

The background image is a detailed illustration of a green dragon's head on the left, with its mouth slightly open and a fierce expression. On the right, a metallic knight's helmet with a blue and gold decorative band is shown. The lighting creates strong highlights and shadows on the scales of the dragon and the metallic surfaces of the helmet.

курс «Глубокое обучение»

Генеративные состязательные сети

Александр Дьяконов

8 декабря 2021 года

План

Генератор и дискриминатор – Adversarial idea

GAN: обучение – min-max-игра

Настройка GAN, советы по настройке GAN

Нестабильность «non-saturating game», Mode-Collapse

Least Square GAN (LSGAN)

Hinge loss based GAN

Wasserstein GAN (WGAN)

WGAN-GP

Спектральная нормировка (Spectral Normalization): SN-GAN

Maximum Mean Discrepancy (MMD)

f-GAN

Relativistic GANs (RGANs)

Energy-Based GAN (EBGAN)

Как оценивать качество: IS, MS, FID, SSIM, Kernel MMD, Wasserstein distance, 1NN

Generative Adversarial Networks (GAN)

**Есть фундаментальная проблема – учим генерировать «котиков»
– как измерить качество модели?**

Generative ~ учим генеративную модель
Adversarial ~ в состязательной парадигме
Networks ~ DNN

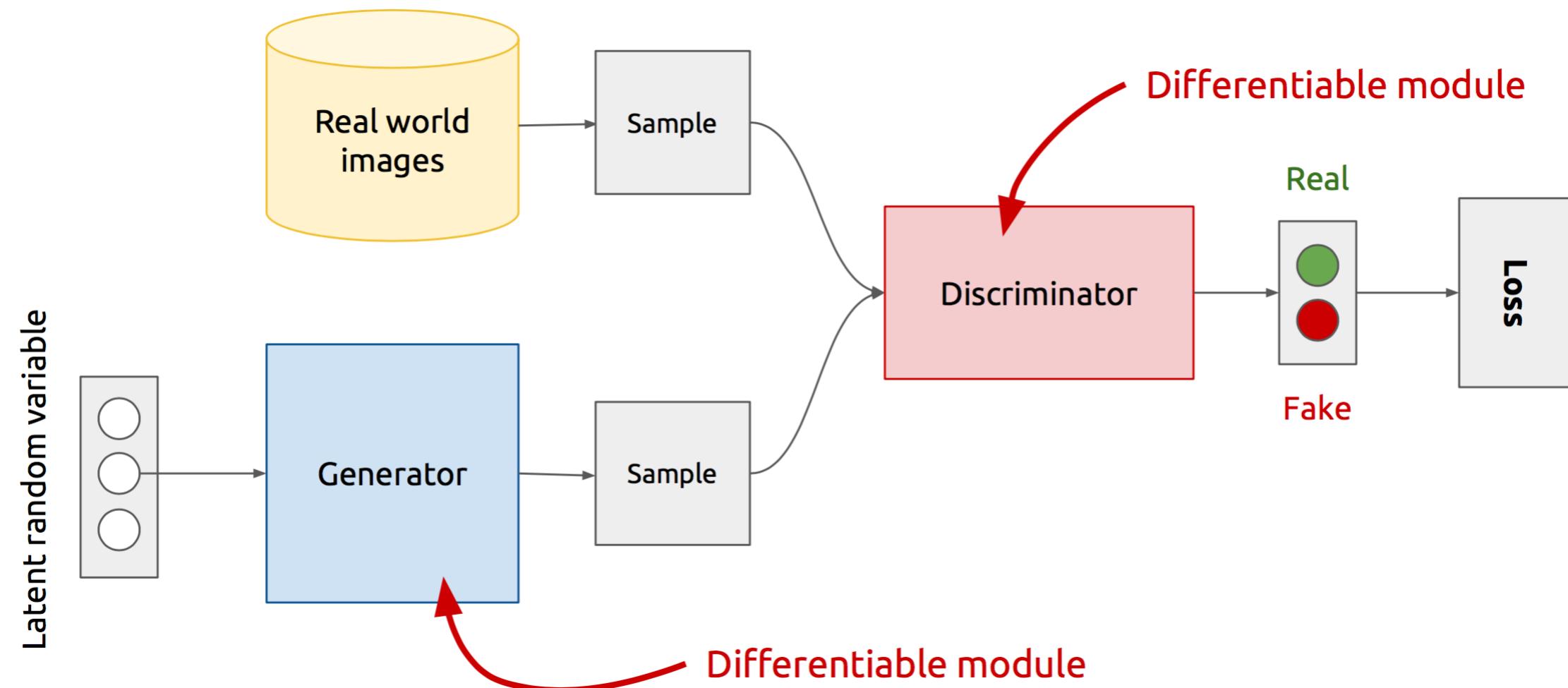
Модели

**Дискриминативные
оценивают $P(X|Y)$**

**Генеративные
моделируют $P(X)$**

[Goodfellow et al., 2014]

GAN: Генератор и дискриминатор



Генератор – сеть, которая порождает объект (изображение) из шума

Дискриминатор – сеть, различающая настоящие и сгенерированные объекты

Generative Adversarial Networks (GAN)

- **главное: по сути, дискриминатор – дифференцируемая функция ошибки!**
эту идею можно использовать там, где нет подходящих функций ошибок...
- **в отличие от PixelCNN, VAE нет явной функции плотности!**
т.н. «*implicit density model*»
- **игровой подход!**

Что могут GAN

- **научились работать со сложными объектами в пространствах высокой размерности**
- **генерация реалистичных объектов**
- **заполнение пропусков (и другие задачи USL)**
- **использование генеративных моделей-ассистентов (при редактировании)**

Задачи

- **улучшение изображений (Image Inpainting)**
- **улучшение звука (Speech Enhancement)**
- **генерация изображений (Image Generation)**
- **супер-разрешение (Super-resolution)**
- **раскраска изображений (Colorization)**
- **творчество (Artwork)**
- **пополнение датасета (Synthetic data), моделирование (Simulation)**

TABLE 2: Applications of GANs discussed in Section 5

Field	Subfield	Method
Image processing and computer vision	Super-resolution	SRGAN [63], ESRGAN [64], Cycle-in-Cycle GANs [65], SRDGAN [66], TGAN [67]
	Image synthesis and manipulation	DR-GAN [68], TP-GAN [69], PG ² [70], PSGAN [71], APDrawingGAN [72], IGAN [73], introspective adversarial networks [74], GauGAN [75]
	Texture synthesis	MGAN [76], SGAN [77], PSGAN [78]
	Object detection	Segan [79], perceptual GAN [80], MTGAN [81]
	Video	VGAN [82], DRNET [83], Pose-GAN [84], video2video [85], MoCoGan [86]
Sequential data	Natural language processing (NLP)	RankGAN [87], IRGAN [88], [89], TAC-GAN [90]
	Music	RNN-GAN (C-RNN-GAN) [91], ORGAN [92], SeqGAN [93], [94]

<https://arxiv.org/pdf/2001.06937.pdf>

Идея состязания (Adversarial idea)

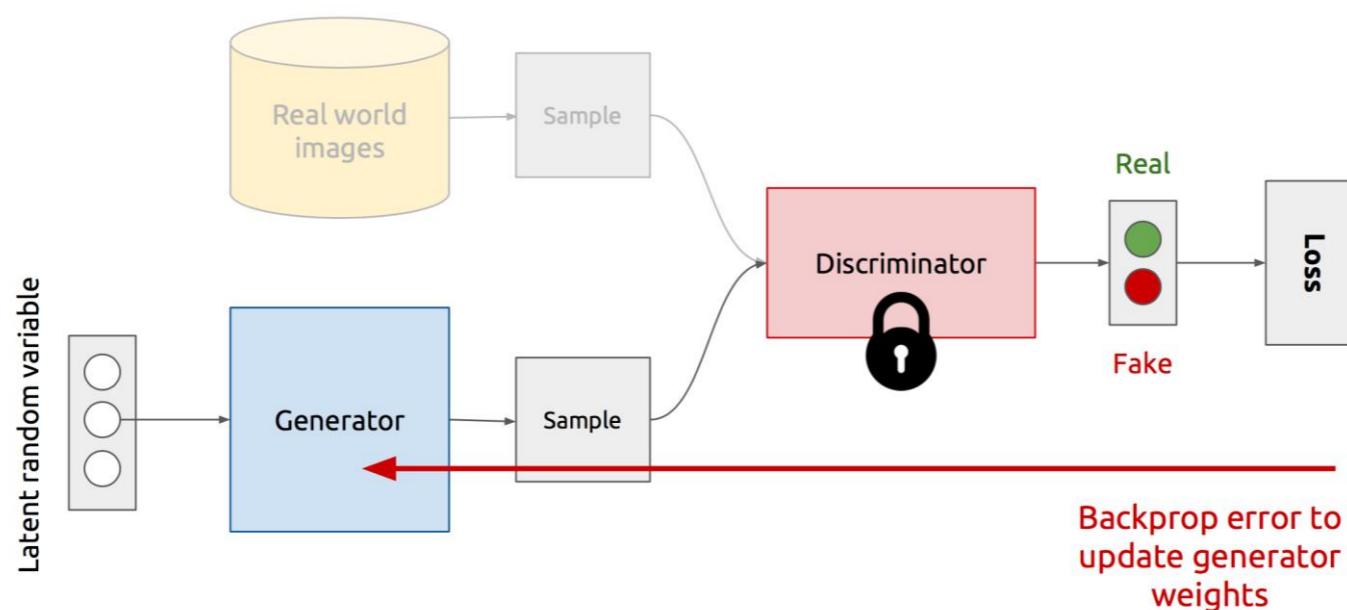
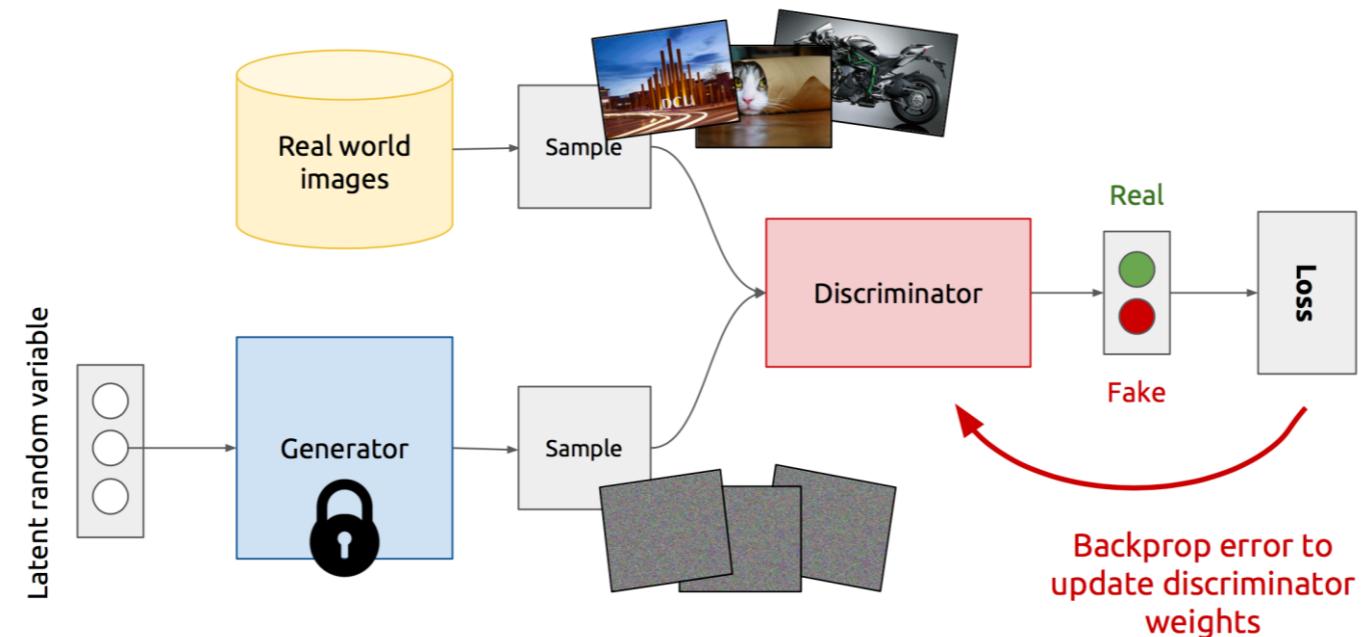
AlphaGo – две сети соперничают друг с другом

