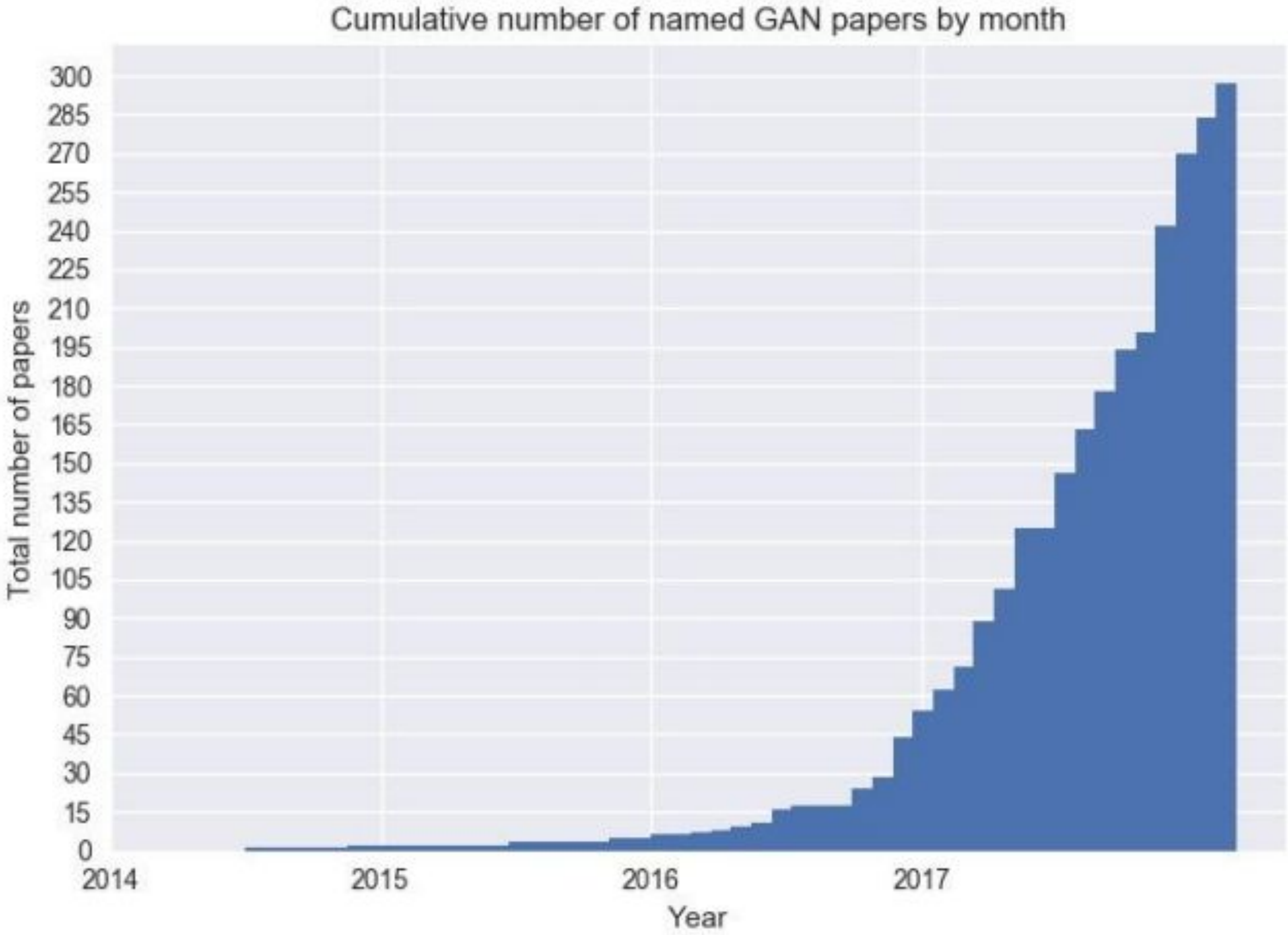
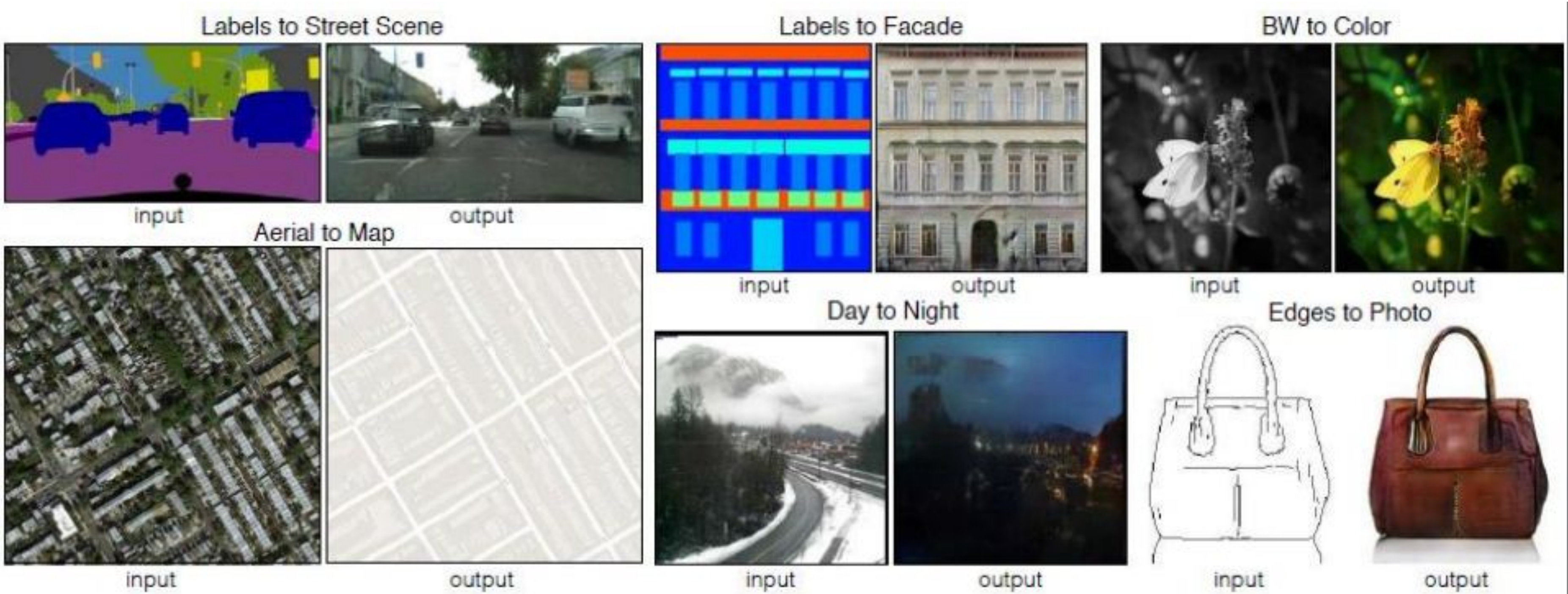
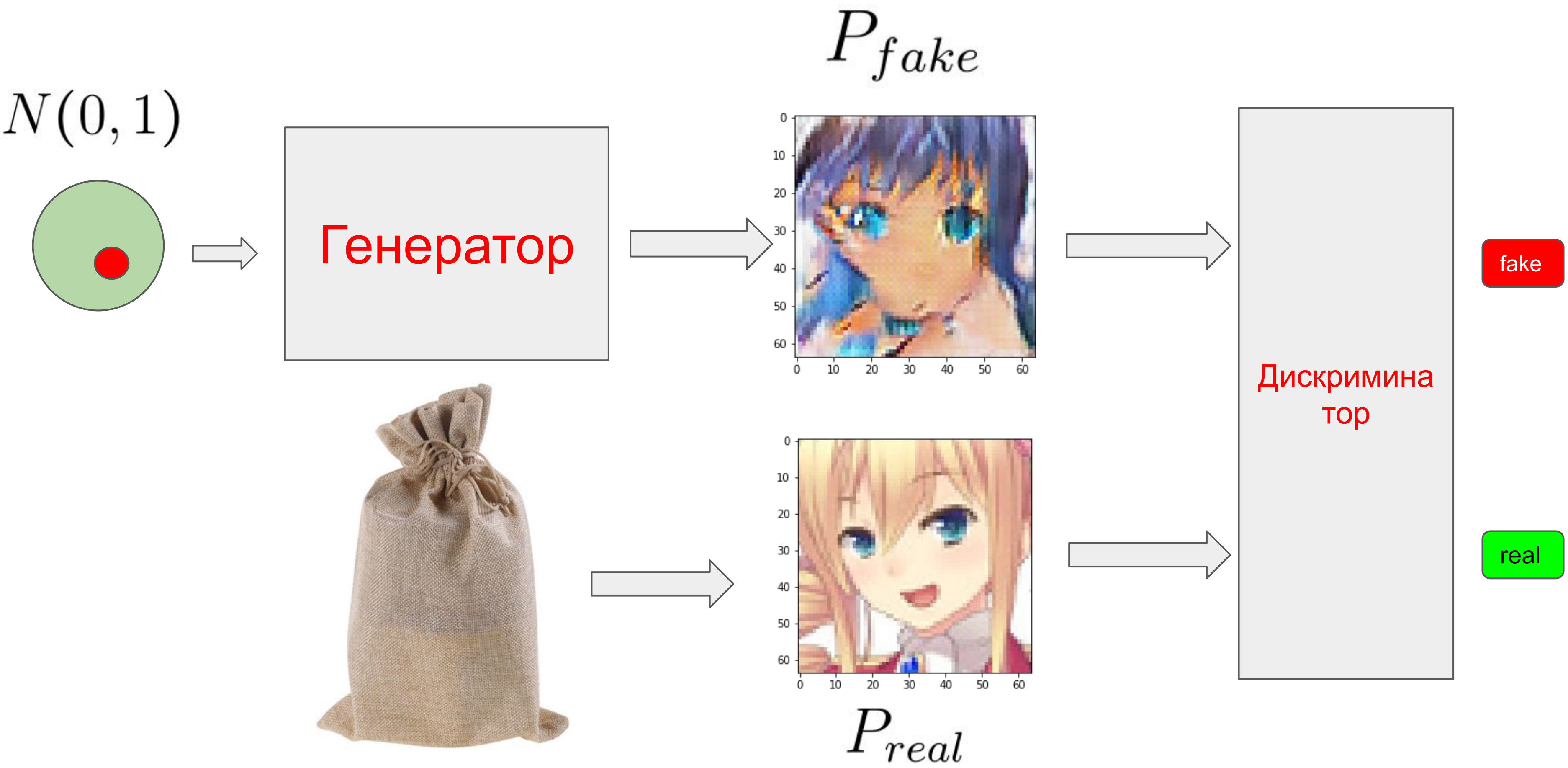
GAN - Generative Adversarial Network

