \sim p_z(z)}[\log(1 - \mathcal{D}(\mathcal{G}(z)))]$$

Исчезающие градиенты

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

P(“хорошая картинка”)

1 - P(“хорошая картинка”)

Для генератора

$$\mathbb{E}_{z \sim p_z(z)}[\log(1 - \mathcal{D}(\mathcal{G}(z)))]$$

$$\mathcal{D}(\mathcal{G}(z)) \approx 0$$

Дискриминатор сильнее генератора

Исчезающие градиенты

Для генератора

$$\mathbb{E}_{z \sim p_z(z)}[\log(1 - \mathcal{D}(\mathcal{G}(z)))]$$

$$\mathcal{D}(\mathcal{G}(z)) \approx 0$$

Дискриминатор сильнее генератора

$$\log(1 - \mathcal{D}(\mathcal{G}(z))) \approx \log 1 = 0$$

Исчезающие градиенты

Для генератора

$$L_G = \mathbb{E}_{z \sim p_z(z)}[\log(1 - \mathcal{D}(\mathcal{G}(z)))]$$

$$\mathcal{D}(\mathcal{G}(z)) \approx 0$$

Дискриминатор сильнее генератора

$$\log(1 - \mathcal{D}(\mathcal{G}(z))) \approx \log 1 = 0$$

$$\frac{\partial L_G}{\partial \mathcal{G}} \approx 0$$

Исчезающие градиенты SOLUTION

В оригинальной статье предлагается решать проблему следующим образом:

1. Обучать дискриминатор не до сходимости, а в течение фиксированного числа шагов (несколько шагов).
2. Использовать более оптимальный лосс, который имеет тот же оптимум, , но лучшие градиенты:

Одинаковый минимум!

$$\mathbb{E}_{z \sim p_z(z)} - \log \mathcal{D}(\mathcal{G}(z)))$$

$$\arg\min_{\theta} \mathbb{E}_{z \sim p(z)} \log\left[1 - D_\phi(G_\theta(z))\right] = \arg\min_{\theta} \mathbb{E}_{z \sim p(z)} - \log D_\phi(G_\theta(z))$$

Результаты из статьи

[Источник]

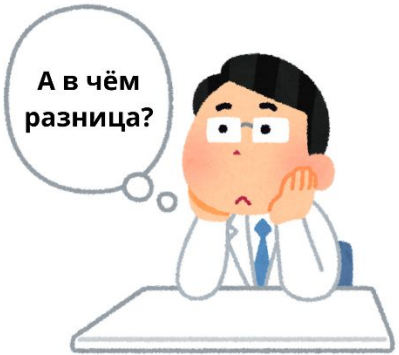
Результаты из статьи

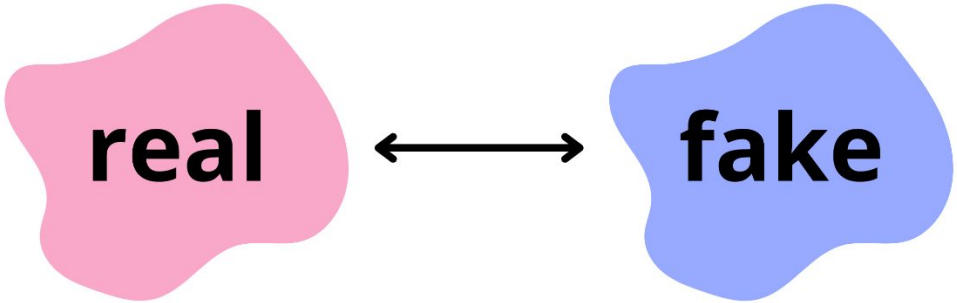
Figure 3: Digits obtained by linearly interpolating between coordinates in z space of the full model.

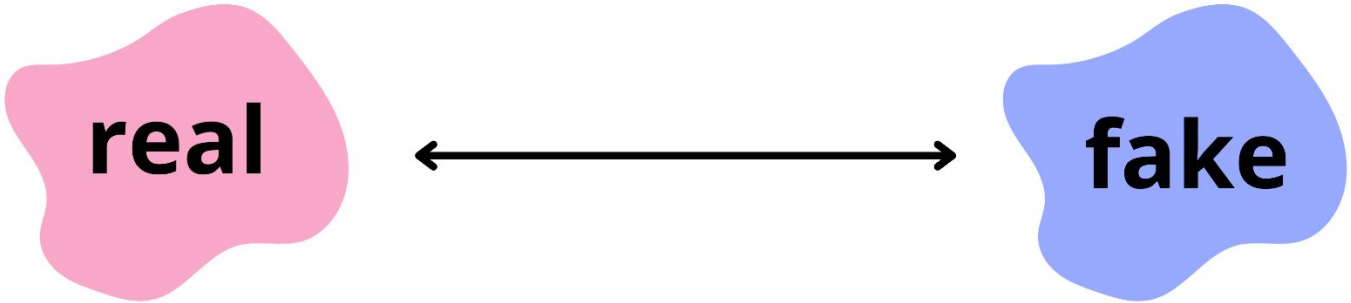
[Источник]

WGAN

WGAN







WGAN

Распределение фейков и действительных данных не пересекается

↓

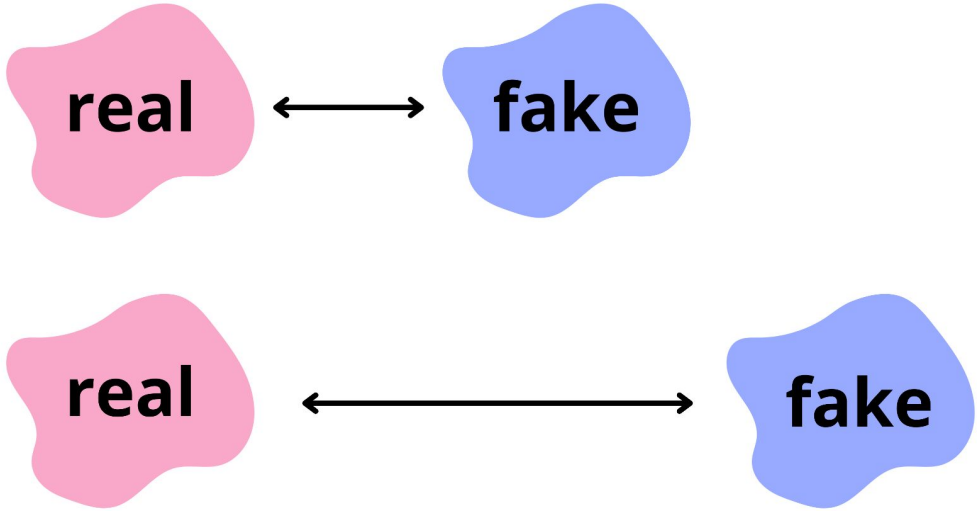
Дискриминатор умеет точно разделять фейки и реальные картинки