Adversarial examples – объекты, которые «неправильно классифицируются»

**ex: небольшая модификация объектов какого-то класса,
меняющая результат классификации**

«adversarial attacks

GAN: обучение



GAN: обучение – min-max-игра

$$\min_{\theta} \max_{\varphi} \left[\mathbf{E}_{x \sim p_{\text{data}}} \log D_{\varphi}(x) + \mathbf{E}_{z \sim p(z)} \log(1 - D_{\varphi}(G_{\theta}(z))) \right]$$

Дискриминатор выводит правдоподобие из [0, 1]

$D_{\varphi}(x)$ – для настоящих данных

$D_{\varphi}(G_{\theta}(z))$ – для сгенерированных данных

Дискриминатор хочет $D_{\varphi}(x) \rightarrow 1, D_{\varphi}(G_{\theta}(z)) \rightarrow 0$

Генератор хочет $D_{\varphi}(G_{\theta}(z)) \rightarrow 1$

Дискриминатор – обычный классификатор

cross entropy loss

Генератор – зависит от того, что генерируем...

Ian J. Goodfellow et al. Generative Adversarial Nets <https://arxiv.org/pdf/1406.2661.pdf>

GAN: обучение – min-max-игра**Если зафиксировать генератор**

$$\min_{\theta} \max_{\varphi} \left[\underbrace{\mathbf{E}_{x \sim p_{\text{data}}} \log D_{\varphi}(x) + \mathbf{E}_{z \sim p(z)} \log(1 - D_{\varphi}(G_{\theta}(z)))}_{\int (p_{\text{data}}(x) \log D_{\varphi}(x) + p_G(x) \log(1 - D_{\varphi}(x)) dx} \right]$$

Взяв производную по D и приравняв к нулю:

$$D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}$$

– оптимальный дискриминатор при фиксированном генераторе**здесь распределение фейков – $p_G(x) \sim x = G_{\theta}(z |_{z \sim p(z)})$**

GAN: обучение – min-max-игра

Если подставить оптимальный дискриминатор (существенное условие)

$$\begin{aligned}
 & \int (p_{\text{data}}(x) \log D_\phi(x) + p_G(x) \log(1 - D_\phi(x))) dx = \\
 &= \int \left(p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)} + p_G(x) \log \frac{p_G(x)}{p_{\text{data}}(x) + p_G(x)} \right) dx = \\
 &= D_{KL} \left(p_{\text{data}}(x) \parallel \frac{p_{\text{data}}(x) + p_G(x)}{2} \right) + D_{KL} \left(p_G(x) \parallel \frac{p_{\text{data}}(x) + p_G(x)}{2} \right) - 2 \log 2
 \end{aligned}$$

дивергенция Йенсена-Шеннона / JS (Jensen-Shannon Divergence)

$$\text{JSP}(p \parallel q) = \frac{\text{KL}(p \parallel (p + q) / 2) + \text{KL}(q \parallel (p + q) / 2)}{2}$$

проблемы: 1) если носители распределений не пересекаются
2) градиенты затухают, когда D обучился

GAN: обучение – мин-макс-игра

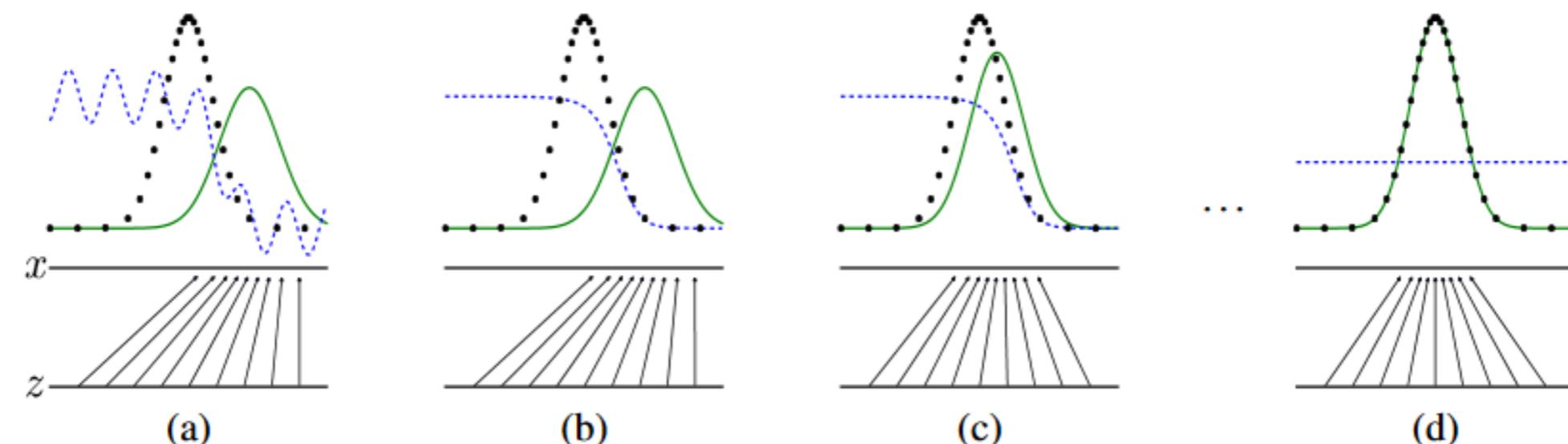


Figure 1: Generative adversarial nets are trained by simultaneously updating the discriminative distribution (D , blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) p_{data} from those of the generative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which z is sampled, in this case uniformly. The horizontal line above is part of the domain of x . The upward arrows show how the mapping $x = G(z)$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g . (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}$. (c) After an update to G , gradient of D has guided $G(z)$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(x) = \frac{1}{2}$.

GAN: практическая оптимизация – «non-saturating game»

для оптимизации лучше «non-saturating game»:

$$\max_{\varphi} \left[\mathbf{E}_{x \sim p_{\text{data}}} \log D_{\varphi}(x) + \mathbf{E}_{z \sim p(z)} \log(1 - D_{\varphi}(G_{\theta}(z))) \right]$$

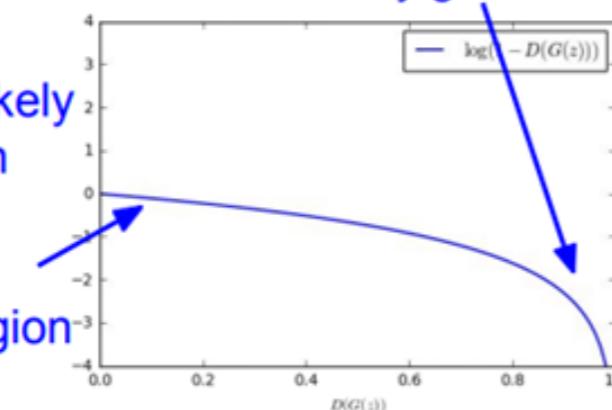
$$\max_{\theta} \mathbf{E}_{z \sim p(z)} \log(D_{\varphi}(G_{\theta}(z)))$$

$$\min_{\theta} \mathbf{E}_{z \sim p(z)} \log(1 - D_{\varphi}(G_{\theta}(z)))$$

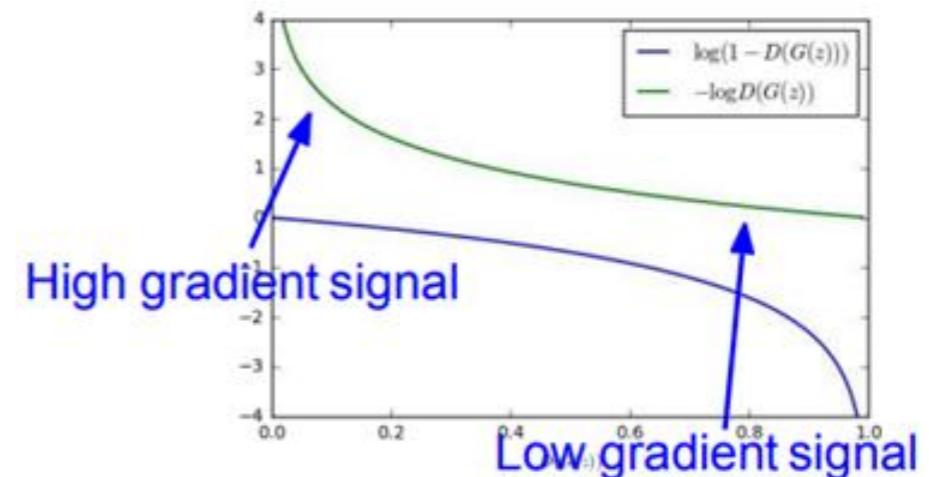
$$\max_{\theta} \mathbf{E}_{z \sim p(z)} \log(D_{\varphi}(G_{\theta}(z)))$$

Gradient signal
dominated by region
where sample is
already good

When sample is likely
fake, want to learn
from it to improve
generator. But
gradient in this region
is relatively flat!



**вместо \min правдоподобия корректности
дискриминатора – \max правдоподобия того,
что дискриминатор ошибался**



Нестабильность «non-saturating game»

если расписать эвристическую функцию ошибки есть некоторое противоречие...

Кроме, того несимметричные ошибки генератора:

If $p_{data}(x) \rightarrow 0$ and $p_g(x) \rightarrow 1$, we have $KL(p_g \| p_{data}) \rightarrow +\infty$.

If $p_{data}(x) \rightarrow 1$ and $p_g(x) \rightarrow 0$, we have $KL(p_g \| p_{data}) \rightarrow 0$.