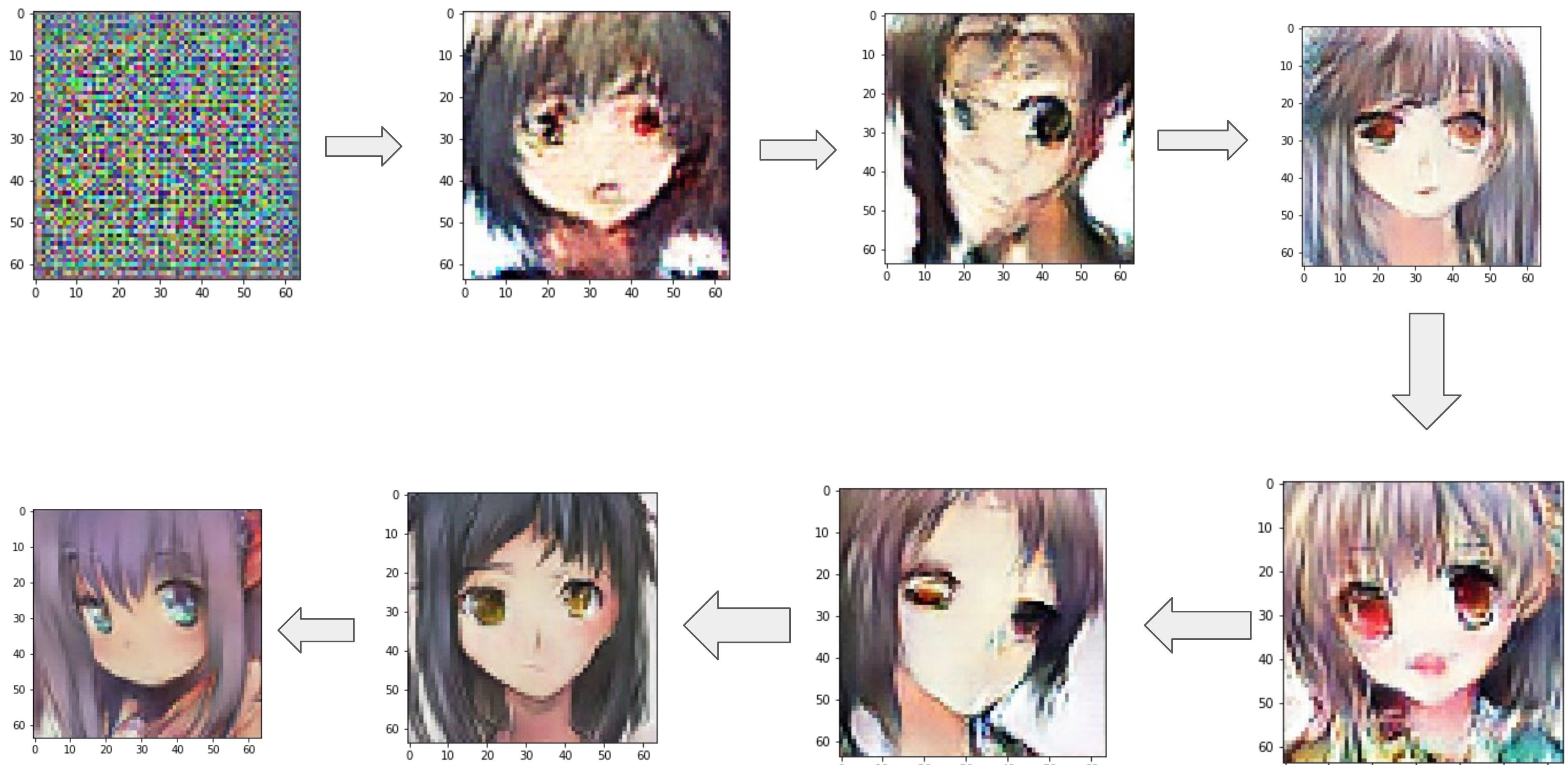

Cumulative number of named GAN papers by month
Total number of papers
0
15
30
45
60
75
90
105
120
135
150
165
180
195
210
225
240
255
270
285
300
2014
2015
2016
2017
Year