↓

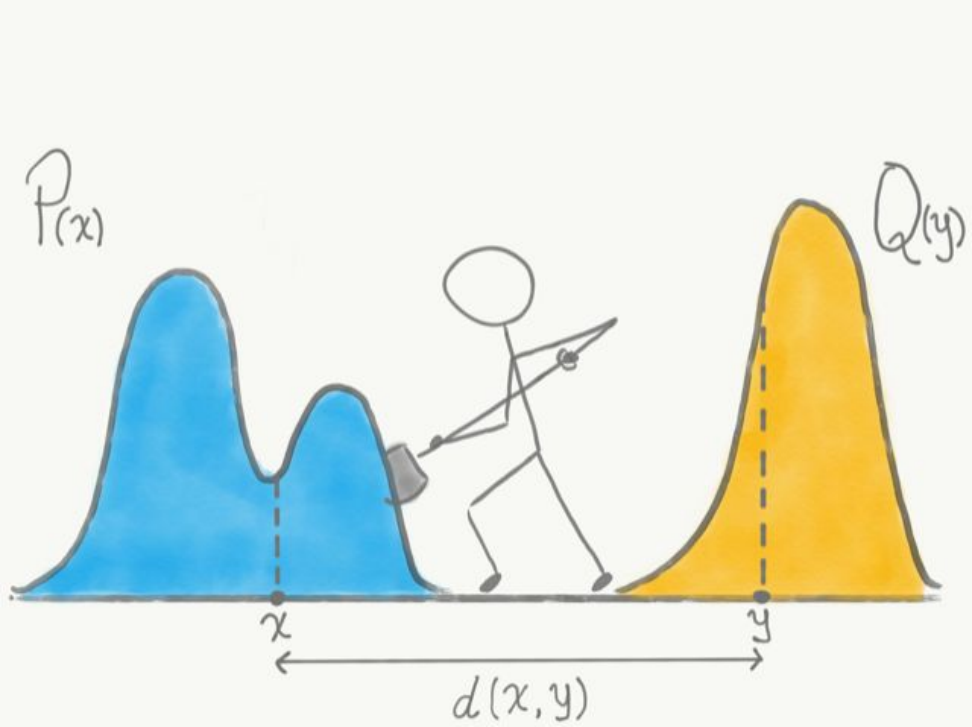
Градиент почти нулевой

↓

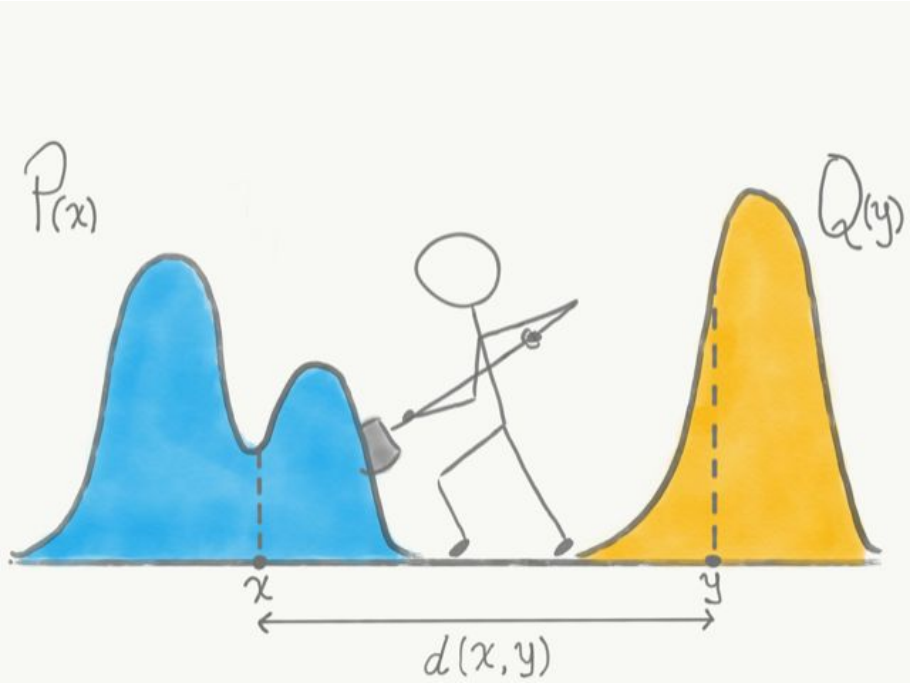
Хотим: уметь различать близко расположенные и далеко расположенные распределения



WGAN



WGAN

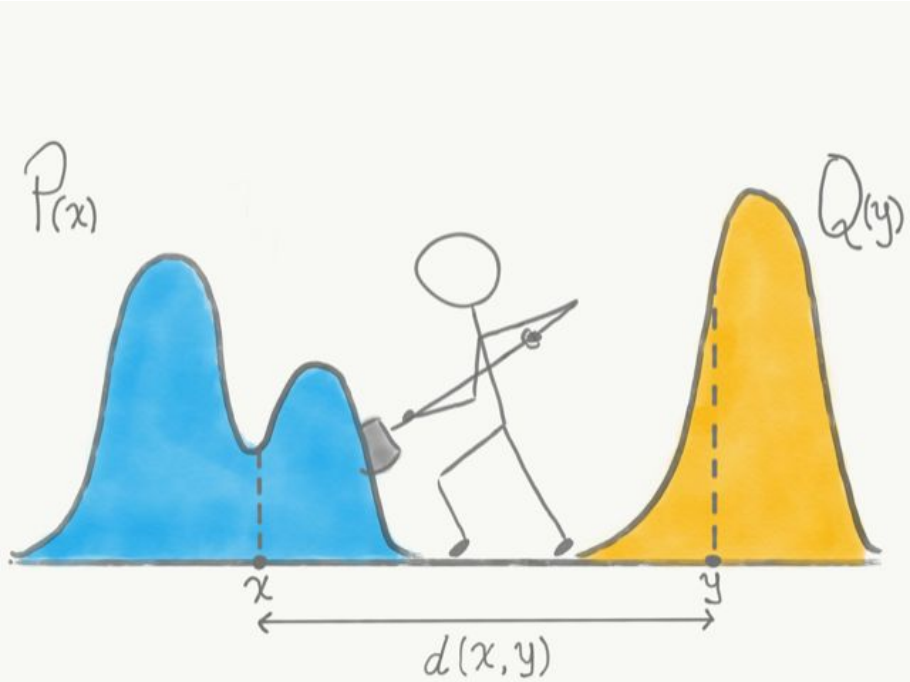


$$W_1(p_{\text{data}}, p_{gen}) = \inf_{\gamma \in \Pi(p_{\text{data}}, p_{gen})} \mathbb{E}_{(x,y)\sim\gamma}[|x-y|]$$

$\Pi(p_{\text{data}}, p_{gen})$ множество всех совместных распределений $\gamma(x, y)$

маргинальные распределения которых соответствуют $p_{\text{data}}(x)$ и $p_{gen}(y)$

WGAN



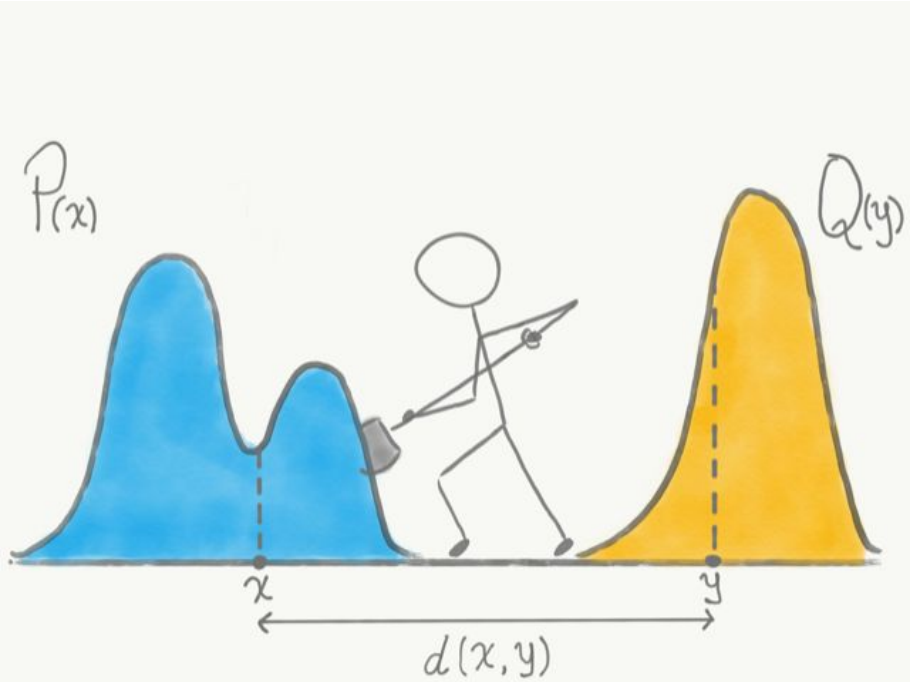
$$W_1(p_{\text{data}}, p_{gen}) = \inf_{\gamma \in \Pi(p_{\text{data}}, p_{gen})} \mathbb{E}_{(x,y)\sim\gamma}[|x-y|]$$

упростим!

$$W_1(p_{\text{data}}, p_{gen}) = \sup_{\|f\|_L \leq 1} \left(\mathbb{E}_{x\sim p_{\text{data}}}[f(x)] - \mathbb{E}_{y\sim p_{gen}}[f(y)]\right)$$