указанная дивергенция входит как слагаемое в функцию ошибки

Maximum likelihood game

$$\begin{aligned} J^{(G)} &= E_{z \sim p_z(z)} \left[-\exp \left(\sigma^{-1} (D(G(z))) \right) \right] \\ &= E_{z \sim p_z(z)} \left[-D(G(z)) / (1 - D(G(z))) \right], \end{aligned}$$

<https://arxiv.org/pdf/2001.06937.pdf>

Сравнение

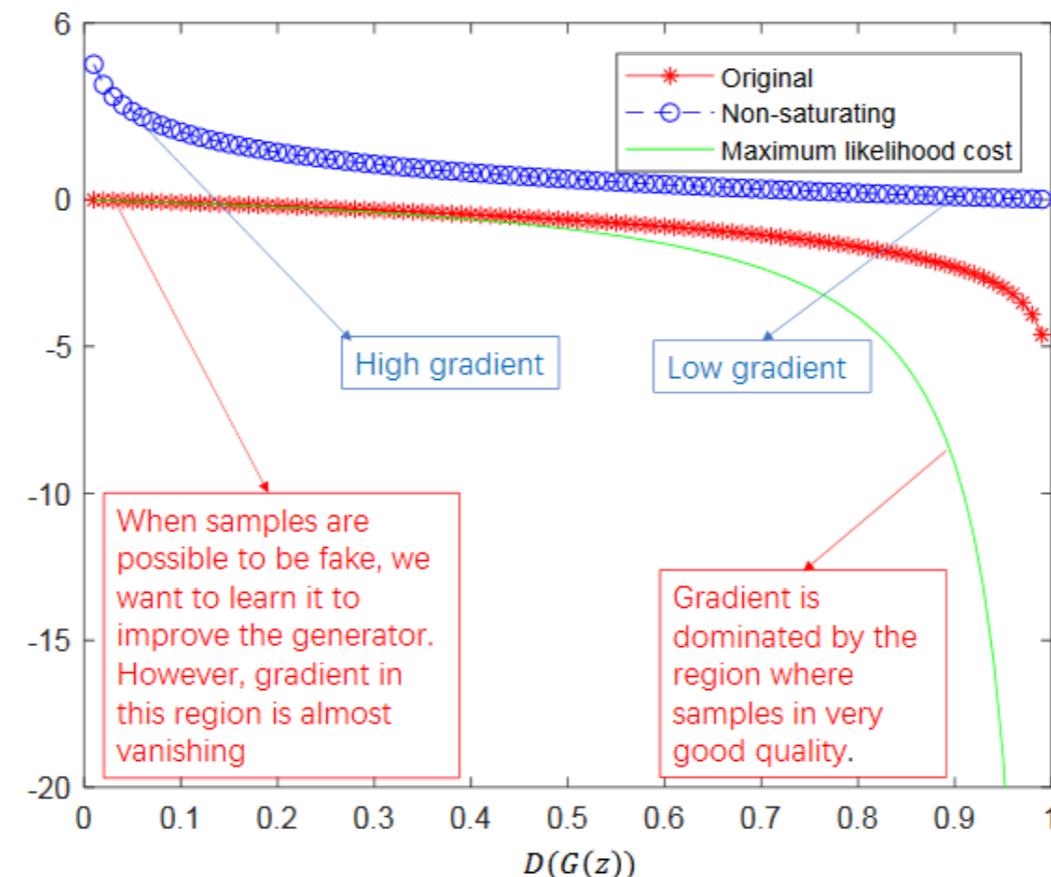


Fig. 1: The three curves of “Original”, “Non-saturating”, and “Maximum likelihood cost” denotes $\log(1 - D(G(z)))$, $-\log(D(G(z)))$, and $-D(G(z))/(1 - D(G(z)))$ in (1), (7), and (13), respectively. The cost that the generator has for generating a sample $G(z)$ is only decided by the discriminator’s response to that generated sample. The larger probability the discriminator gives the real label to the generated sample, the less cost the generator gets. This figure is reproduced from [103], [123].

Алгоритм настройки

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

end for

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

GAN: первые примеры



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)

Ian J. Goodfellow et al. Generative Adversarial Nets <https://arxiv.org/pdf/1406.2661.pdf>

GAN: проблемы первых моделей

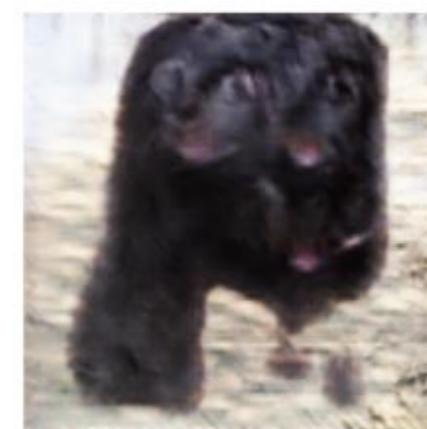
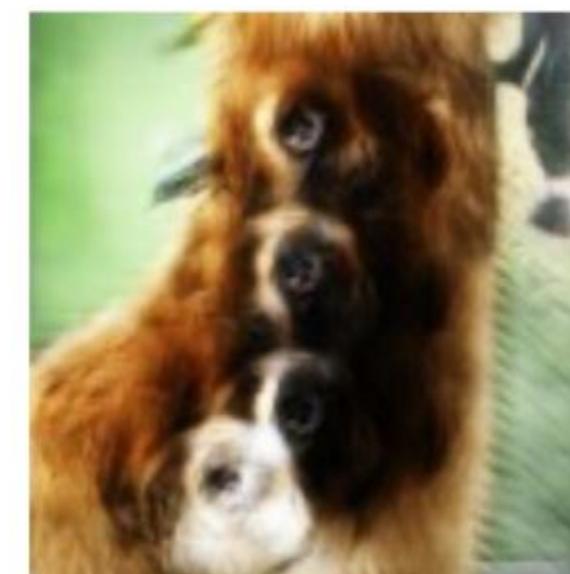
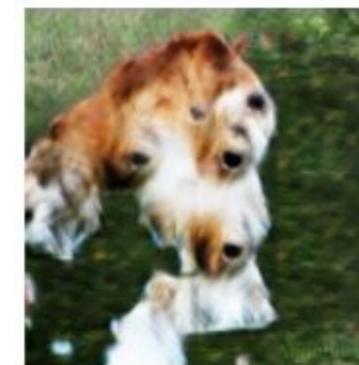
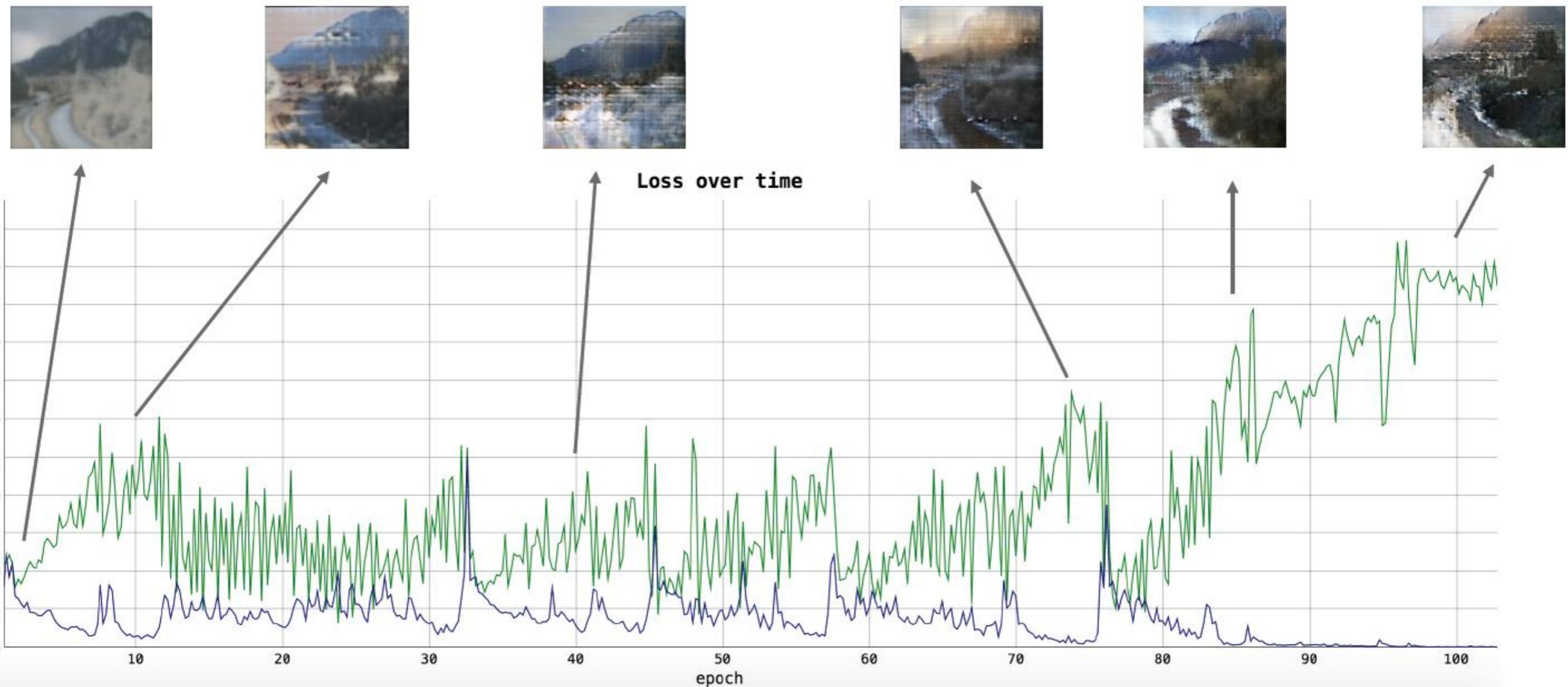


Figure 29: GANs on 128×128 ImageNet seem to have trouble with counting, often generating animals with the wrong number of body parts.

– глобальная структура и неправильное число элементов изображения

Ian Goodfellow «NIPS 2016 Tutorial: Generative Adversarial Networks» <https://arxiv.org/abs/1701.00160>

Обучение GAN: проблемы на практике



проблемка... вспоминаем про затухание градиентов;)

Обучение GAN: проблемы на практике

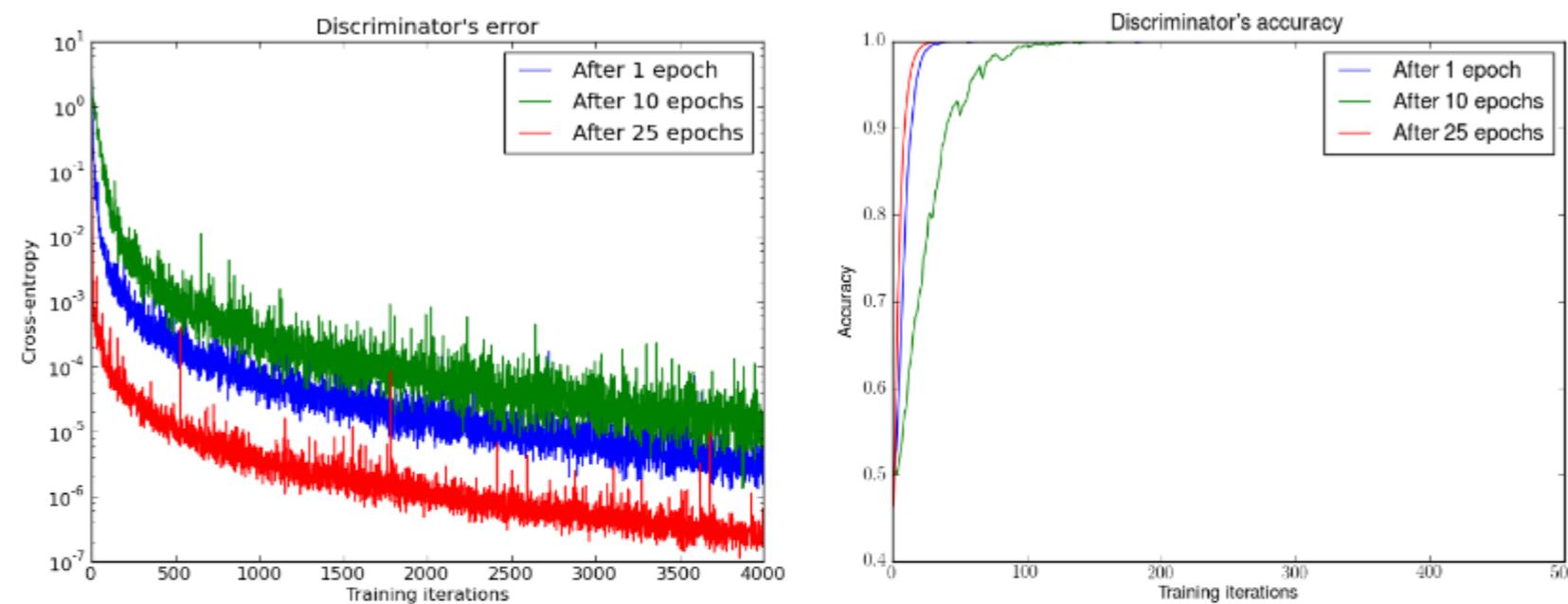


Figure 1: First, we trained a DCGAN for 1, 10 and 25 epochs. Then, with the generator fixed we train a discriminator from scratch. We see the error quickly going to 0, even with very few iterations on the discriminator. This even happens after 25 epochs of the DCGAN, when the samples are remarkably good and the supports are likely to intersect, pointing to the non-continuity of the distributions. Note the logarithmic scale. For illustration purposes we also show the accuracy of the discriminator, which goes to 1 in sometimes less than 50 iterations. This is 1 even for numerical precision, and the numbers are running averages, pointing towards even faster convergence.