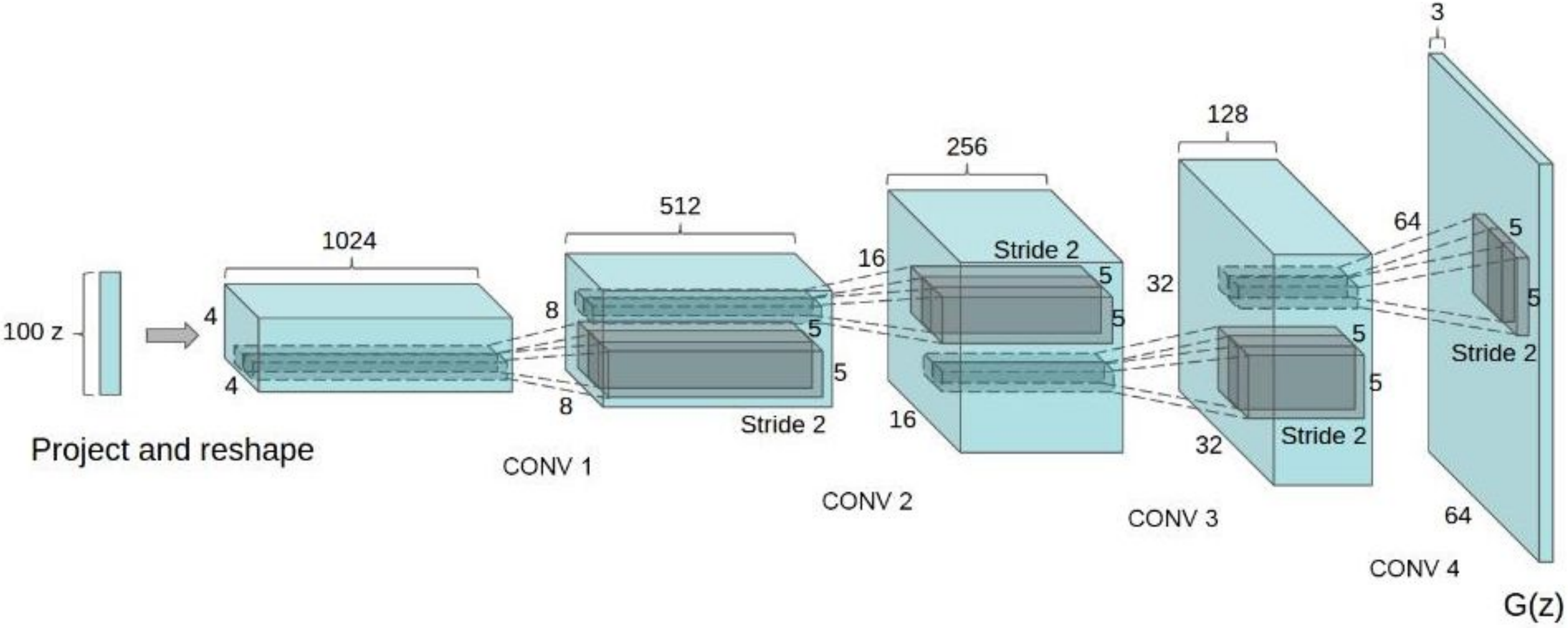
https://arxiv.org/pdf/1611.07004


mojo


$N(0, 1)$
Генератор
P_{fake}
Дискримина
тор
fake
real
P_{real}


0
10
20
30
40
50
60

DCGAN



100 z
Project and reshape
1024
4
4
CONV 1
512
8
8
5
5
Stride 2
CONV 2
256
16
16
Stride 2
5
5
CONV 3
128
32
32
5
5
Stride 2
CONV 4
3
64
64
5
5
Stride 2
G(z)

Checkerboard artefacts

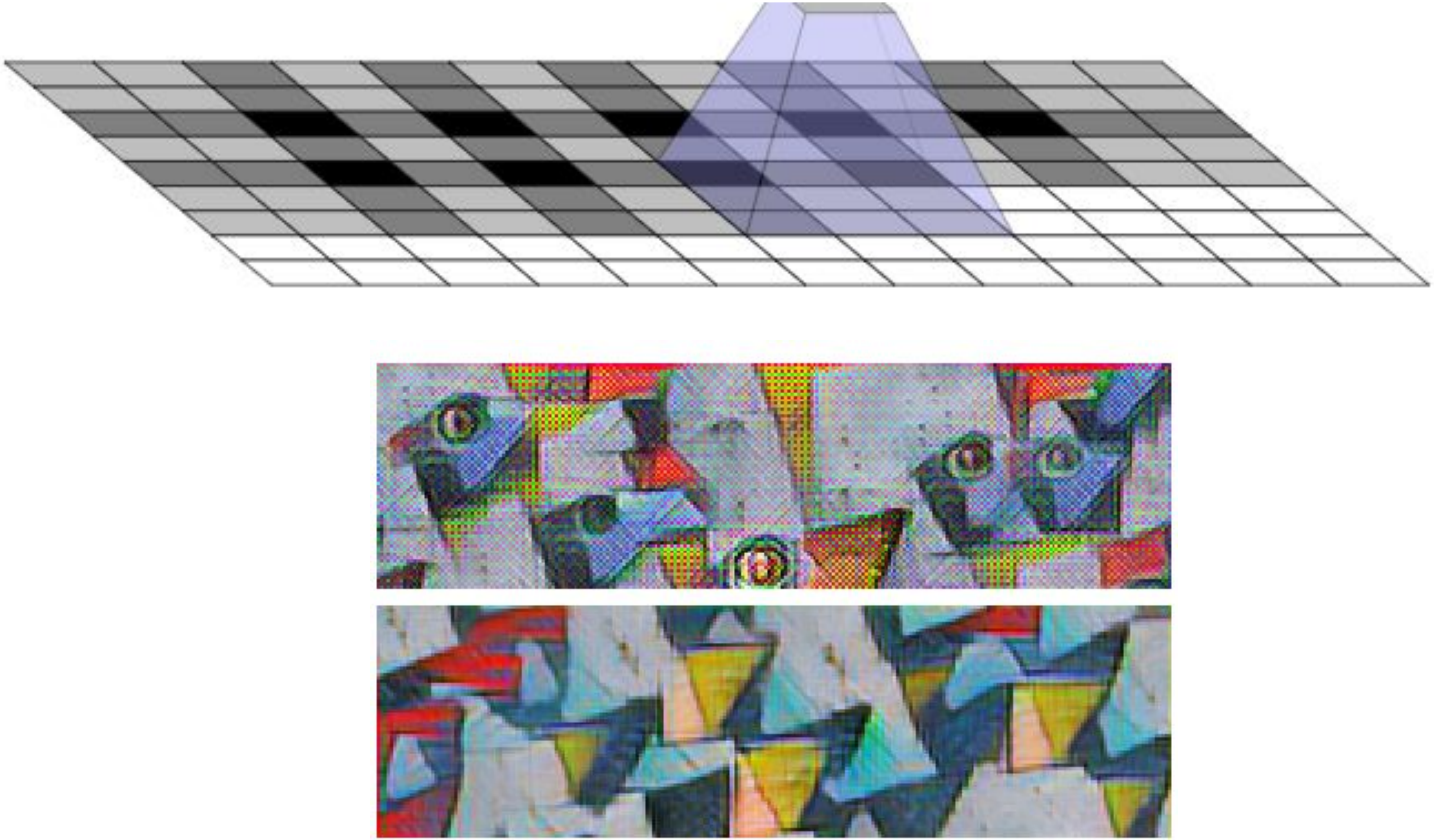


Table 9: BigGAN-deep architecture for 512×512 images.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$ Embed(y) $\in \mathbb{R}^{128}$
Linear $(128 + 128) \rightarrow 4 \times 4 \times 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 4ch$
Non-Local Block (64×64)
ResBlock $4ch \rightarrow 4ch$
ResBlock up $4ch \rightarrow 2ch$
ResBlock $2ch \rightarrow 2ch$
ResBlock up $2ch \rightarrow ch$
ResBlock $ch \rightarrow ch$
ResBlock up $ch \rightarrow ch$
BN, ReLU, 3×3 Conv $ch \rightarrow 3$
Tanh