дуальность Канторовича-Рубинштейна

WGAN



дуальность Канторовича-Рубинштейна

$$W_1(p_{\text{data}}, p_{gen}) = \sup_{\|f\|_L \leq 1} \left(\mathbb{E}_{x \sim p_{\text{data}}}[f(x)] - \mathbb{E}_{y \sim p_{gen}}[f(y)]\right)$$

$$|f(x) - f(y)| \leq |x - y|$$

функция, удовлетворяющая условию 1-Липшица

WGAN

GAN

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

дискриминатор – выдаёт вероятности

VS

WGAN

GAN

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

дискриминатор – выдаёт вероятности

VS

WGAN

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

критик выдает непрерывные значения

WGAN

GAN

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

дискриминатор – выдаёт вероятности

WGAN

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

критик выдает непрерывные значения

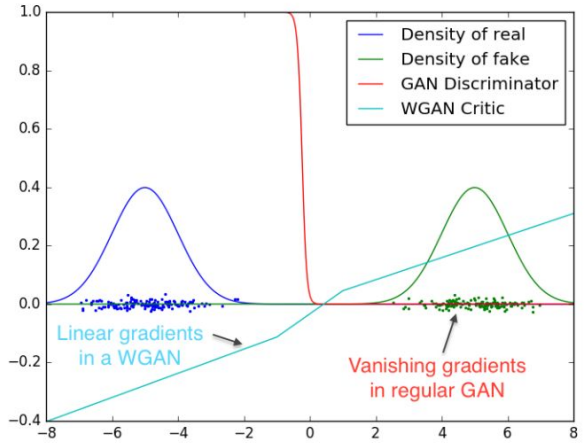


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

[Источник]

WGAN

WGAN

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

как достичь условия Липшица?

$$|f(x) - f(y)| \leq |x - y|$$

WGAN

WGAN

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

weight clipping!

WGAN

pros

✅ **Более стабильность обучения**

✅ **Меньше проблем с mode collapse**

✅ **Интерпретируемость функции потерь**: значение функции потерь WGAN напрямую коррелирует с расстоянием между распределениями, что делает её более информативной для оценки прогресса обучения.

cons

❌ **Вычислительная сложность**: Обучение критика требует больше итераций (обычно 5–10 шагов критика на 1 шаг генератора), что увеличивает время обучения.

❌ **Реализация условия Липшица**: Weight clipping в оригинальном WGAN ограничивает выразительность критика.

WGAN-GP

$$|\nabla_x \mathcal{D}(x)|_2 \leq 1$$

$$L = \mathbb{E}_{x \sim p_{\text{data}}}[\mathcal{D}(x)] - \mathbb{E}_{z \sim p_z(z)}[\mathcal{D}(\mathcal{G}(z))] + \lambda \mathbb{E}_{\hat{x} \sim p_{\hat{x}}} \left[(|\nabla_{\hat{x}} \mathcal{D}(\hat{x})|_2 - 1)^2 \right]$$

WGAN-GP

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

while θ has not converged **do**
  for $t = 1, \ldots, n_{\text{critic}}$ **do**
    for $i = 1, \ldots, m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

WGAN-GP

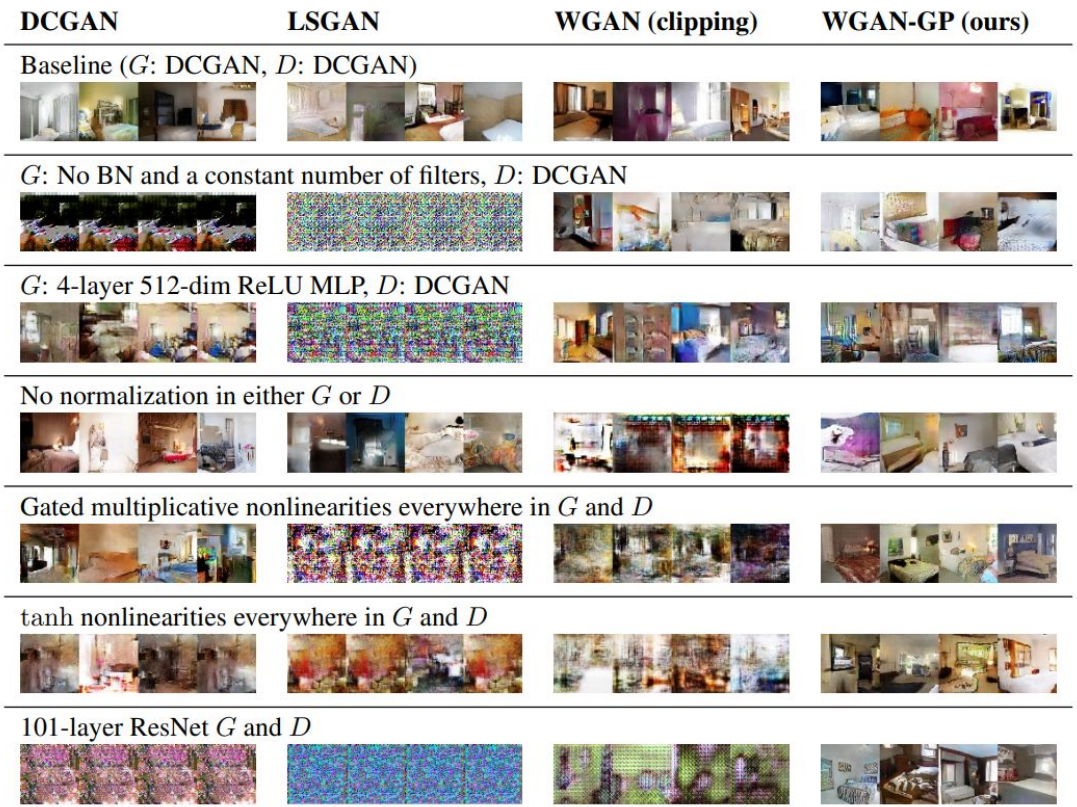
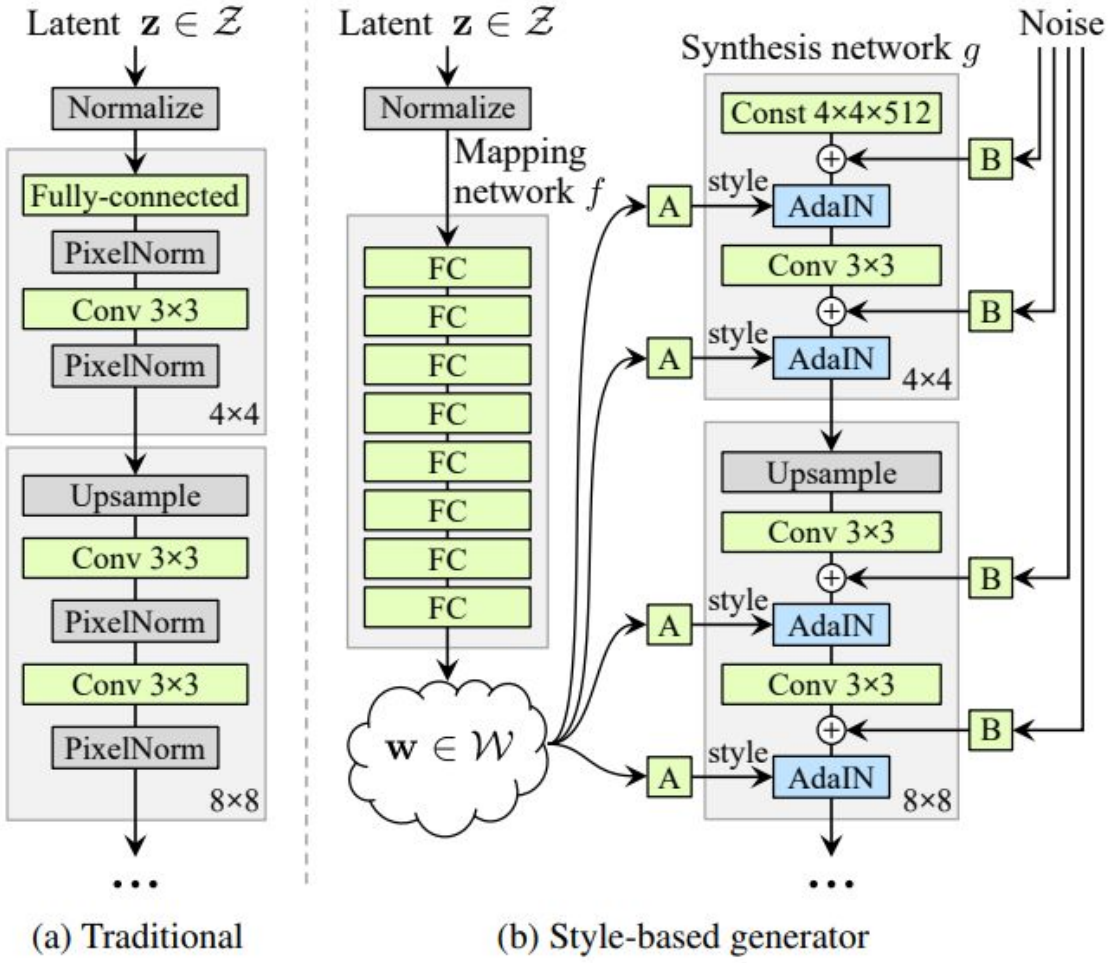


Figure 2: Different GAN architectures trained with different methods. We only succeeded in training every architecture with a shared set of hyperparameters using WGAN-GP.