Дискриминатор получается лучше, генератор существенно хуже...

Martin Arjovsky, Léon Bottou Towards Principled Methods for Training Generative Adversarial Networks //
<https://arxiv.org/abs/1701.04862>

Обучение GAN: проблемы на практике

Теоретически показано, что в JSD носители распределений могут не пересекаться, что вырождает задачу оптимизации (см. формулу) ■

Theorem 2.3. Let \mathbb{P}_r and \mathbb{P}_g be two distributions whose support lies in two manifolds \mathcal{M} and \mathcal{P} that don't have full dimension and don't perfectly align. We further assume that \mathbb{P}_r and \mathbb{P}_g are continuous in their respective manifolds. Then,

$$JSD(\mathbb{P}_r \parallel \mathbb{P}_g) = \log 2$$

$$KL(\mathbb{P}_r \parallel \mathbb{P}_g) = +\infty$$

$$KL(\mathbb{P}_g \parallel \mathbb{P}_r) = +\infty$$

Обучение GAN: к чему приводит «non-saturating game»

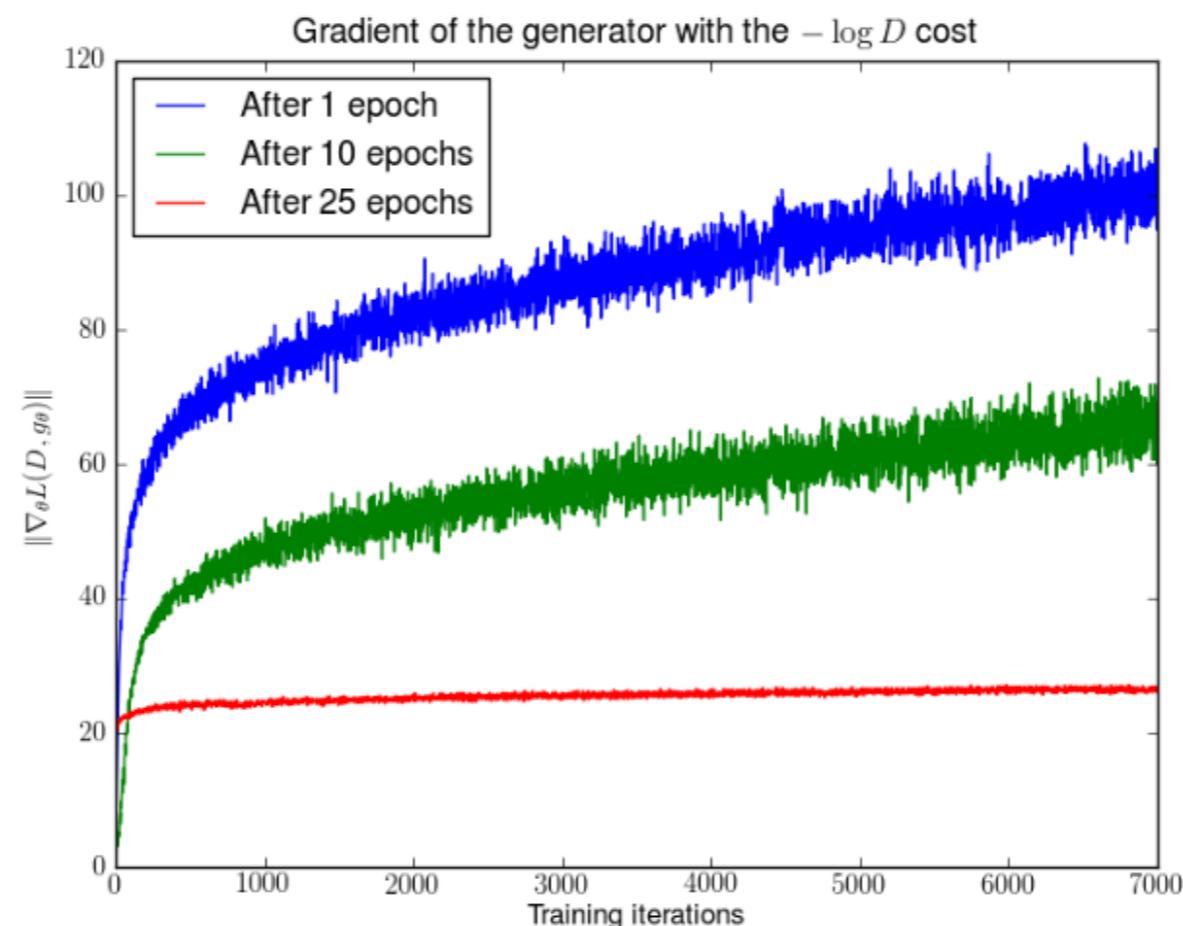


Figure 3: First, we trained a DCGAN for 1, 10 and 25 epochs. Then, with the generator fixed we train a discriminator from scratch and measure the gradients with the $-\log D$ cost function. We see the gradient norms grow quickly. Furthermore, the noise in the curves shows that the variance of the gradients is also increasing. All these gradients lead to updates that lower sample quality notoriously.

нестабильность – норма градиента (и дисперсия нормы) вырастает!

Обучение GAN: проблемы на практике

**Очень сложно обучать
Есть проблемы со стабильностью
плюс дальше опишем сложности**

⇒ **много трюков и приёмов**
например для стабильности добавляем шум,
но тогда изображения размываются

**в дискриминативной постановке решаем невыпуклую задачу оптимизации – можем
сойтись в локальный, а не глобальный минимум**

**В состояние равновесия можем вообще не попасть
В игровой парадигме хочется найти равновесия, но
SGD не для этого, плохой пример – $\min_x \max_y xy$**

Советы по настройке GAN (не все полезны для всех видов GAN-ов)

- 1. Нормализация входа (изображения $\rightarrow [-1, +1]$, выход – \tanh)**
- 2. Вместо $\min(\log(1-D(G)))$ лучше $\max(\log(D(G)))$**
приём: меняем метки местами `real \leftrightarrow fake`
- 3. z сэмплируется не из равномерного, а гауссовского распределения, см. также <https://arxiv.org/abs/1609.04468>**
- 4. Минибатчи лучше делать чистыми (все `real` или все `fake`)**
или специально составляют минибатч **Virtual batch normalization (VBN)**:
референсные объекты + текущий – лучше считаются статистики + по сравнению с
референсными можем справиться с Mode Collapse
- 5. Не использовать Sparse Gradients (ReLU, Pool), лучше LeakyReLU**
Downsampling: Average Pooling, Conv2d + stride
Upsampling: PixelShuffle, ConvTranspose2d + stride
- 6. Лучше размывать метки (label smoothing): $0 \rightarrow [0, 0.3]$, $1 \rightarrow [0.7, 1.2]$**
см. потом LSGAN, чаще только 1 размывают
(чтобы не портить и без того плохие фейки)

Советы по настройке GAN

- 7. Используйте DCGAN (или гибридные: KL + GAN или VAE + GAN)**
- 8. Используйте трюки из RL (например, Experience Replay)**
- 9. Adam для генератора, SGD для дискриминатора**
- 10. Мониторьте ошибки (ex: loss(D)~0 failure mode – проверяйте градиенты)**
- 11. Добавляйте шум ко входу / к слоям генератора / меткам**
- 12. Дискретные переменные в условных GANах:**
используйте Embedding layer, добавляйте как новый канал в изображениях,
поддерживайте низкой embedding dimensionality
- 13. Используйте Dropouts в G**
- 14. Регуляризация нормы градиента**
+ специальное слагаемое, чтобы скорректированные параметры не отличались от средних по всем коррекциям (для стабильности)

Советы по настройке GAN

16. Новые loss-функции

дальше подробнее

17. «Соответствие признаков» (Feature matching)

не только смотреть на ответ дискриминатора, но и на схожесть в признаках средних слоёв фейков и реальных объектов

18. Пусть дискриминатор определяет не только фейк / не фейк, а ещё и класс

Salimans T., et al «Improved Techniques for Training GANs» 2016 //

<https://arxiv.org/pdf/1606.03498.pdf>

How to Train a GAN? Tips and tricks to make GANs work //

<https://github.com/soumith/ganhacks>

Пример настройки GAN <https://poloclub.github.io/ganlab/>

GAN в зависимости от функции ошибки в дискриминаторе

функция ошибки	вид GAN
Binary cross-entropy	Vanilla GAN
Least squares (L2)	LSGAN
EM-distance / Wasserstein-1	WGAN
Wasserstein GAN + Gradient penalty	WGAN-GP

другие функции ошибки – один из ключевых способов решения проблем обучения

Least Square GAN (LSGAN)

квадратичная функция ошибки

$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}(x)} [(D(x) - b)^2] + \frac{1}{2} \mathbb{E}_{z \sim p_z(z)} [(D(G(z)) - a)^2]$$

$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2} \mathbb{E}_{z \sim p_z(z)} [(D(G(z)) - c)^2],$$

борьба с затуханием градиента

Mao et al «Least Squares Generative Adversarial Networks» //
<https://arxiv.org/pdf/1611.04076.pdf>

Hinge loss based GAN

$$V_D(\hat{G}, D) = E_{x \sim p_{data}(x)} [\min(0, -1 + D(x))] \\ + E_{z \sim p_z(z)} [\min(0, -1 - D(\hat{G}(z)))] .$$

$$V_D(G, \hat{D}) = -E_{z \sim p_z(z)} [\hat{D}(G(z))] .$$

J.H. Lim, J.C. Ye «Geometric gan» // <https://arxiv.org/pdf/1705.02894.pdf>

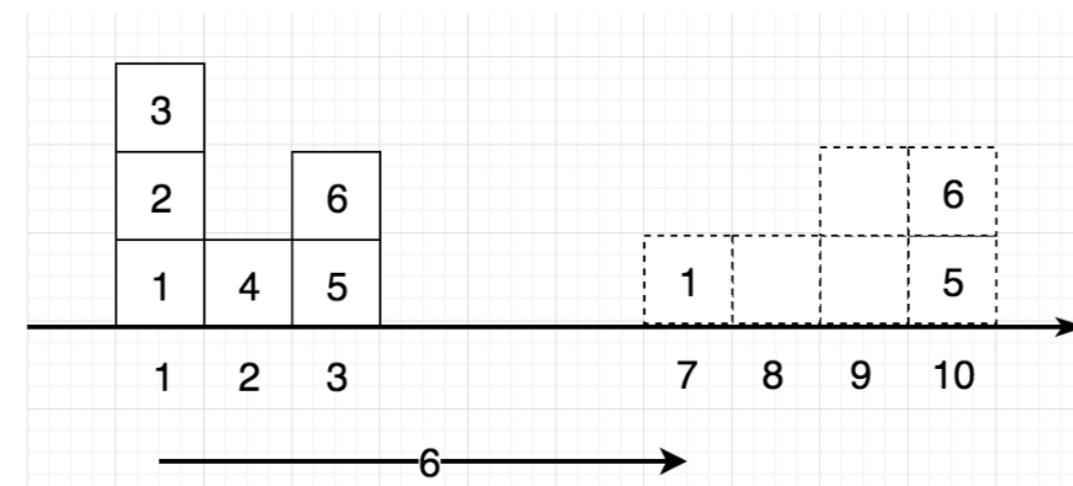
Wasserstein GAN (WGAN)