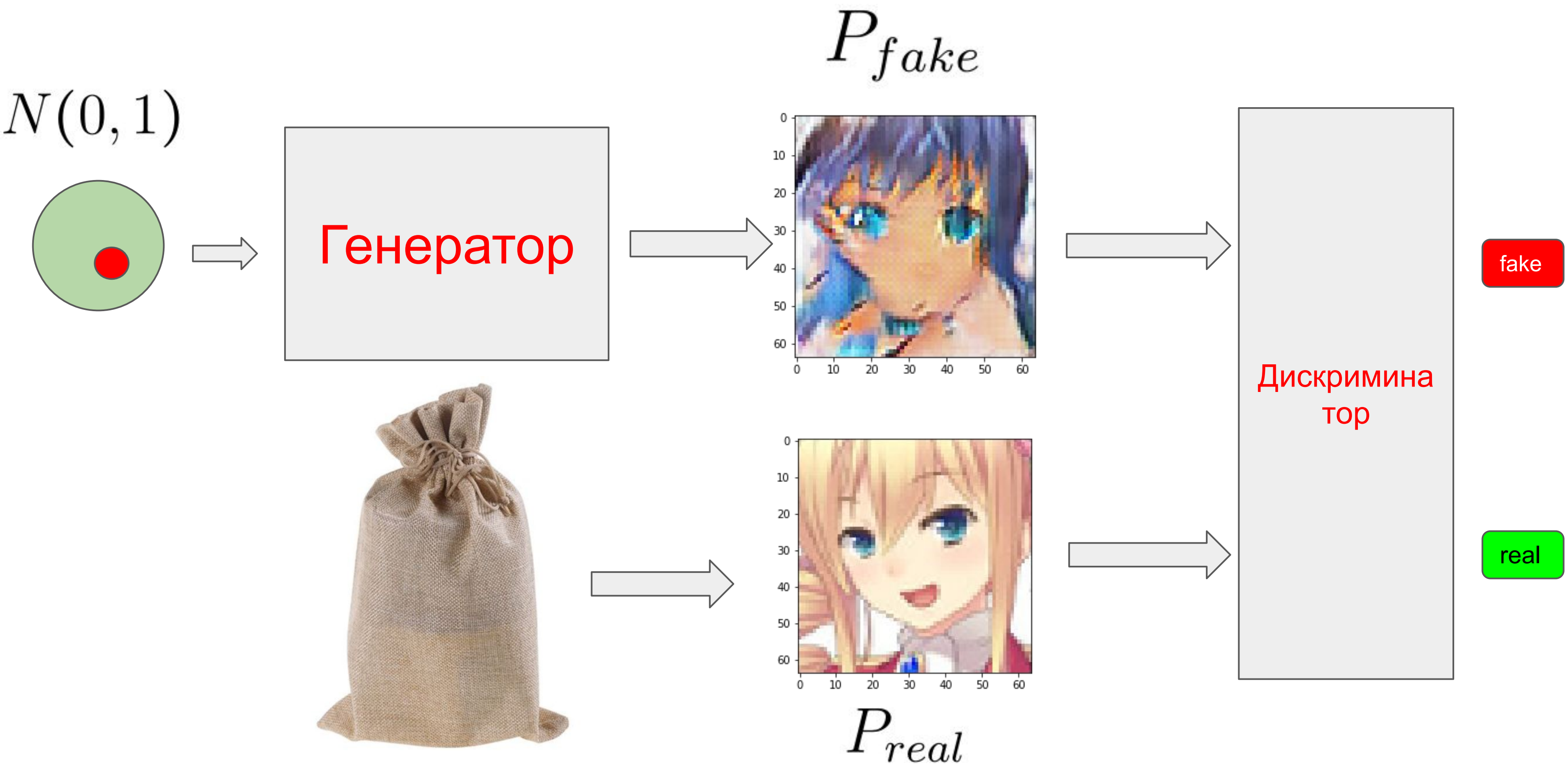
(a) Generator

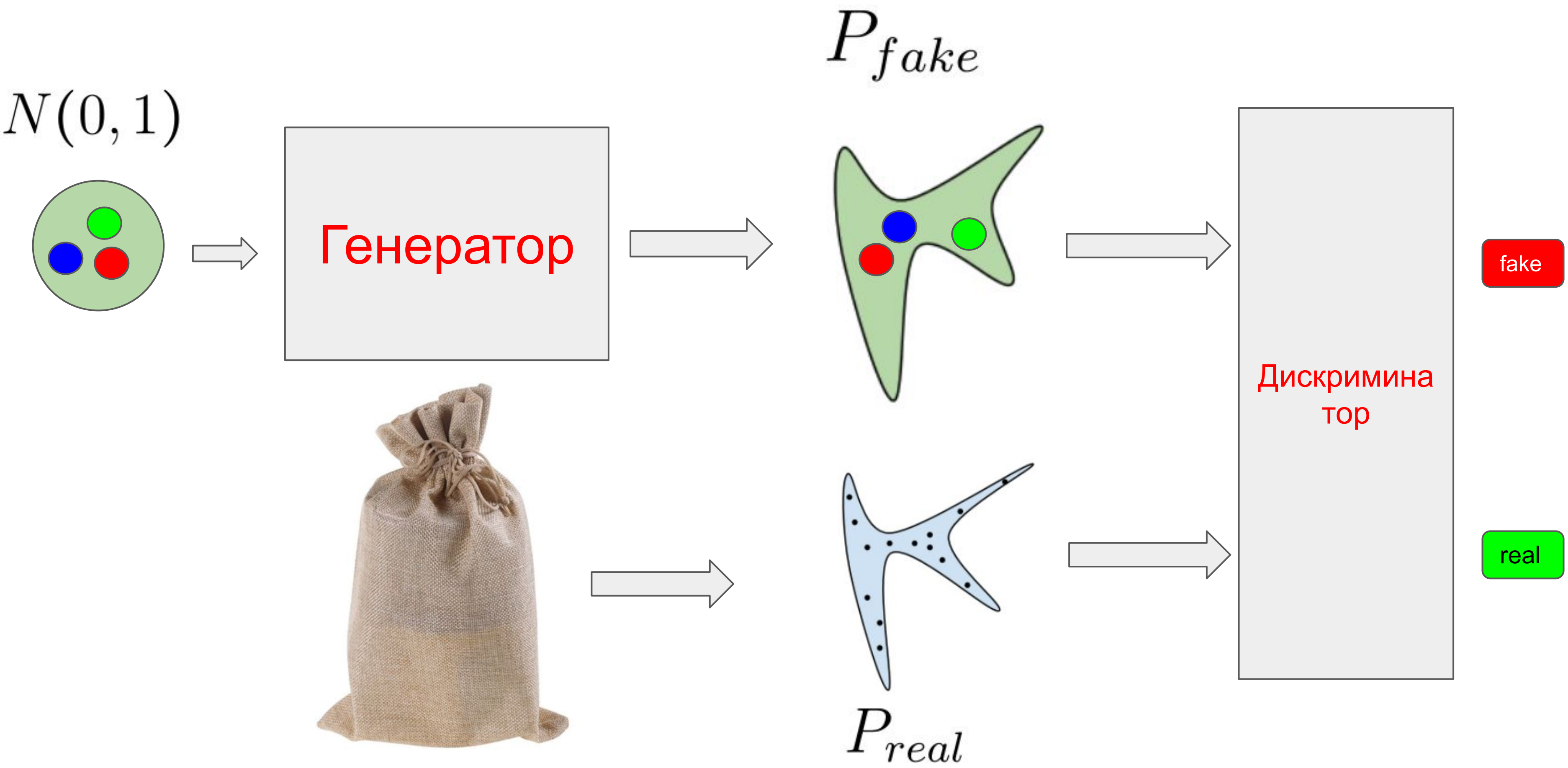
RGB image $x \in \mathbb{R}^{512 \times 512 \times 3}$
3×3 Conv $3 \rightarrow ch$
ResBlock down $ch \rightarrow ch$
ResBlock $ch \rightarrow ch$
ResBlock down $ch \rightarrow 2ch$
ResBlock $2ch \rightarrow 2ch$
ResBlock down $2ch \rightarrow 4ch$
ResBlock $4ch \rightarrow 4ch$
Non-Local Block (64×64)
ResBlock down $4ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock down $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ReLU, Global sum pooling
Embed(y)$\cdot\boldsymbol{h}$ + (linear $\rightarrow$ 1)