Применение

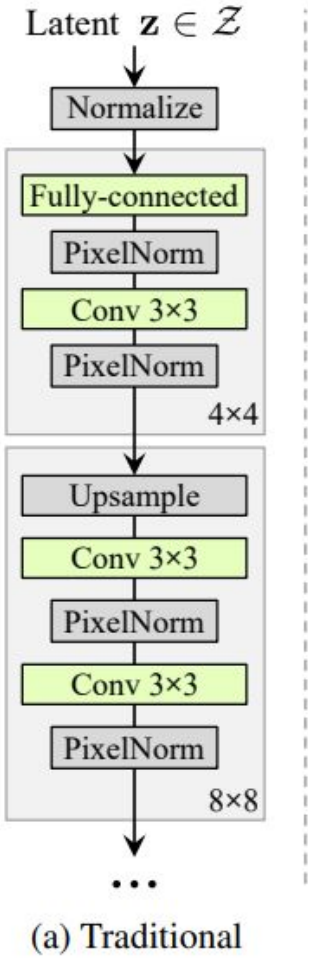
StyleGAN



(a) Traditional

(b) Style-based generator

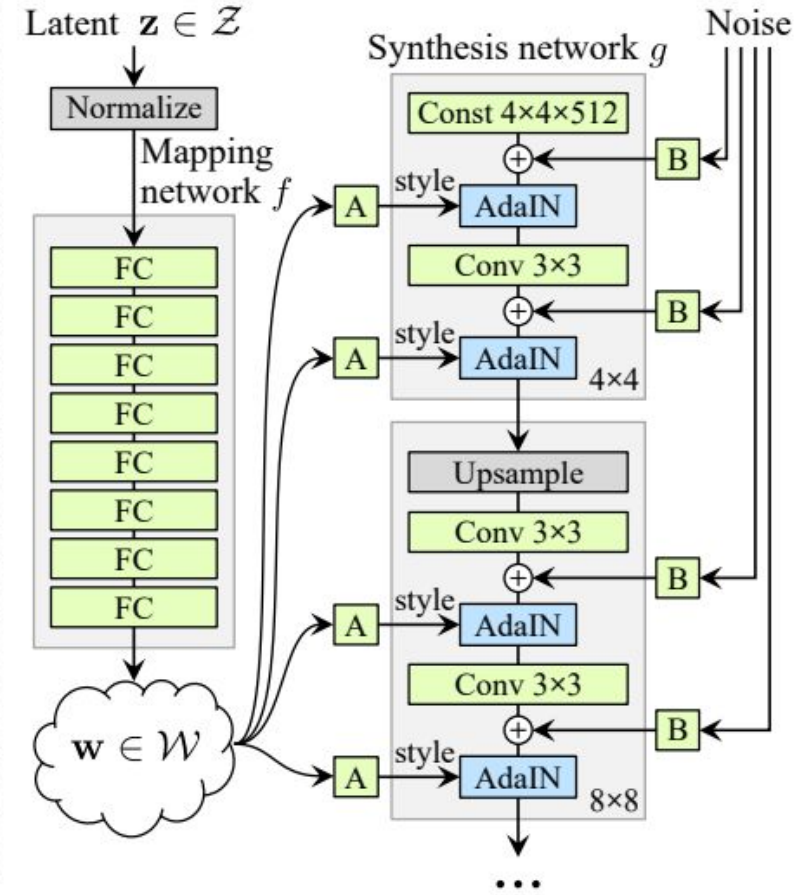
Usual GAN



(a) Traditional

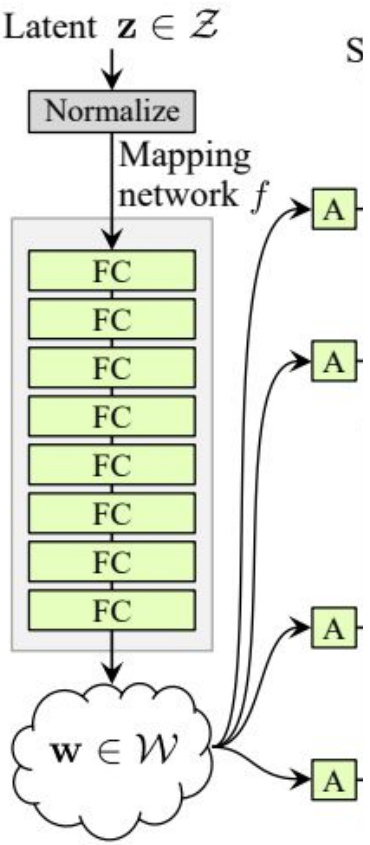
StyleGAN

- mapping network
- generator

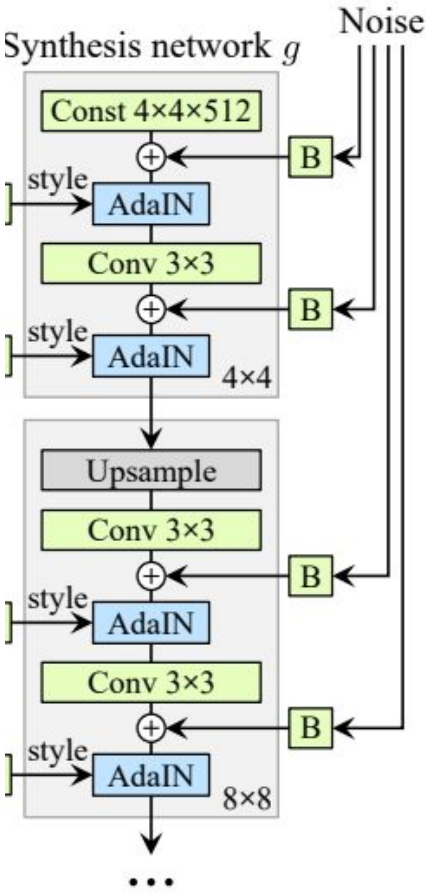


(b) Style-based generator

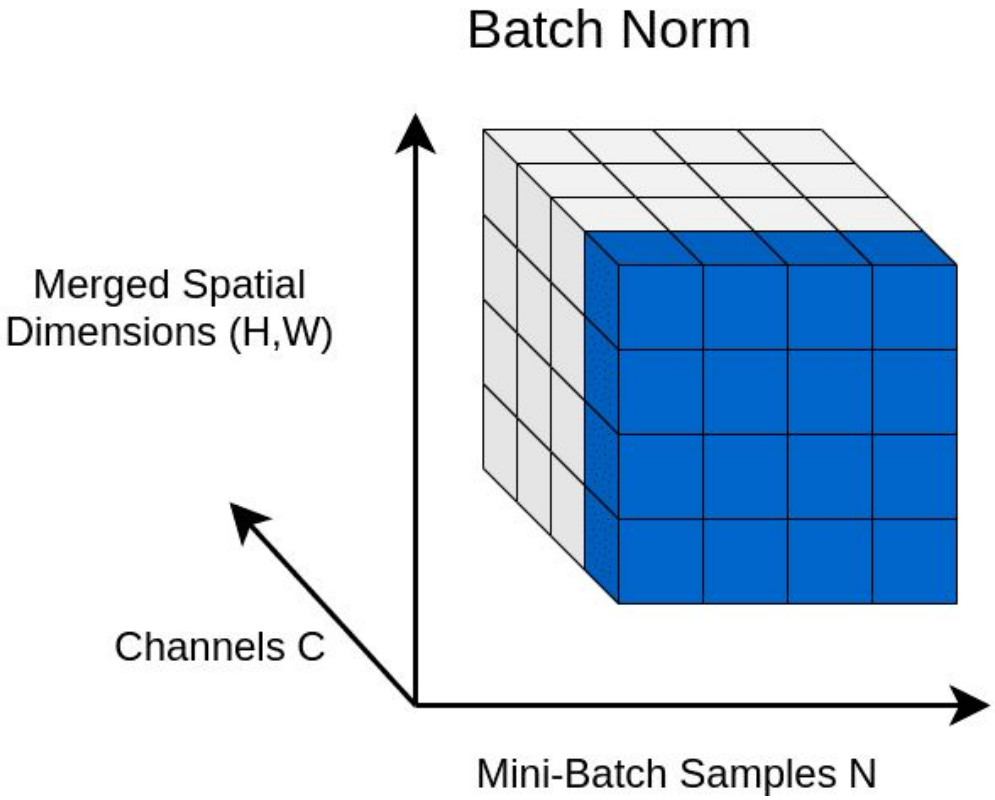
Mapping network



Generator


Synthesis network g
Noise
Const 4×4×512
B
style
AdaIN
Conv 3×3
B
style
AdaIN
4×4
Upsample
Conv 3×3
B
style
AdaIN
Conv 3×3
B
style
AdaIN
8×8
...

Batch Norm

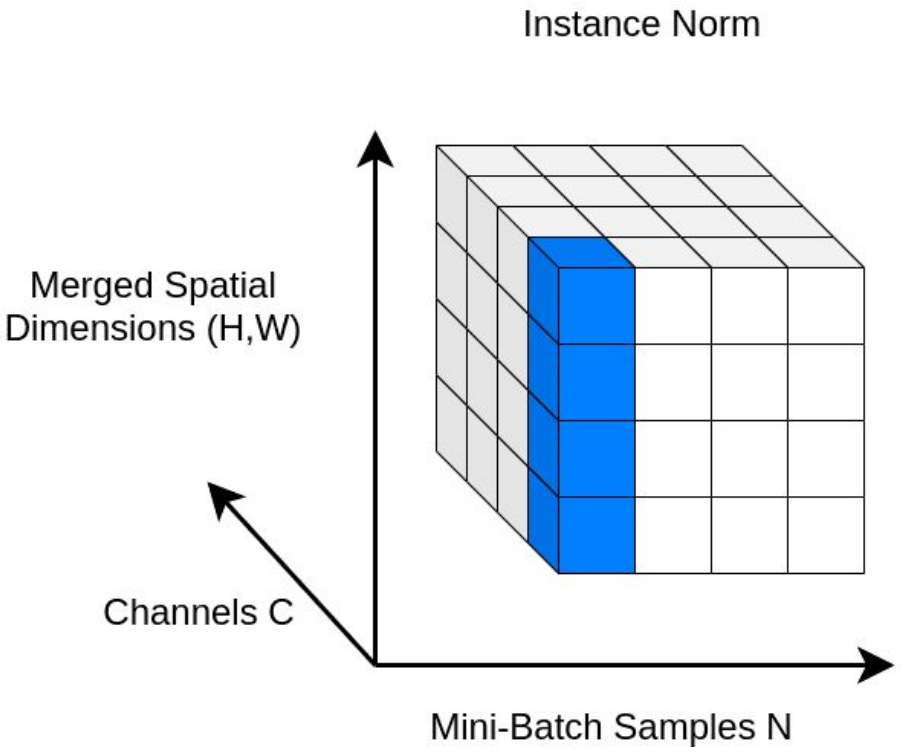