Earth-Mover (EM) distance (Wasserstein-1)

$$W(p, p') = \inf \mathbf{E}_{(x, y) \sim \Pi(p, p')} \|x - y\|$$

интерпретация – как «выровнять» распределения, если мы как бы переносим кучи песка... тут нет проблемы с пустыми пересечениями носителей

$\Pi(p, p')$ – семейство совместных распределений с заданными маргинальными



стоимость переноса = вес·расстояние

Wasserstein GAN (WGAN)

Как вычислять?

$$W(p, p') = \inf \mathbf{E}_{(x,y) \sim \Pi(p,p')} \|x - y\|$$

Двойственность Монжа-Канторовича (Kantorovich-Rubinstein duality)

$$W(p, p') = \sup_{\|f\|_L \leq 1} \mathbf{E}_{x \sim p} f(x) - \mathbf{E}_{x \sim p'} f(x)$$

по семейству Липшецевых функций с константой Липшица = 1

На практике для $\|f\|_L \leq 1$ обрезаем градиенты (и работает!)
есть теоретический результат,
но тут подвох в том, что гарантируется только

$$\|f\|_L \leq \text{const}$$

Martin Arjovsky, Soumith Chintala, Léon Bottou «Wasserstein GAN» // <https://arxiv.org/abs/1701.07875>

Wasserstein GAN (WGAN)

Обычный GAN: $\min_G \max_D \mathbf{E}_{x \sim p_{\text{data}}(x)} \log D(x) + \mathbf{E}_{z \sim p_z(z)} \log(1 - D(G(z)))$

WGAN: $\min_G W(p_{\text{data}}, p_G) = \min_G \max_f \mathbf{E}_{x \sim p_{\text{data}}(x)} [f(x)] - \mathbf{E}_{z \sim p_z(z)} [f(G(z))]$

Вместо дискриминатора ~ «клиппированная» функция (НС), которую надо → max назвали «Критиком», нет сигмоиды и решается задача регрессии – это средство для оценки расстояния между распределениями

	Discriminator/Critic	Generator
GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D(\mathbf{x}^{(i)}) + \log (1 - D(G(\mathbf{z}^{(i)}))) \right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (D(G(\mathbf{z}^{(i)})))$
WGAN	$\nabla_w \frac{1}{m} \sum_{i=1}^m [f(x^{(i)}) - f(G(z^{(i)}))]$	$\nabla_{\theta} \frac{1}{m} \sum_{i=1}^m f(G(z^{(i)}))$

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

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.

```
1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  do
3:     Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$  a batch from the real data.
4:     Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
5:      $g_w \leftarrow \nabla_w \left[ \frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)})) \right]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSPProp}(w, g_w)$ 
7:      $w \leftarrow \text{clip}(w, -c, c)$ 
8:   end for
9:   Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
10:   $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$ 
11:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSPProp}(\theta, g_\theta)$ 
12: end while
```

Wasserstein GAN

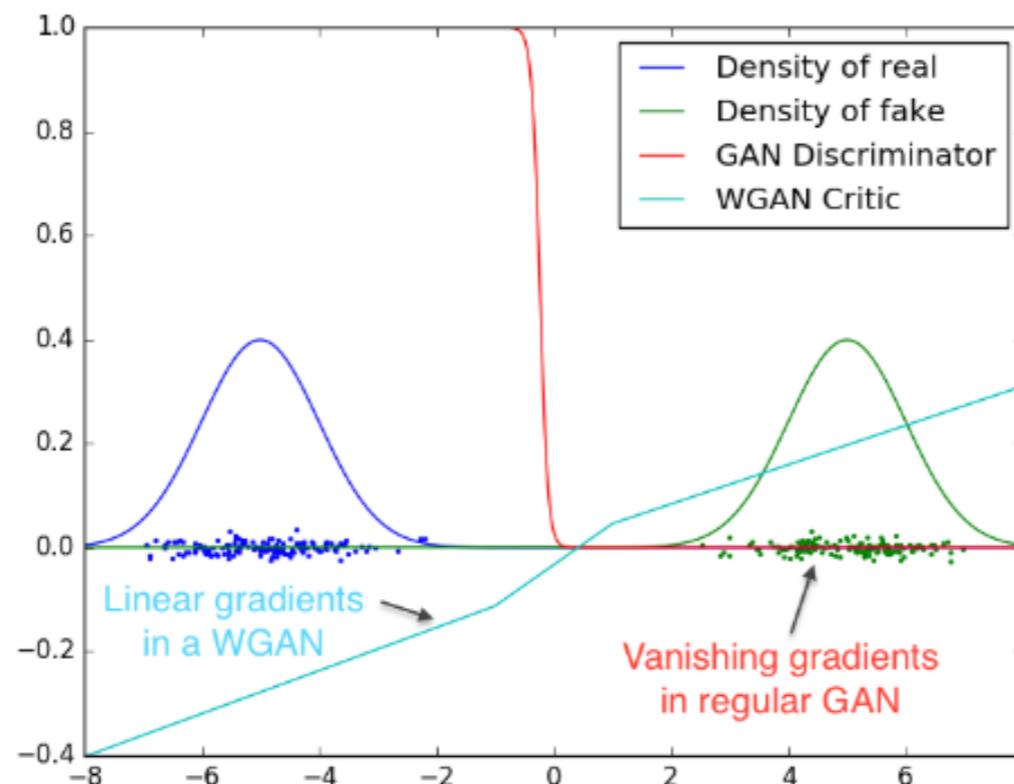


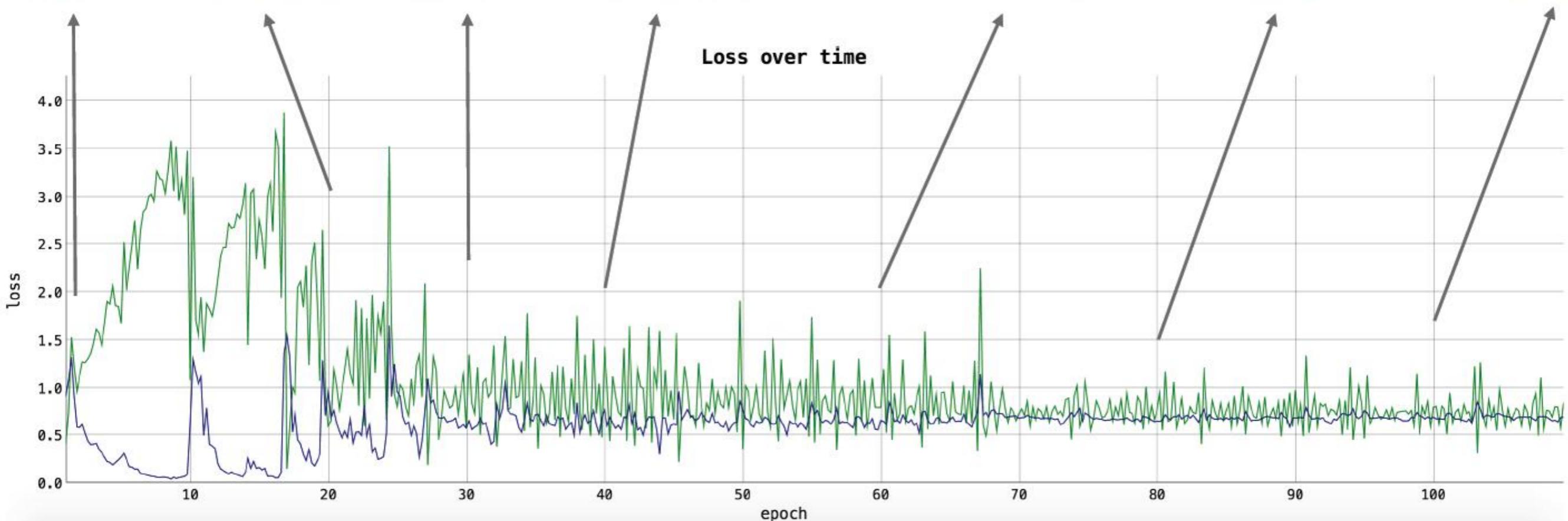
Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

здесь дискриминатор переименовали в критика

нет эффекта коллапса см. дальше

генератор продолжает обучаться, когда критик работает хорошо

Wasserstein GAN



ошибка прямо отражает качество



Figure 5: Algorithms trained with a DCGAN generator. Left: WGAN algorithm. Right: standard GAN formulation. Both algorithms produce high quality samples.