(b) Discriminator

1. Нормализация картинки тангенсом
2. min log(1-D) -> -log(D)
3. Нормальное распределение
4. BatchNorm
5. Без нулевых градиентов (ReLU, MaxPool)
6. Noisy labels
7. DCGAN
8. Параметры Adam

https://github.com/soumith/ganhacks


$N(0,1)$
Генератор
P_{fake}
P_{real}
Дискримина
тор
fake
real


$N(0,1)$
Генератор
P_{fake}
P_{real}
Дискримина
тор
fake
real

$$Loss = E_{x \in P_{real}}[\log D(x)] + E_{x \in P_{fake}}[\log(1 - D(x))]$$

Дискриминатор = argmax (Loss)
Генератор = argmin (Loss)

Пусть задан генератор G, каков дискриминатор D?

$$Loss = \int_x P_{real}(x)[\log D(x)] + \int_x P_{fake}(x)[\log(1 - D(x))] \to max$$

$$Loss = \int_x P_{real}(x)[\log D(x)] + P_{fake}(x)[\log(1 - D(x))] \to max$$

Интеграл - сумма по всем x, значит достаточно максимизировать каждый x в отдельности

Пусть задан генератор G и точка x, каков дискриминатор D(x)?

$$P_{real}(x)[\log D(x)] + P_{fake}(x)[\log(1 - D(x))] \to max$$

Берем производную по D приравниваем 0

$$P_{real}(x)\frac{1}{D(x)} + P_{fake}(x)\frac{-1}{1 - D(x)} = 0$$

$$(1 - D(x))P_{real}(x) = D(x)P_{fake}(x)$$

$$P_{real}(x) - D(x)P_{real}(x) = D(x)P_{fake}(x)$$

$$P_{real}(x) = D(x)(P_{fake}(x) + P_{real}(x))$$

$$D(x) = \frac{P_{real}(x)}{P_{fake}(x) + P_{real}(x)}$$

Подставляем в исходное выражение

Получаем оптимальный дискриминатор

$$\int_x P_{real}(x)[\log \frac{P_{real}(x)}{P_{fake}(x) + P_{real}(x)}] + P_{fake}(x)[\log \frac{P_{fake}(x)}{P_{fake}(x) + P_{real}(x)}]$$

Осталось лишь привести его к красивому виду

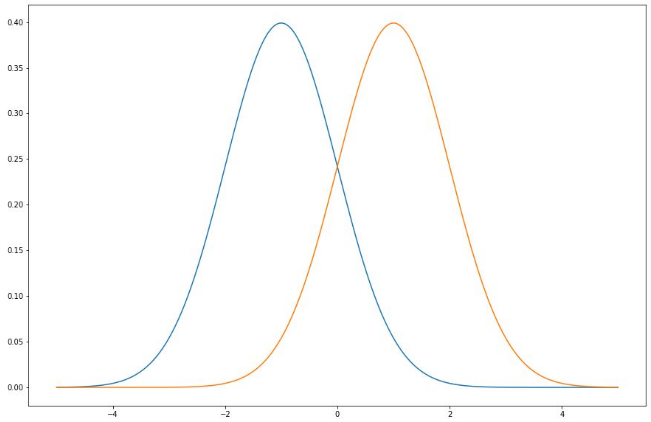
$$-2\log 2 + \int_x P_{real}(x)[\log \frac{P_{real}(x)}{(P_{fake}(x) + P_{real}(x))/2}] + \int_x P_{fake}(x)[\log \frac{P_{fake}(x)}{(P_{fake}(x) + P_{real}(x))/2}]$$

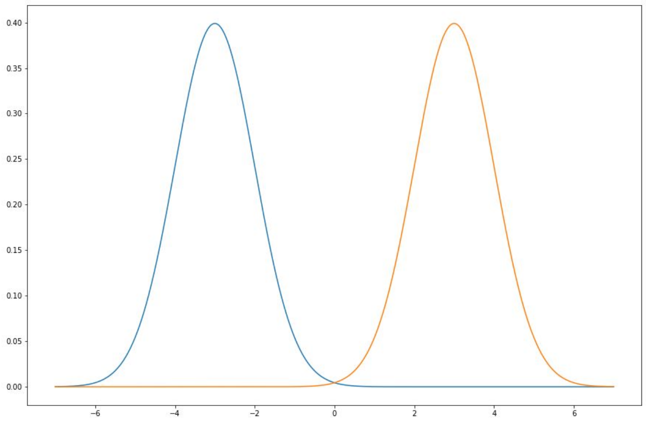
$$-2\log 2 + KL\left(P_{real} \middle| \frac{P_{real} + P_{fake}}{2}\right) + KL\left(P_{fake} \middle| \frac{P_{real} + P_{fake}}{2}\right)$$

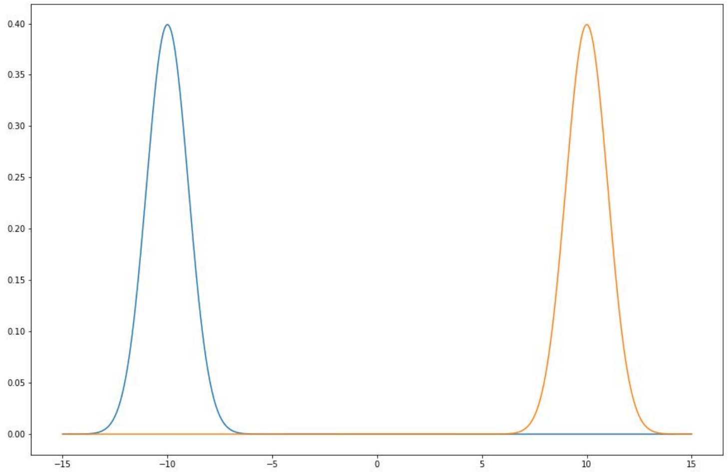
$$D = -2\log 2 + 2JS(P_{real}|P_{fake})$$

$$GAN = \underset{G}{\operatorname{argmin}}\, \underset{D}{\operatorname{argmax}}\, Loss(P_{real}, P_{fake}) = \underset{G}{\operatorname{argmin}}\, JS(P_{real}|P_{fake})$$