$$\mathrm{BN}(x) = \gamma \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta$$

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

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

Instance Norm



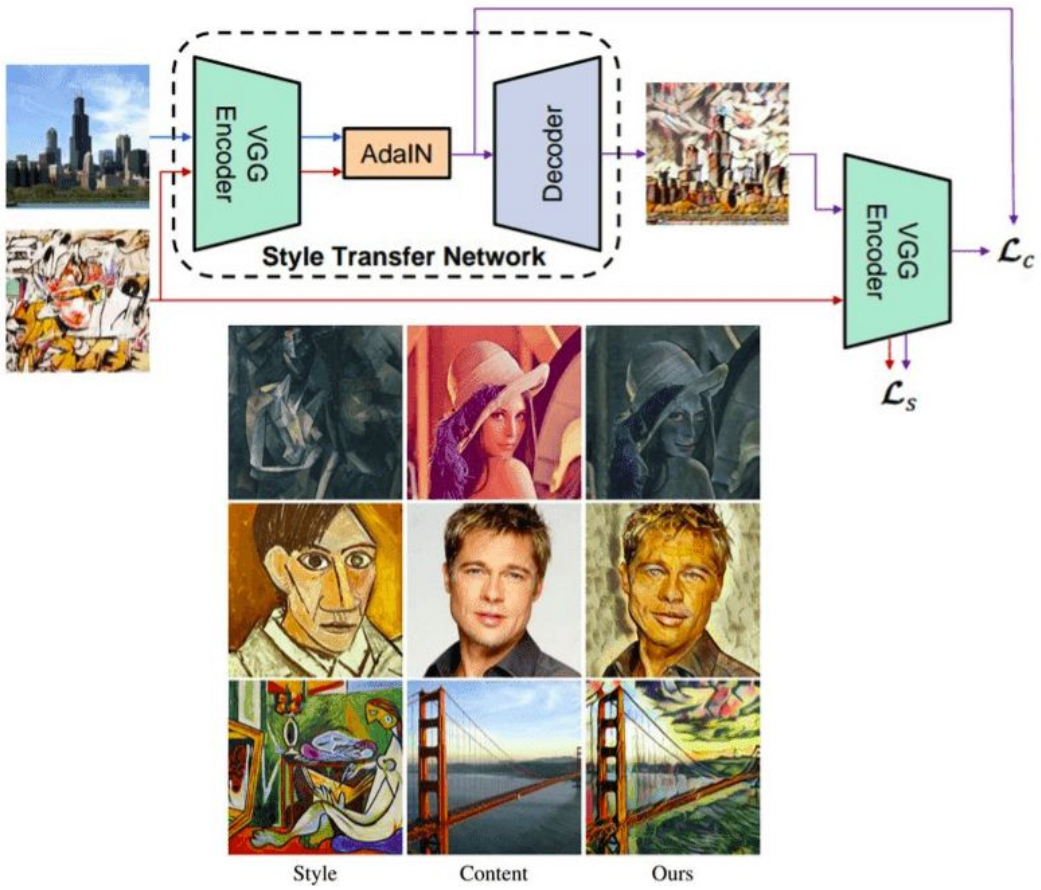
$$\mathrm{IN}(x) = \gamma\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \beta$$

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

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

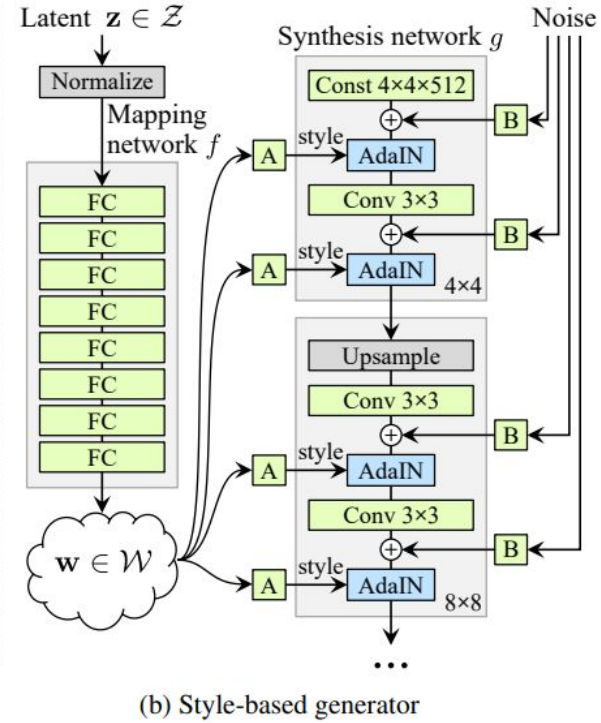
AdaIn

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

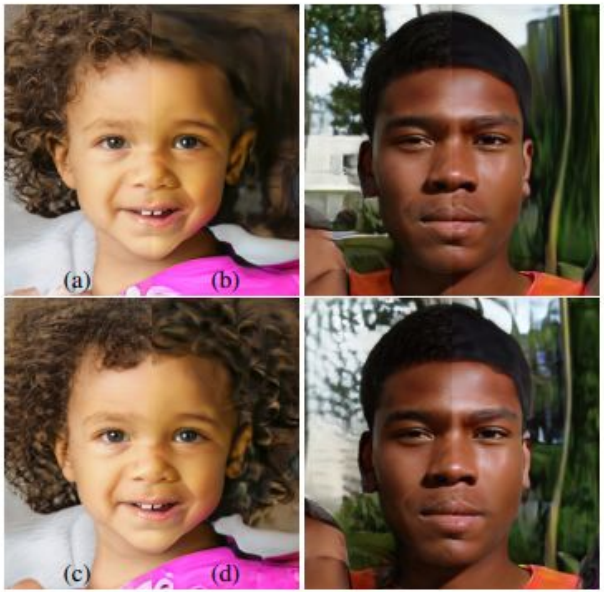


[Источник]