**если не применять BN и другие
приёмы (ех: удваивание
каналов) – всё равно результат
«неплохой»**

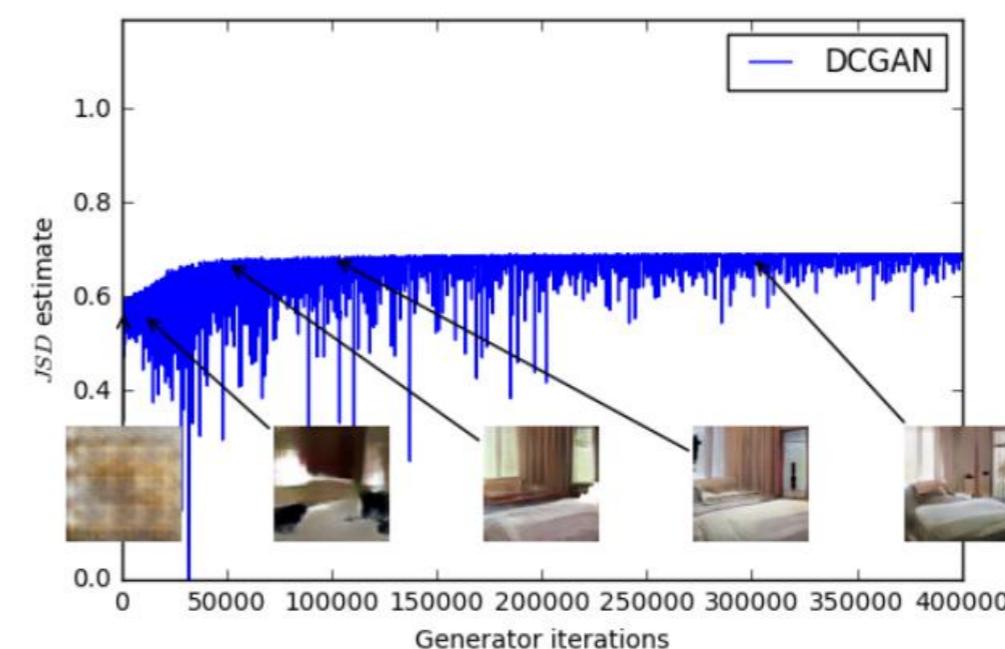
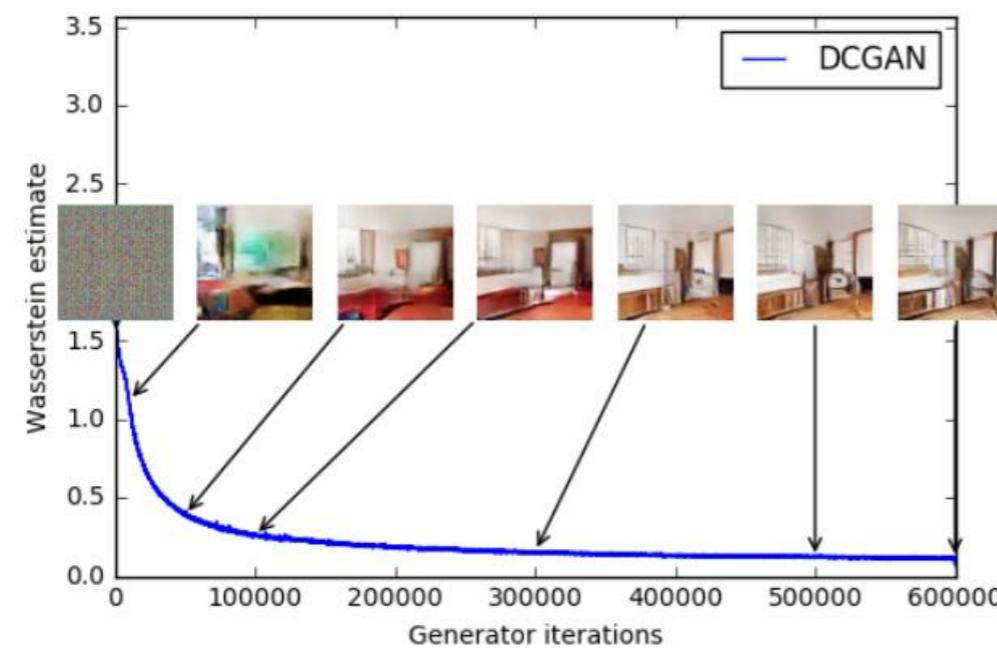
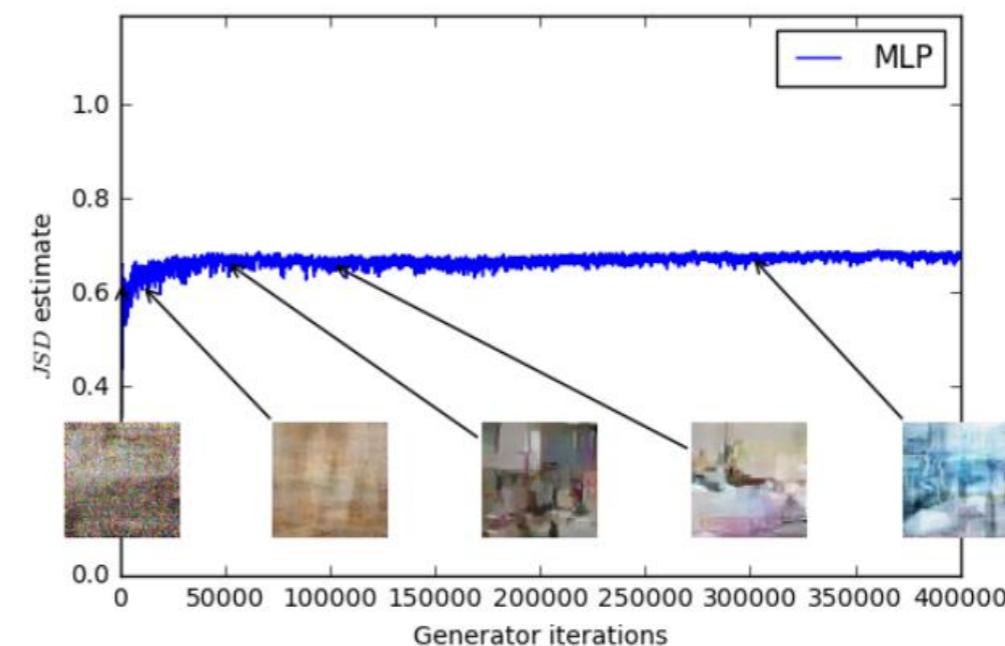
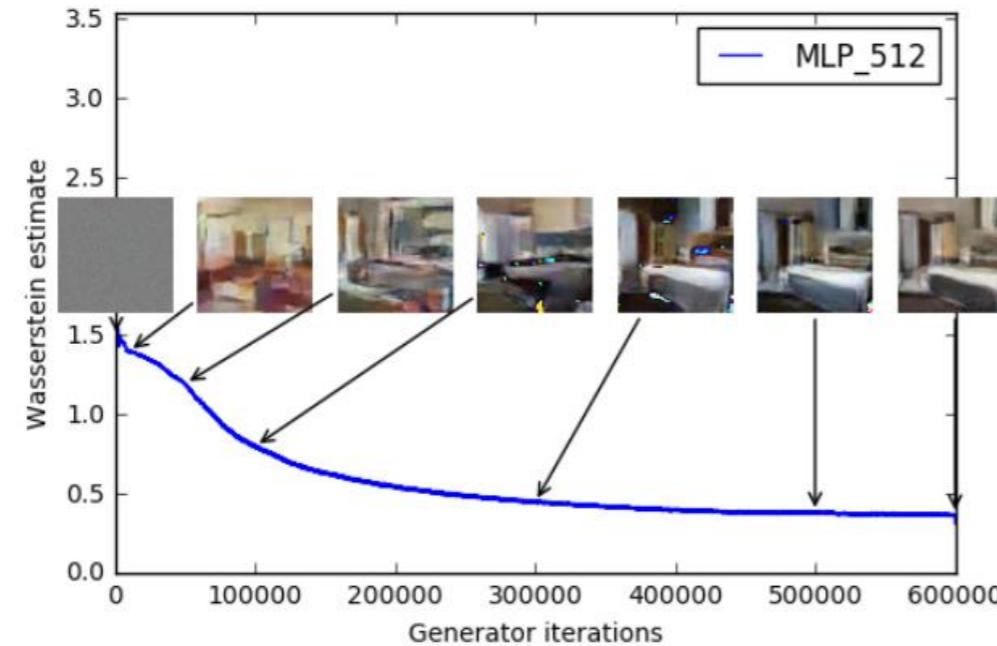


Figure 6: Algorithms trained with a generator without batch normalization and constant number of filters at every layer (as opposed to duplicating them every time as in [18]). Aside from taking out batch normalization, the number of parameters is therefore reduced by a bit more than an order of magnitude. Left: WGAN algorithm. Right: standard GAN formulation. As we can see the standard GAN failed to learn while the WGAN still was able to produce samples.

нет mode collapse



Figure 7: Algorithms trained with an MLP generator with 4 layers and 512 units with ReLU nonlinearities. The number of parameters is similar to that of a DCGAN, but it lacks a strong inductive bias for image generation. Left: WGAN algorithm. Right: standard GAN formulation. The WGAN method still was able to produce samples, lower quality than the DCGAN, and of higher quality than the MLP of the standard GAN. Note the significant degree of mode collapse in the GAN MLP.

Wasserstein GAN: хорошая сходимость до ~min + WE коррелирует с видимым качеством

WGAN-GP

вместо обрезания – мягкий штраф на градиенты – «Gradient Penalty»

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

**но штраф хитрый – по точкам из отрезков с концами в разных распределениях
есть вывод, что это обеспечивает липшицевость**

нет critic batch normalization

устойчивость к выбору архитектуры

потом использовался в SOTA-моделях (Prog GAN, Style GAN и т.п.)

Gulrajani et al «Improved Training of Wasserstein GANs» // <https://arxiv.org/pdf/1704.00028.pdf>

DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)	
Baseline (G : DCGAN, D : DCGAN)				
G : No BN and a constant number of filters, D : DCGAN				
G : 4-layer 512-dim ReLU MLP, D : DCGAN				
No normalization in either G or D				
Gated multiplicative nonlinearities everywhere in G and D				
tanh nonlinearities everywhere in G and D				
101-layer ResNet G and D				

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.

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 .

```
1: while  $\theta$  has not converged do
2:   for  $t = 1, \dots, n_{\text{critic}}$  do
3:     for  $i = 1, \dots, m$  do
4:       Sample real data  $\mathbf{x} \sim \mathbb{P}_r$ , latent variable  $\mathbf{z} \sim p(\mathbf{z})$ , a random number  $\epsilon \sim U[0, 1]$ .
5:        $\tilde{\mathbf{x}} \leftarrow G_\theta(\mathbf{z})$ 
6:        $\hat{\mathbf{x}} \leftarrow \epsilon \mathbf{x} + (1 - \epsilon) \tilde{\mathbf{x}}$ 
7:        $L^{(i)} \leftarrow D_w(\tilde{\mathbf{x}}) - D_w(\mathbf{x}) + \lambda(\|\nabla_{\hat{\mathbf{x}}} D_w(\hat{\mathbf{x}})\|_2 - 1)^2$ 
8:     end for
9:      $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$ 
10:   end for
11:   Sample a batch of latent variables  $\{\mathbf{z}^{(i)}\}_{i=1}^m \sim p(\mathbf{z})$ .
12:    $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^m -D_w(G_\theta(\mathbf{z})), \theta, \alpha, \beta_1, \beta_2)$ 
13: end while
```

WGAN-GP: исследования на разных архитектурах

успех = inception_score > min_score

Table 1: WGAN-GP's ability to train the architectures in this set.

Nonlinearity (G) [ReLU, LeakyReLU, $\frac{\text{softplus}(2x+2)}{2} - 1$, tanh]Nonlinearity (D) [ReLU, LeakyReLU, $\frac{\text{softplus}(2x+2)}{2} - 1$, tanh]Depth (G) [4, 8, 12, 20]Depth (D) [4, 8, 12, 20]Batch norm (G) [True, False]Batch norm (D ; layer norm for WGAN-GP) [True, False]Base filter count (G) [32, 64, 128]Base filter count (D) [32, 64, 128]

Table 2: Outcomes of training 200 random architectures, for different success thresholds. For comparison, our standard DCGAN scored 7.24.

Min. score	Only GAN	Only WGAN-GP	Both succeeded	Both failed
1.0	0	8	192	0
3.0	1	88	110	1
5.0	0	147	42	11
7.0	1	104	5	90
9.0	0	0	0	200

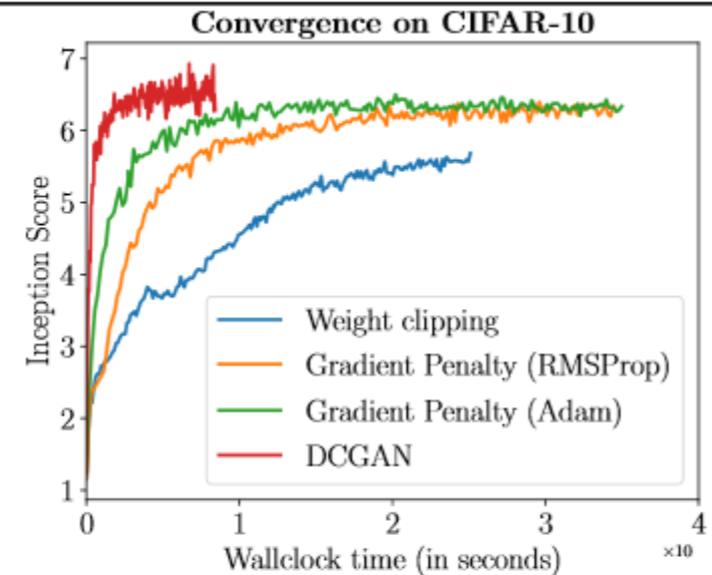
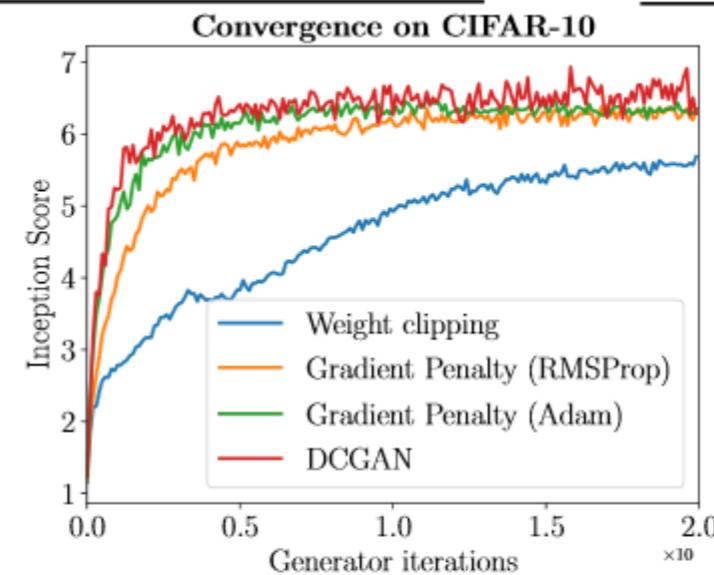


Figure 3: CIFAR-10 Inception score over generator iterations (left) or wall-clock time (right) for four models: WGAN with weight clipping, WGAN-GP with RMSProp and Adam (to control for the optimizer), and DCGAN. WGAN-GP significantly outperforms weight clipping and performs comparably to DCGAN.