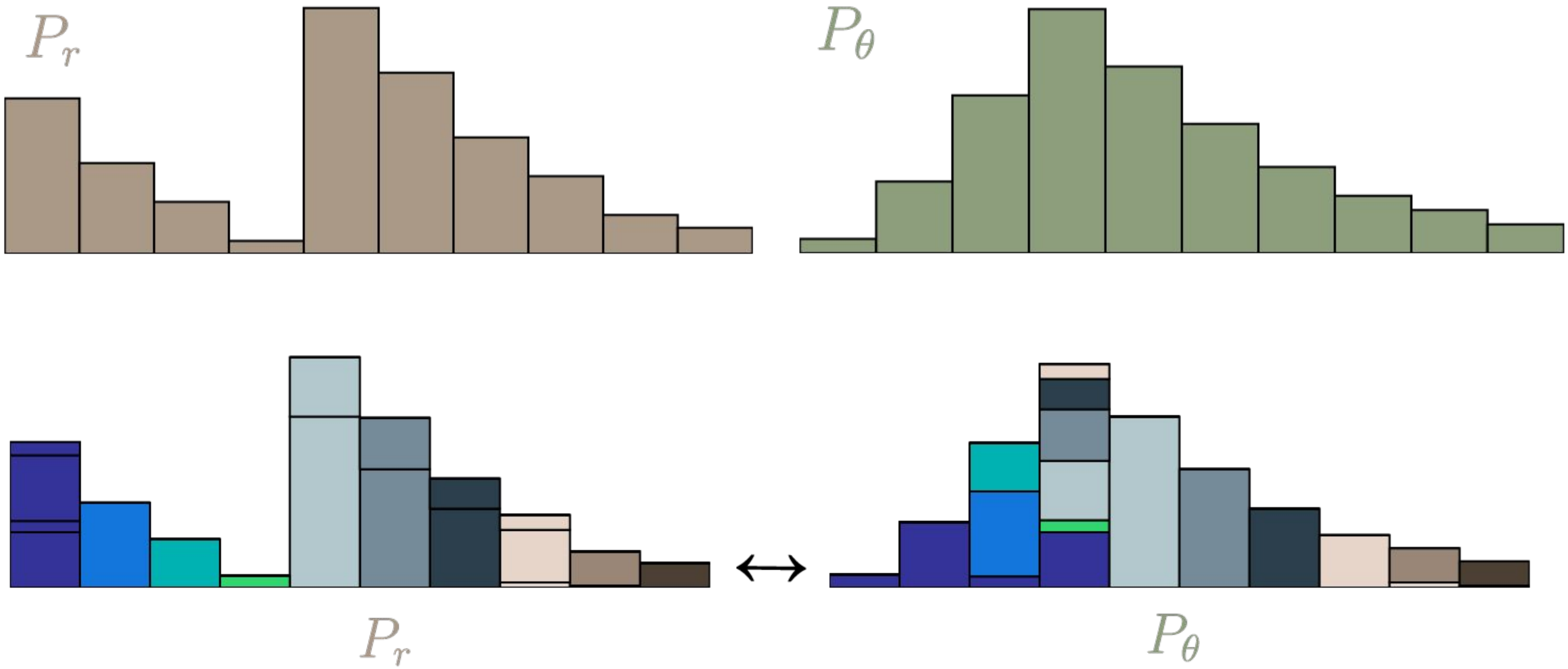
Проблема JS divergence



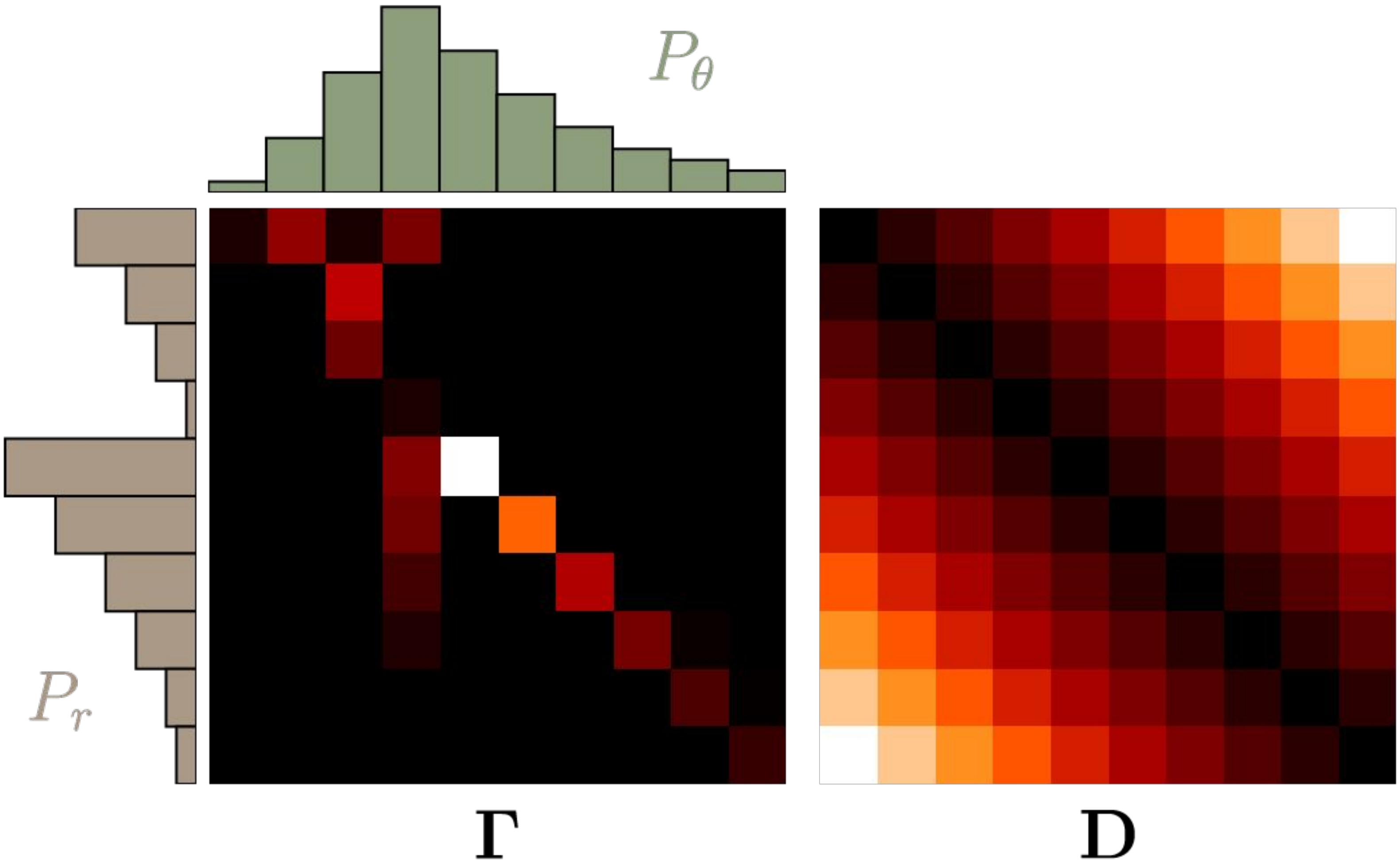




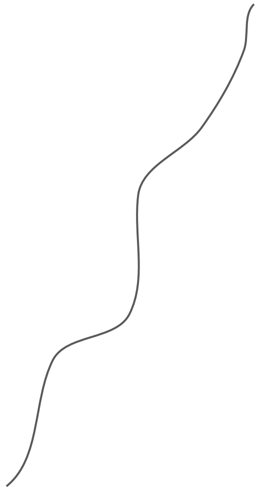
Earth mover distance

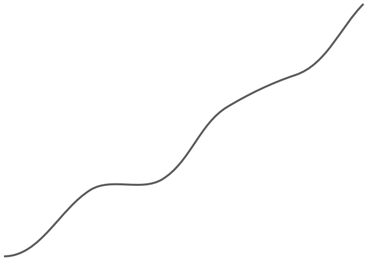


$$EMD(P_r, P_\theta) = \min_S \sum_{x,y} |x - y| S(x, y)$$


P_θ
P_r
$\mathbf{\Gamma}$
$\mathbf{D}$

1-Липшиц функции





$$\forall x, y : |f(x) - f(y)| \leq |x - y| \Leftrightarrow \forall x : |f'(x)| \leq 1$$

$$WGAN = \underset{G}{\operatorname{argmin}} \underset{D \in 1-lipschitz}{\operatorname{argmax}} E_{x \in P_{real}(x)} D(x) - E_{x \in P_{fake}(x)} D(x)$$

Способы сделать 1-липшиц регуляризацию

WGAN

$$w \leftarrow w + \alpha \cdot \mathrm{RMSProp}(w, g_w)$$
$$w \leftarrow \mathrm{clip}(w, -c, c)$$