StyleGAN



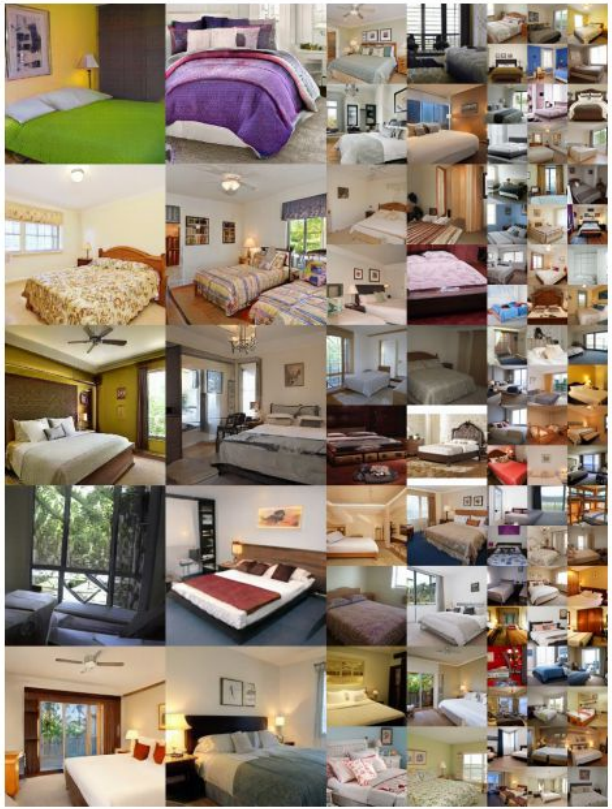
(b) Style-based generator

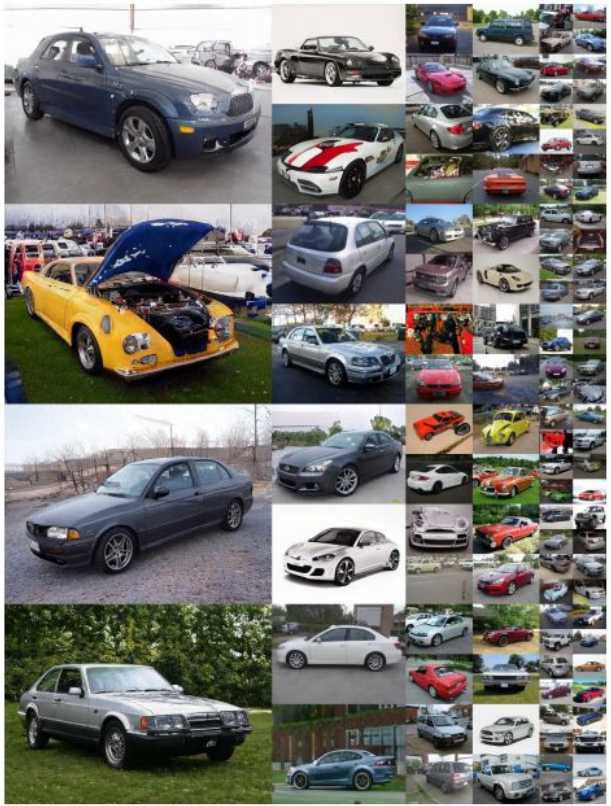


Влияние шума на генерацию. a) шум, добавленный во все слои, b) без добавления шума c) шум на второй половине блоков 64x64 -- 1024x1024, d) шума на первой половине блоков 4x4-32x32

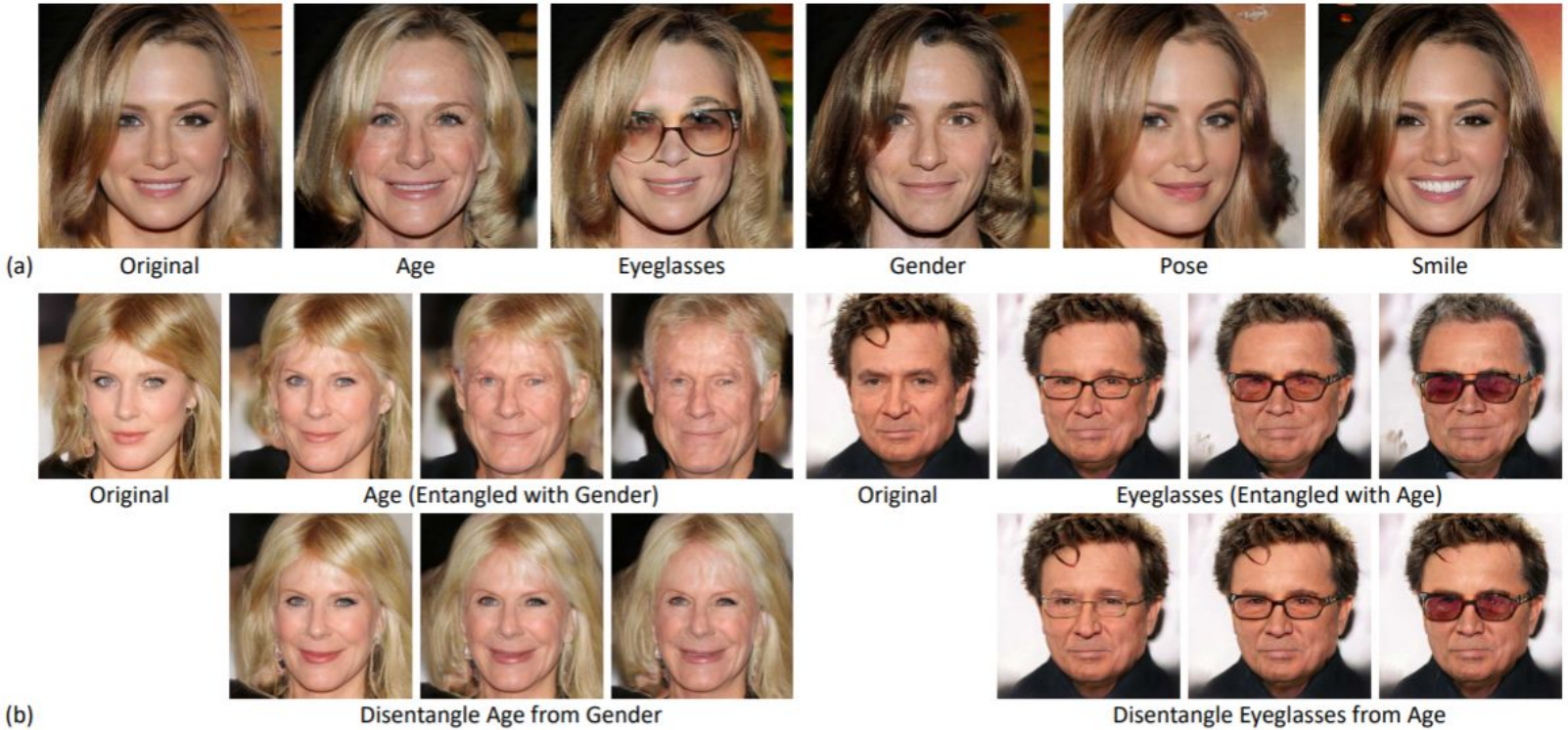
StyleGAN results

StyleGAN results





Распутанность



Распутанность

Fig. 13. **Manipulating real faces with GAN inversion**. Given a target image, we first invert it by optimizing the latent code and then manipulate the inverted code with InterFaceGAN. StyleGAN [3] is used as the generative model.