Спектральная нормировка (Spectral Normalization): SN-GAN

$$\|A\|_2 = \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \alpha_{\max}(A)$$

**Здесь сингулярное число (а не собственное значение)
Есть результат: деление на спектральную норму тоже
гарантирует липшецевость**

Algorithm 1 SGD with spectral normalization

- Initialize $\tilde{u}_l \in \mathcal{R}^{d_l}$ for $l = 1, \dots, L$ with a random vector (sampled from isotropic distribution).
- For each update and each layer l :
 1. Apply power iteration method to a unnormalized weight W^l :

$$\tilde{v}_l \leftarrow (W^l)^T \tilde{u}_l / \| (W^l)^T \tilde{u}_l \|_2 \quad (20)$$

$$\tilde{u}_l \leftarrow W^l \tilde{v}_l / \| W^l \tilde{v}_l \|_2 \quad (21)$$

2. Calculate \bar{W}_{SN} with the spectral norm:

$$\bar{W}_{\text{SN}}^l(W^l) = W^l / \sigma(W^l), \text{ where } \sigma(W^l) = \tilde{u}_l^T W^l \tilde{v}_l \quad (22)$$

3. Update W^l with SGD on mini-batch dataset \mathcal{D}_M with a learning rate α :

$$W^l \leftarrow W^l - \alpha \nabla_{W^l} \ell(\bar{W}_{\text{SN}}^l(W^l), \mathcal{D}_M) \quad (23)$$

Maximum Mean Discrepancy (MMD)

Более общее понятие, чем EM-расстояние

$$W_F(p, p') = \sup_{f \in F} \left[\mathbb{E}_{x \sim p} f(x) - \mathbb{E}_{x \sim p'} f(x) \right]$$

и общее понятие из f-GAN

$$F(\theta, \omega) = \mathbb{E}_{x \sim P} [g_f(V_\omega(x))] + \mathbb{E}_{x \sim Q_\theta} [-f^*(g_f(V_\omega(x)))]$$

**здесь f^* – сопряжённая по Фенхелю (Fenchel conjugate) функции,
которая соответствует генератору (см. дальше)**

f-дивергенция:

$$D_f(p \parallel p') = \int p'(x) f\left(\frac{p(x)}{p'(x)}\right) dx$$

f-GAN**Базовые функции соответствуют разным дивергенциям**

Name	$D_f(P\ Q)$	Generator $f(u)$	$T^*(x)$
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} dx$	$u \log u$	$1 + \log \frac{p(x)}{q(x)}$
Reverse KL	$\int q(x) \log \frac{q(x)}{p(x)} dx$	$-\log u$	$-\frac{q(x)}{p(x)}$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)} dx$	$(u-1)^2$	$2(\frac{p(x)}{q(x)} - 1)$
Squared Hellinger	$\int \left(\sqrt{p(x)} - \sqrt{q(x)} \right)^2 dx$	$(\sqrt{u} - 1)^2$	$(\sqrt{\frac{p(x)}{q(x)}} - 1) \cdot \sqrt{\frac{q(x)}{p(x)}}$
Jensen-Shannon	$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} dx$	$-(u+1) \log \frac{1+u}{2} + u \log u$	$\log \frac{2p(x)}{p(x)+q(x)}$
GAN	$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} dx - \log(4)$	$u \log u - (u+1) \log(u+1)$	$\log \frac{p(x)}{p(x)+q(x)}$

Table 1: List of f -divergences $D_f(P\|Q)$ together with generator functions. Part of the list of divergences and their generators is based on [26]. For all divergences we have $f : \text{dom}_f \rightarrow \mathbb{R} \cup \{+\infty\}$, where f is convex and lower-semicontinuous. Also we have $f(1) = 0$ which ensures that $D_f(P\|P) = 0$ for any distribution P . As shown by [10] GAN is related to the Jensen-Shannon divergence through $D_{\text{GAN}} = 2D_{\text{JS}} - \log(4)$.

Sebastian Nowozin et al. «f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization» <https://arxiv.org/pdf/1606.00709.pdf>

f-GAN

(a) GAN

(b) KL

(c) Squared Hellinger

Figure 3: Samples from three different divergences.

GAN: проблемы

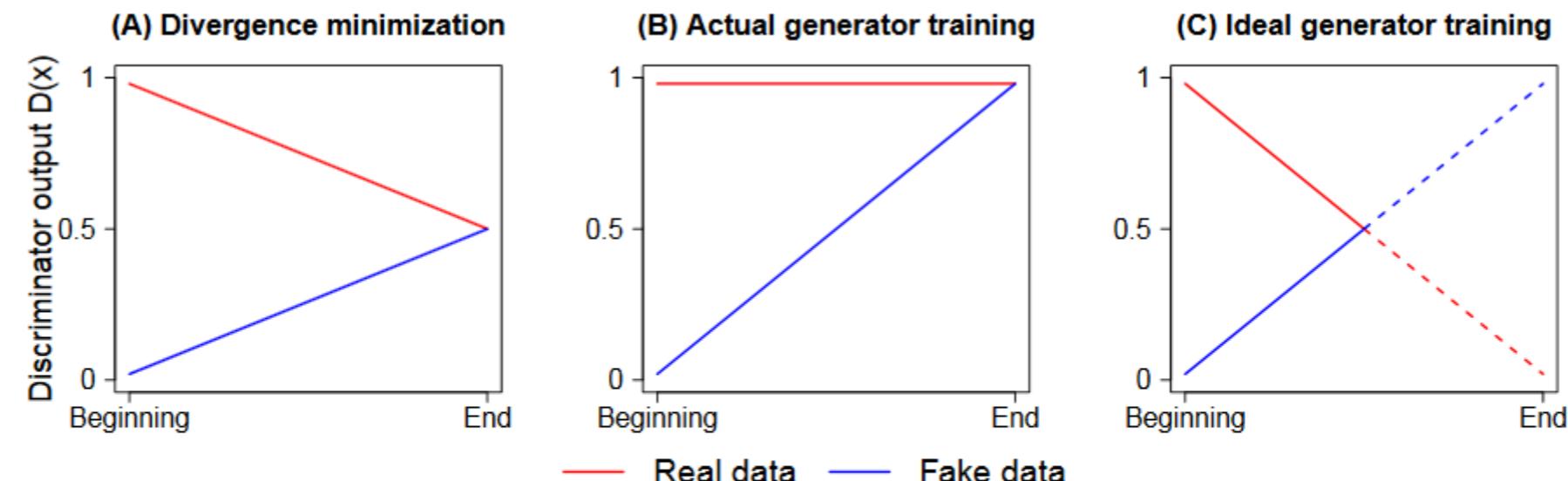


Figure 1: Expected discriminator output of the real and fake data for the a) direct minimization of the Jensen–Shannon divergence, b) actual training of the generator to minimize its loss function, and c) ideal training of the generator to minimize its loss function (lines are dotted when they cross beyond the equilibrium to signify that this may or may not be necessary).

Одна из попыток решения проблемы – Relativistic GANs, см. дальше

Alexia Jolicoeur-Martineau «The relativistic discriminator: a key element missing from standard GAN» // <https://arxiv.org/pdf/1807.00734.pdf>

Relativistic GANs (RGANs)

RSGAN = Relativistic standard

$$L_D^{RSGAN} = -\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [\log(\text{sigmoid}(C(x_r) - C(x_f)))]$$

$$L_G^{RSGAN} = -\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [\log(\text{sigmoid}(C(x_f) - C(x_r)))]$$

**здесь интерпретируем как оценку вероятности,
что один объект натуральнее другого**

В более общем случае

$$L_D^{RGAN*} = \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_r) - C(x_f))]$$

$$L_G^{RGAN*} = \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_f) - C(x_r))]$$

GAN: проблемы – Mode-Collapse

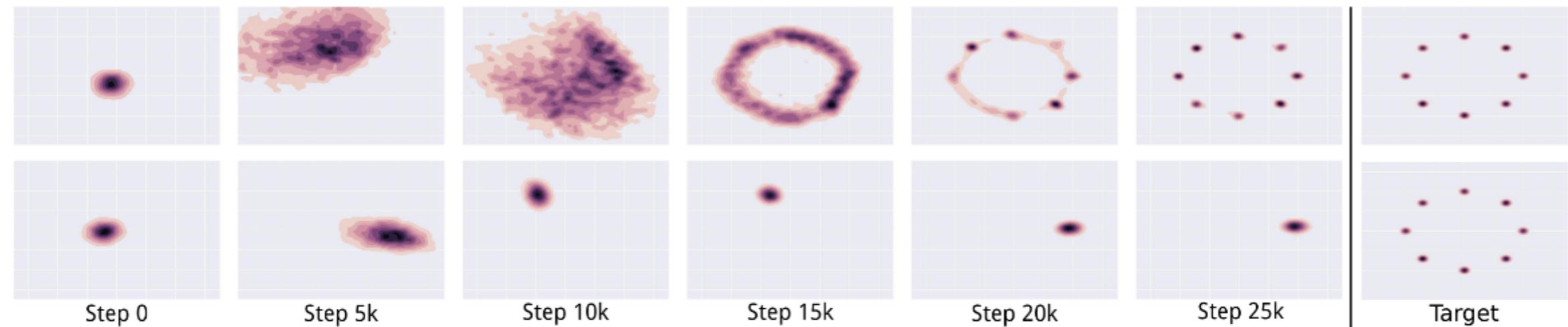


Figure 2: Unrolling the discriminator stabilizes GAN training on a toy 2D mixture of Gaussians dataset. Columns show a heatmap of the generator distribution after increasing numbers of training steps. The final column shows the data distribution. The top row shows training for a GAN with 10 unrolling steps. Its generator quickly spreads out and converges to the target distribution. The bottom row shows standard GAN training. The generator rotates through the modes of the data distribution. It never converges to a fixed distribution, and only ever assigns significant probability mass to a single data mode at once.