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

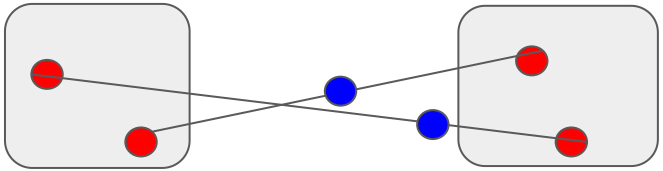
while θ has not converged **do**
    for $t = 0, ..., n_{\text{critic}}$ **do**
        Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
        Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
        $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
        $w \leftarrow w + \alpha \cdot \mathrm{RMSProp}(w, g_w)$
        $w \leftarrow \mathrm{clip}(w, -c, c)$
    end for
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
    $\theta \leftarrow \theta - \alpha \cdot \mathrm{RMSProp}(\theta, g_\theta)$
end while

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 $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **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



$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r}[D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}\left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right]}_{\text{Our gradient penalty}}.$$

Spectral normalisation

$$||Ax|| \leq K||x|$$

$$\langle Ax, Ax\rangle \leq K^2\langle x, x\rangle, \forall x \in I$$

$$\langle (A^T A - K^2)x, x\rangle \leq 0, \forall x \in I.$$

$$\begin{aligned}
\langle (A^T A - K^2)x, x\rangle &= \langle (A^T A - K^2)\sum_i x_i v_i, \sum_j x_j v_j\rangle \\
&= \sum_i \sum_j x_i x_j \langle (A^T A - K^2)v_i, v_j\rangle \\
&= \sum_i (\lambda_i - K^2)x_i^2 \leq 0 \\
&\implies \sum_i (K^2 - \lambda_i)x_i^2 \geq 0.
\end{aligned}$$

$$K^2 - \lambda_i \geq 0 \text{ for all } i = 1 \ldots n.$$

$$||f||_{\text{Lip}} = \sup_x \sigma(\nabla f(x))$$

$$\nabla(g \circ f)(x) = \nabla g(f(x))\nabla f(x).$$

$$\nabla(g \circ f)(x) = \nabla g(f(x))\nabla f(x).$$

$$\sigma(\nabla f(x)) = \sup_{||v|| \leq 1} ||[\nabla f(x)]v||$$

$$\sigma(\nabla(g \circ f)(x)) = \sup_{||v|| \leq 1} ||[\nabla g(f(x))][\nabla f(x)]v||$$

$$\sup_{||v|| \leq 1} ||[\nabla g(f(x))][\nabla f(x)]v|| \leq \sup_{||u|| \leq 1} ||[\nabla g(f(x))]u|| \sup_{||v|| \leq 1} ||[\nabla f(x)]v||.$$

$$||g \circ f||_{\text{Lip}} \leq ||g||_{\text{Lip}}||f||_{\text{Lip}}.$$

Power iteration

$$v_{t+1} = \frac{W^T W v_t}{||W^T W v_t||}$$

$$\begin{aligned} v_t &= \frac{(W^T W)^t \sum_i v_i e_i}{||(W^T W)^t \sum_i v_i e_i||} \\ &= \frac{\sum_i v_i \lambda_i^t e_i}{||\sum_i v_i \lambda_i^t e_i||} \\ &= \frac{v_1 \lambda_1^t \sum_i \frac{v_i}{v_1} \left(\frac{\lambda_i}{\lambda_1}\right)^t e_i}{||v_1 \lambda_1^t \sum_i \frac{v_i}{v_1} \left(\frac{\lambda_i}{\lambda_1}\right)^t e_i||}. \end{aligned}$$

$$\sigma(W) = ||Wv|| = u^T W v.$$

https://christiancosgrove.com/blog/2018/01/04/spectral-normalization-explained.html

Progressive growing of GAN

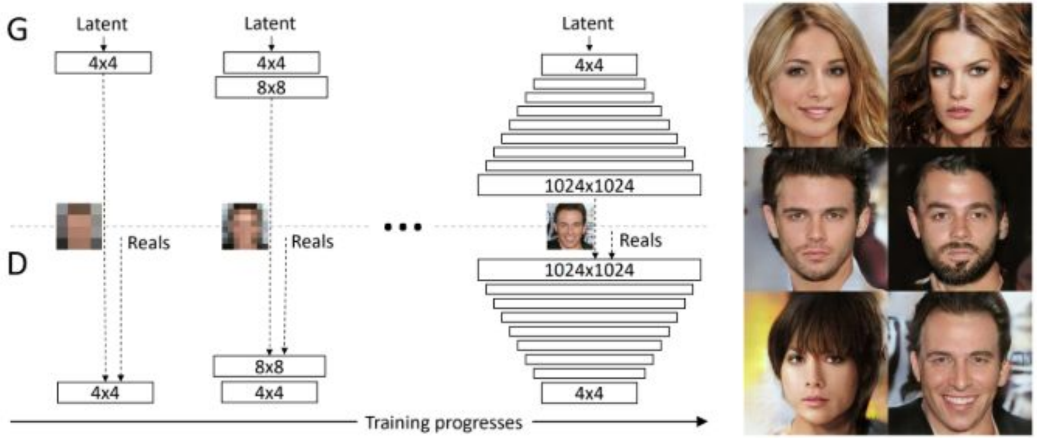


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $\boxed{N \times N}$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. One the right we show six example images generated using progressive growing at 1024×1024.

Discriminator	Act.	Output shape	Params
Input image	–	3 × 1024 × 1024	–
Conv 1 × 1	LReLU	16 × 1024 × 1024	64
Conv 3 × 3	LReLU	16 × 1024 × 1024	2.3k
Conv 3 × 3	LReLU	32 × 1024 × 1024	4.6k
Downsample	–	32 × 512 × 512	–
Conv 3 × 3	LReLU	32 × 512 × 512	9.2k
Conv 3 × 3	LReLU	64 × 512 × 512	18k
Downsample	–	64 × 256 × 256	–
Conv 3 × 3	LReLU	64 × 256 × 256	37k
Conv 3 × 3	LReLU	128 × 256 × 256	74k
Downsample	–	128 × 128 × 128	–
Conv 3 × 3	LReLU	128 × 128 × 128	148k
Conv 3 × 3	LReLU	256 × 128 × 128	295k
Downsample	–	256 × 64 × 64	–
Conv 3 × 3	LReLU	256 × 64 × 64	590k
Conv 3 × 3	LReLU	512 × 64 × 64	1.2M
Downsample	–	512 × 32 × 32	–
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Downsample	–	512 × 16 × 16	–
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Downsample	–	512 × 8 × 8	–
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Downsample	–	512 × 4 × 4	–
Minibatch stddev	–	513 × 4 × 4	–
Conv 3 × 3	LReLU	512 × 4 × 4	2.4M
Conv 4 × 4	LReLU	512 × 1 × 1	4.2M
Fully-connected	linear	1 × 1 × 1	513
Total trainable parameters			**23.1M**