Fig. 14. **Manipulating real faces with encoder-decoder generative model**, LIA [51].

VQGAN

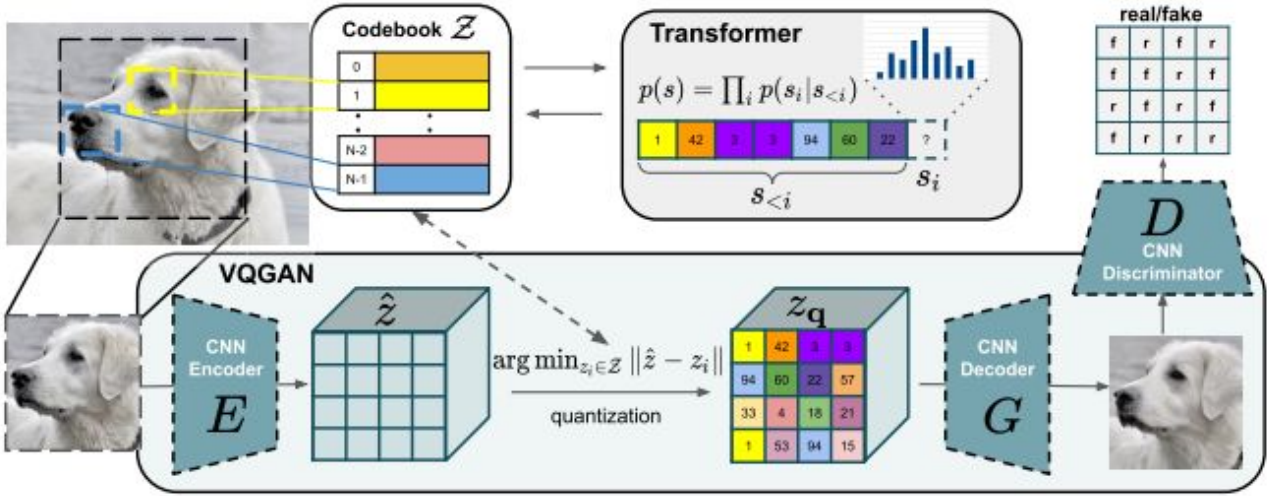


Figure 2. Our approach uses a convolutional *VQGAN* to learn a codebook of context-rich visual parts, whose composition is subsequently modeled with an autoregressive transformer architecture. A discrete codebook provides the interface between these architectures and a patch-based discriminator enables strong compression while retaining high perceptual quality. This method introduces the efficiency of convolutional approaches to transformer based high resolution image synthesis.

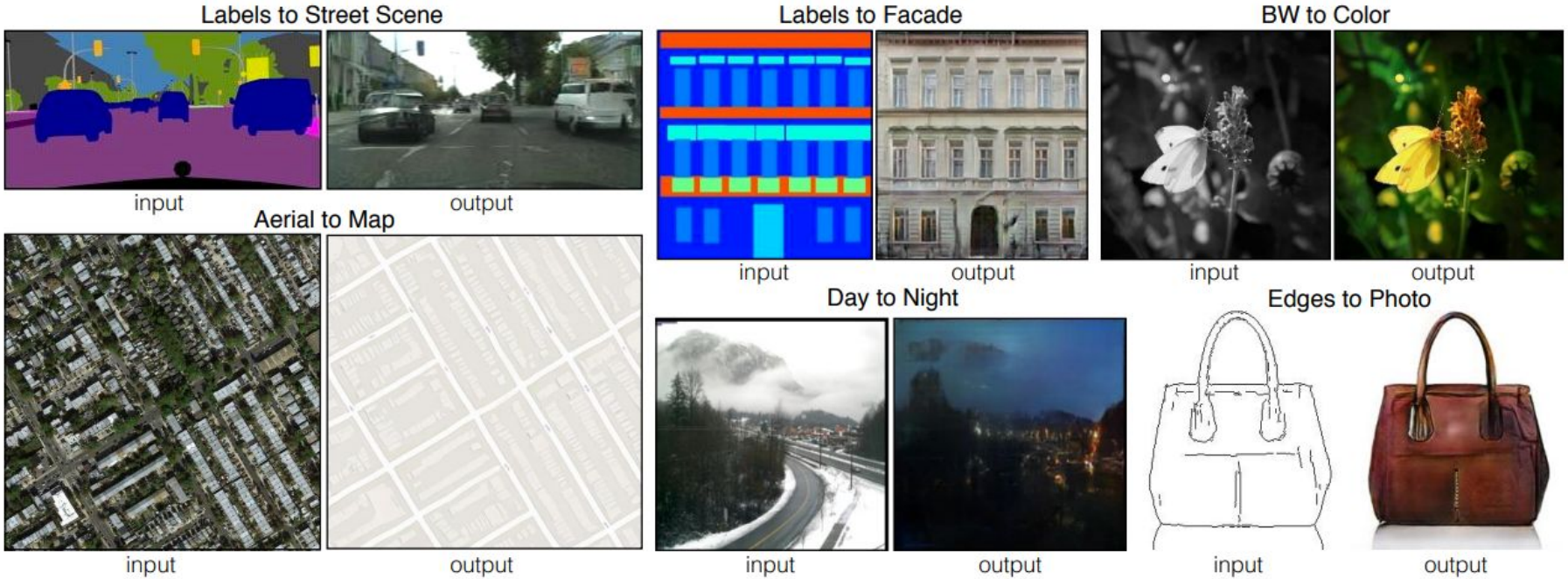
[Источник]

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.

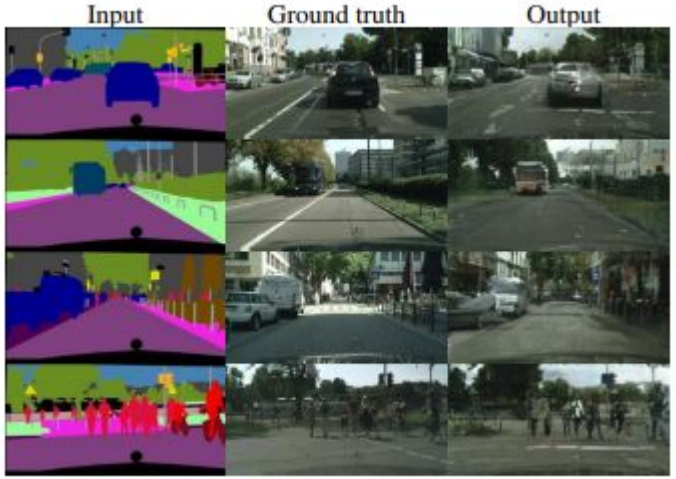
[Источник]