Вопросы

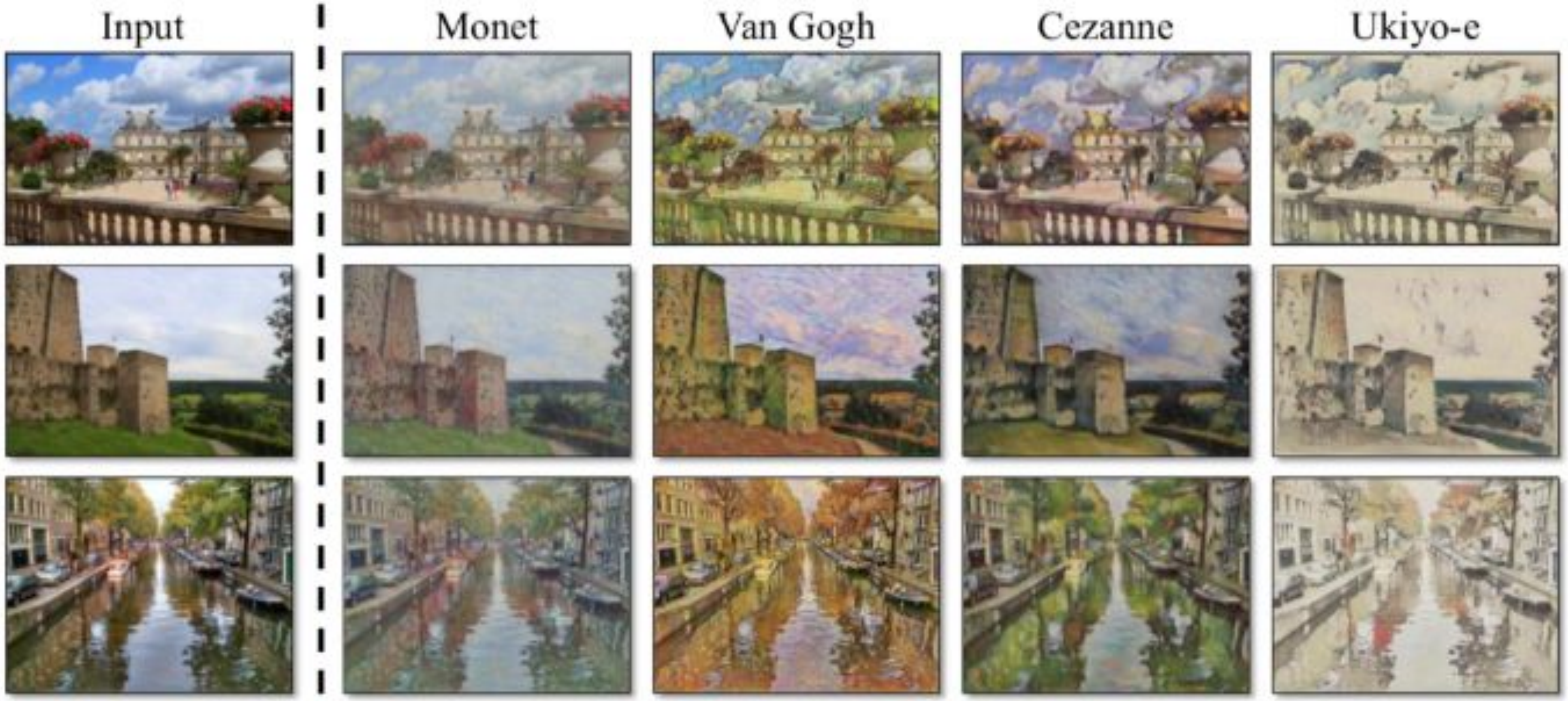
1. Почему wasserstein gan обучается лучше и стабильнее обычного (проблема kl divergence)

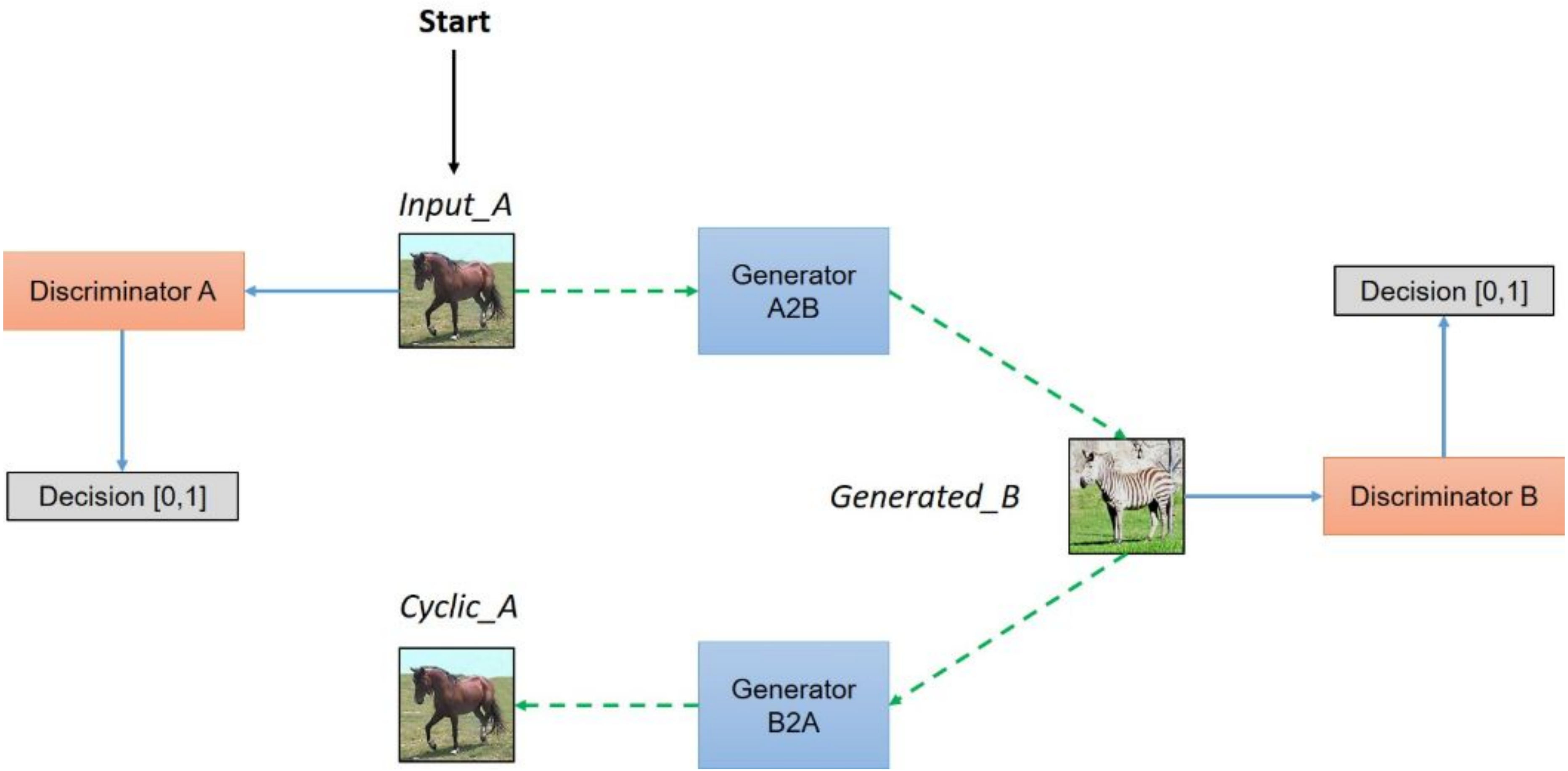
2. Что такое расстояние тракториста (earth mover distance), как его можно считать (ЛП, разница функций распределений по модулю в одномерном случае)

3. Два способа из 3 поддерживать 1 липшиц св-во, интуиция

4. Чем помогает progressive growing of gans, как он устроен

Cycle GAN




Start
Input_A
Discriminator A
Generator
A2B
Decision [0,1]
Decision [0,1]
Generated_B
Discriminator B
Cyclic_A
Generator
B2A

Источники

https://www.youtube.com/channel/UC2ggjtuuWvxrHHHiaDH1dlQ

https://blog.acolyer.org/2018/05/10/progressive-growing-of-gans-for-improved-quality-stability-and-variation/

https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

https://sthalles.github.io/advanced_gans/

https://christiancosgrove.com/blog/2018/01/04/spectral-normalization-explained.html

https://github.com/soumith/ganhacks