Pix2pix

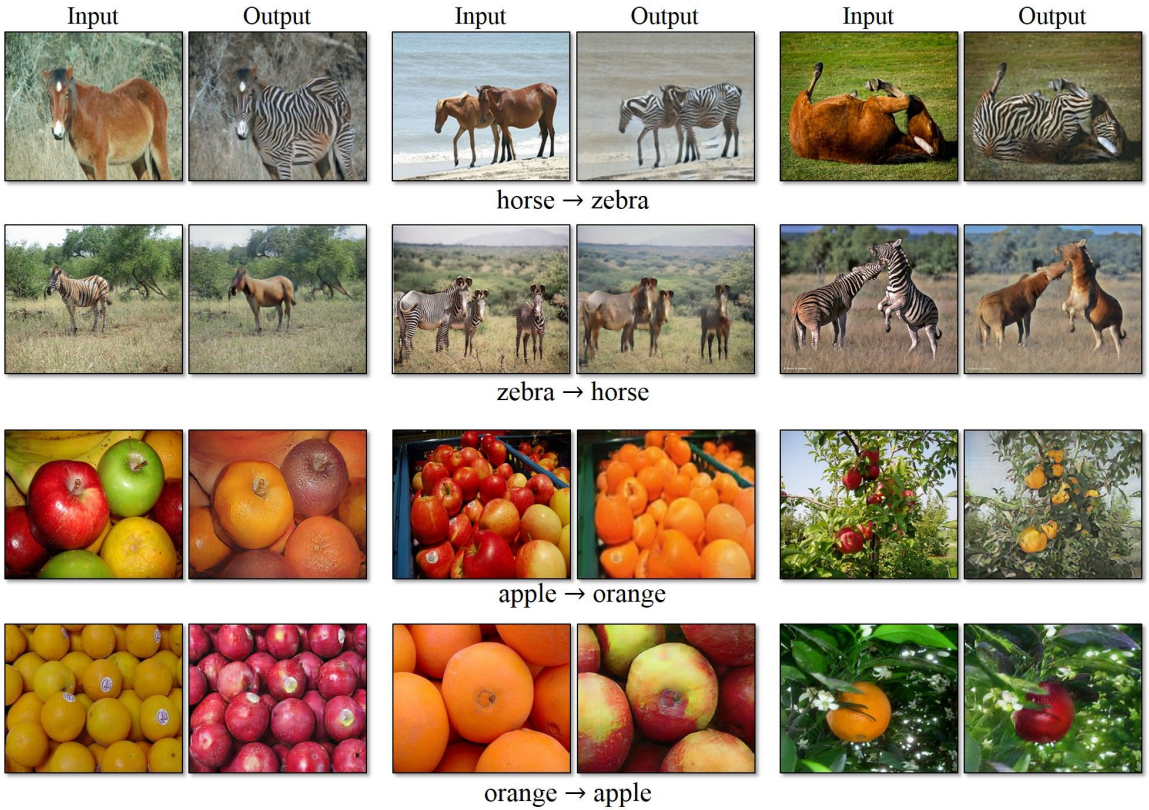

Labels to Street Scene
input
output
Labels to Facade
input
output
BW to Color
input
output
Aerial to Map
input
output
Day to Night
input
output
Edges to Photo
input
output

Pix2pix

Figure 2: Training a conditional GAN to map edges→photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.



CycleGAN


Input
Output
Input
Output
Input
Output
horse → zebra
zebra → horse
apple → orange
orange → apple

CycleGAN

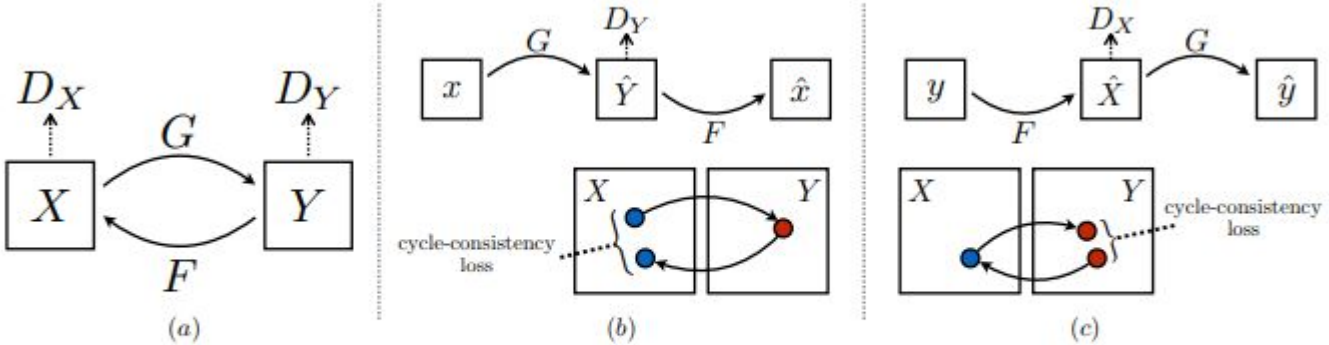
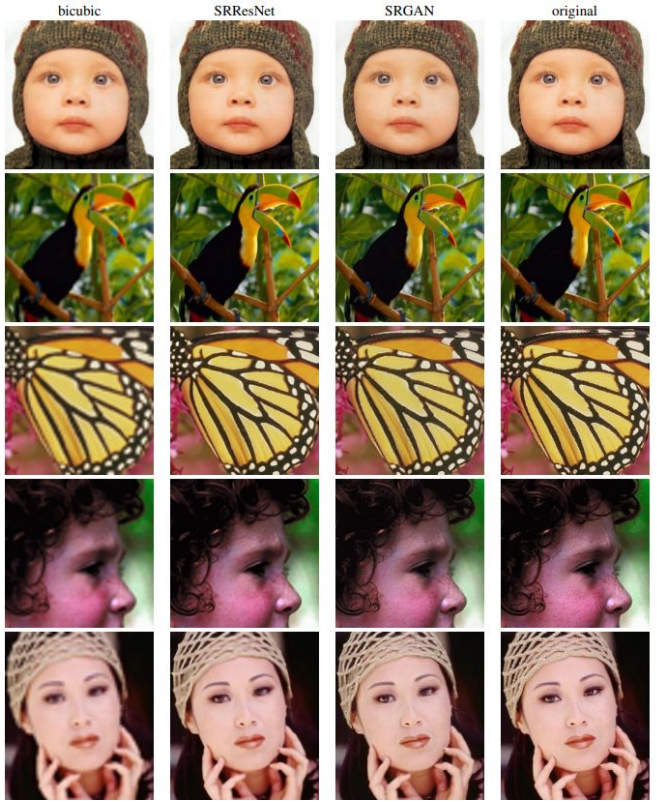


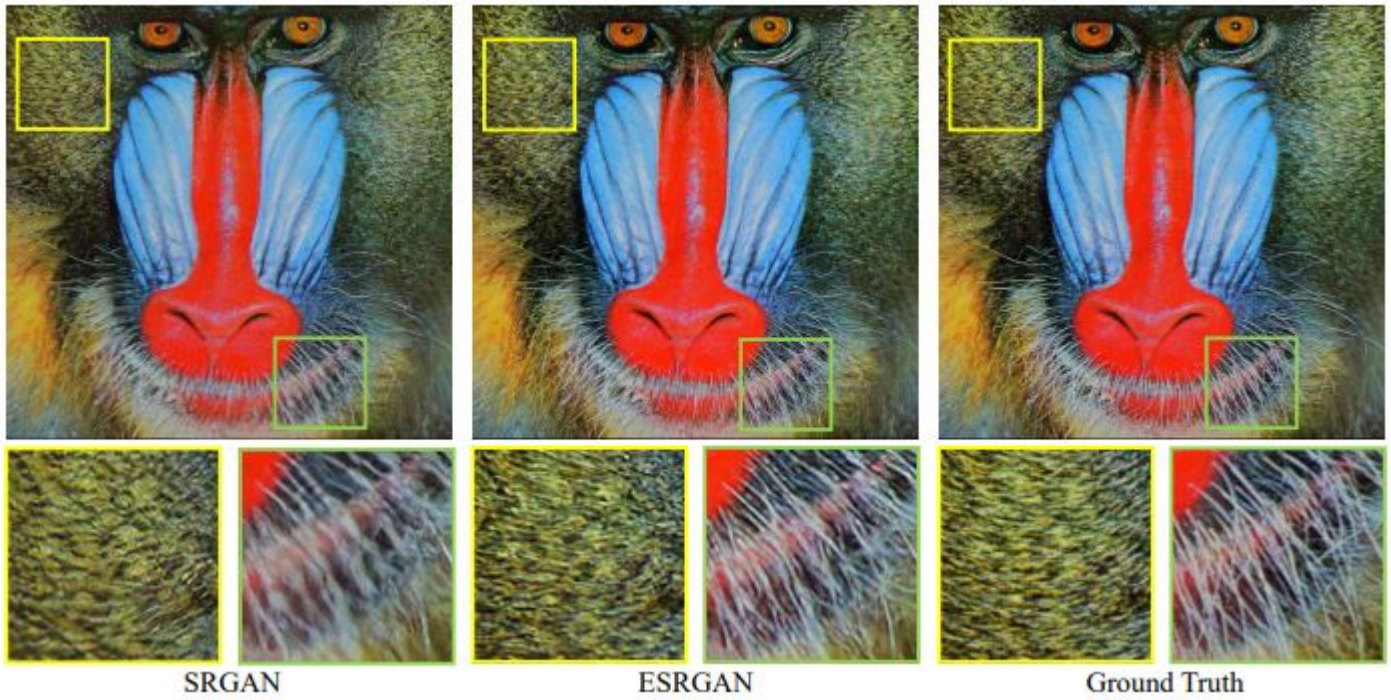
Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

Superresolution GAN - SRGAN

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



ESRGAN



Pixel2style2pixel