нижний ряд – «типичный GAN», нет разнообразия в ответах (sample diversity) см ниже

Metz, Luke, et al. «Unrolled Generative Adversarial Networks» //

<https://arxiv.org/pdf/1611.02163.pdf>

GAN: проблемы – Mode-Collapse

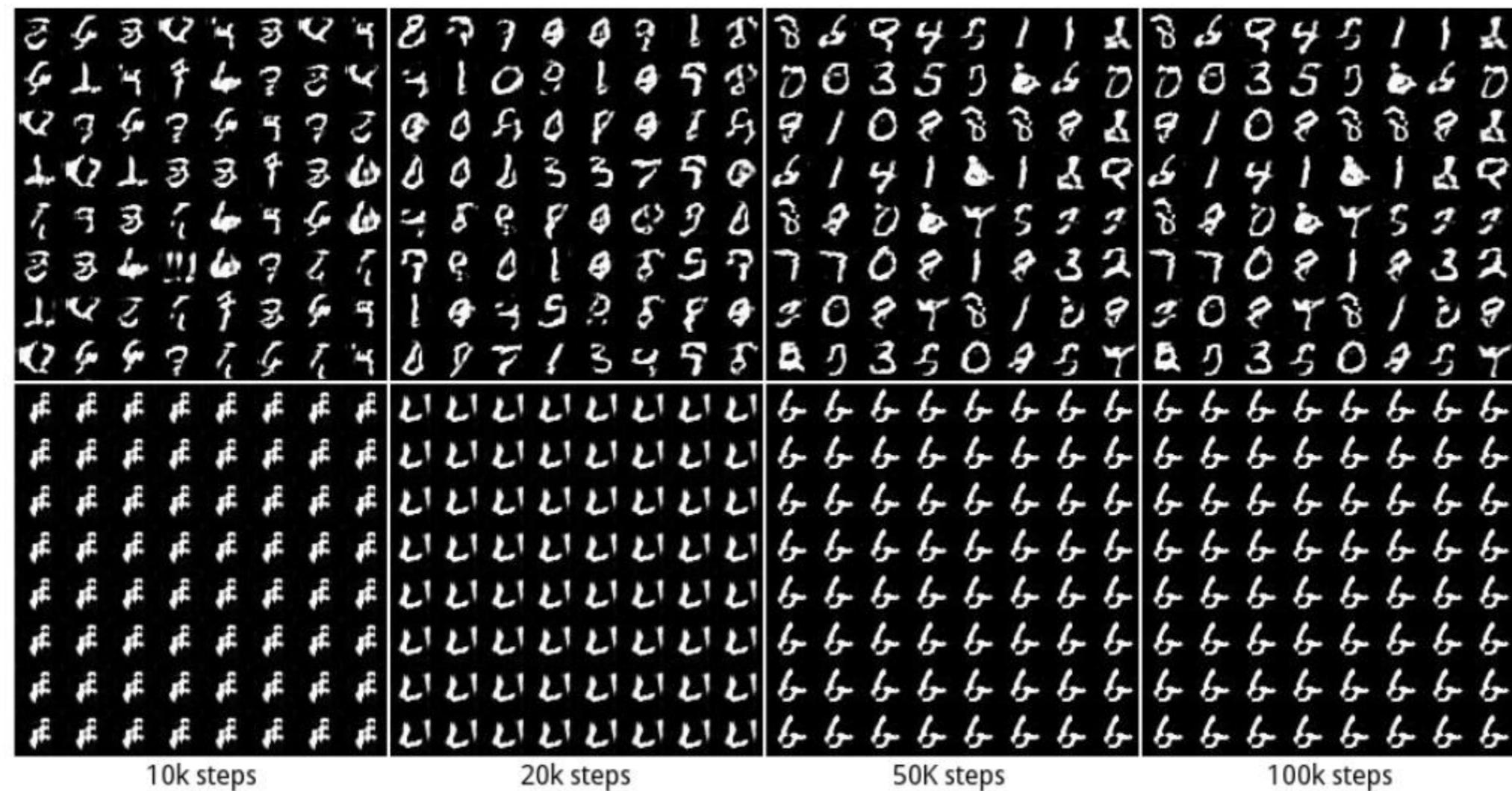


Figure 3: Unrolled GAN training increases stability for an RNN generator and convolutional discriminator trained on MNIST. The top row was run with 20 unrolling steps. The bottom row is a standard GAN, with 0 unrolling steps. Images are samples from the generator after the indicated number of training steps.

GAN: проблемы – Mode-Collapse

решение из статьи – Unrolled GAN

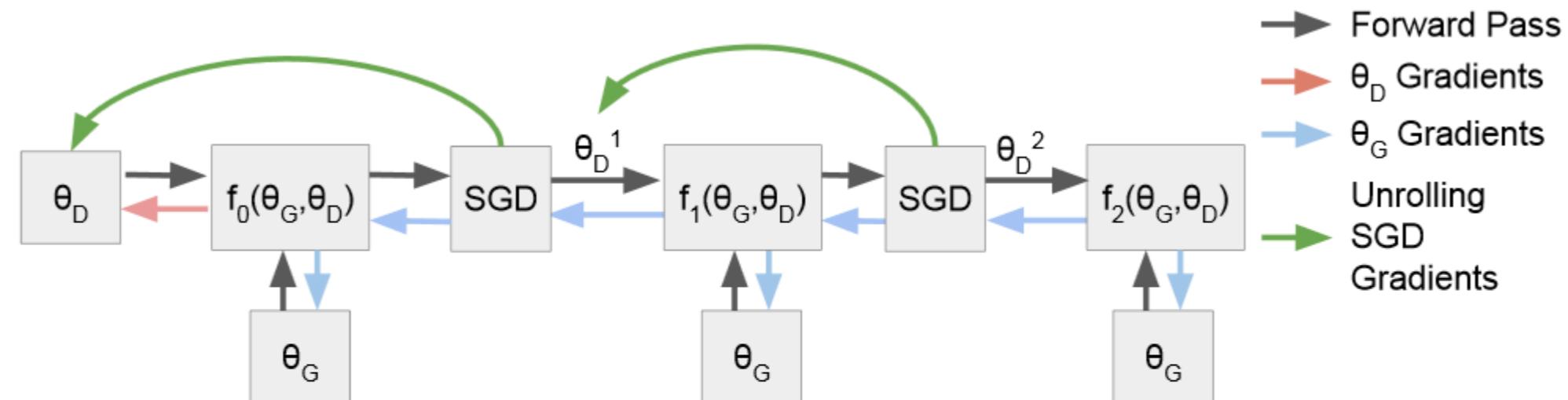


Figure 1: An illustration of the computation graph for an unrolled GAN with 3 unrolling steps. The generator update in Equation 10 involves backpropagating the generator gradient (blue arrows) through the unrolled optimization. Each step k in the unrolled optimization uses the gradients of f_k with respect to θ_D^k , as described in Equation 7 and indicated by the green arrows. The discriminator update in Equation 11 does not depend on the unrolled optimization (red arrow).

ещё решения: организация батчей, W-GANs, LSGANs (было выше) и т.п.
если батч примеров, то дискриминатор видит их однообразность
можно помочь – добавить специальные признаки,
ex: L2-норму разности каждой пары в батче

Energy-Based GAN (EBGAN) ■

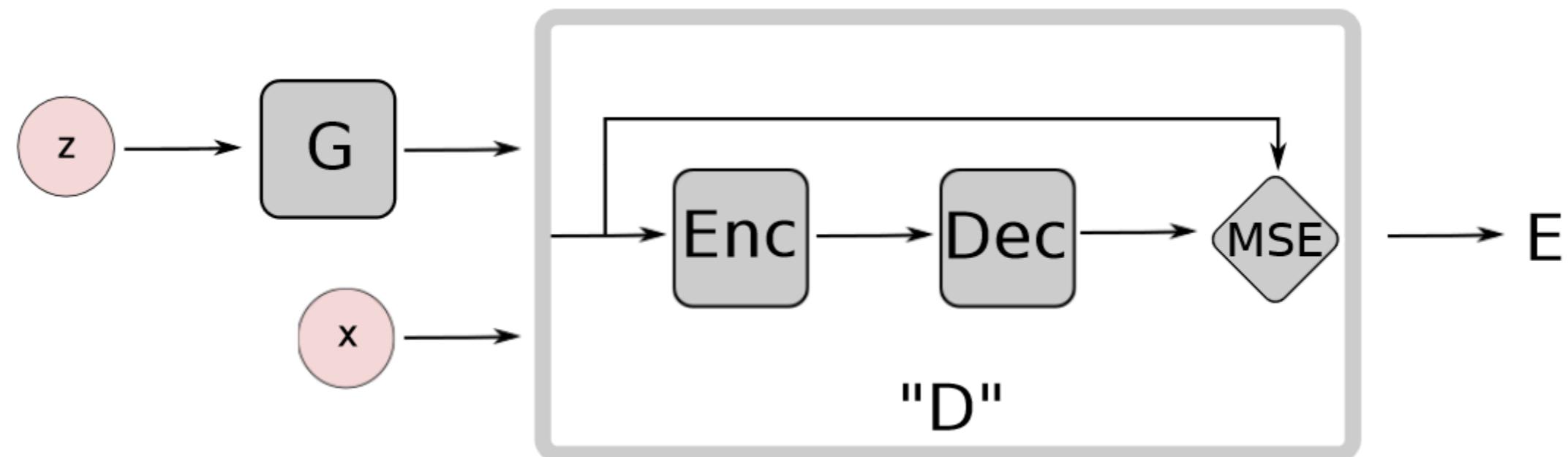


Figure 1: EBGAN architecture with an auto-encoder discriminator.

Пусть дискриминатор – автокодировщик, учим его на реальных данных

Если он хорошо восстанавливает сгенерированные данные, то данные хорошие

Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang «A tutorial on energy-based learning» // <http://yann.lecun.com/exdb/publis/pdf/lecun-06.pdf>

Energy-Based GAN (EBGAN) ■

может быть и такая функция ошибки

$$\mathcal{L}_D(x, z) = D(x) + [m - D(G(z))]^+,$$

$$\mathcal{L}_G(z) = D(G(z)),$$

Есть развитие идеи – Boundary Equilibrium GAN (BEGAN)

Как оценивать качество (сгенерированные картинки)

**Главная идея – сравнить распределения
(сэмплы из распределений)
настоящих объектов и фейков**

на каких-то признаках!

на пикселях не стоит...

например, на признаках, получаемыми обученными сетями

Как оценивать качество (сгенерированные картинки): Inception score (IS)

порождённые картинки → SOTA-классификатор $p(y | x)$
(Inception model, натренированный на ImageNet)

1) уверенная классификация

$p(y | x)$ – это распределение должно иметь низкую энтропию

$$H(y | x) = - \sum_y p(y | x) \log p(y | x)$$

2) большое разнообразие

$$p(y) = \int p(y | x = G(z)) p(z) dz$$

– тут должна быть большая энтропия (что мы научились генерировать)

$$H(y) = - \sum_y p(y) \log p(y)$$

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. «Improved techniques for training gans» // <https://arxiv.org/abs/1606.03498>

Как оценивать качество (сгенерированные картинки): Inception score (IS)

$$\begin{aligned}
 H(y) - \int H(y|x)p(x)\partial x &= -\sum_y p(y) \log p(y) + \sum_y \int p(y|x) \log p(y|x)p(x)\partial x \\
 &= \int \sum_y [-p(y|x)p(x) \log p(y) + p(y|x) \log p(y|x)p(x)]\partial x = \\
 &= \int \sum_y [p(y|x) \log \frac{p(y|x)}{p(y)}]p(x)\partial x = \mathbf{E}_x \sum_y \left[p(y|x) \log \frac{p(y|x)}{p(y)} \right]
 \end{aligned}$$

$$\text{IS}(p(x)) = \exp\left(\mathbf{E}_{x \sim p(x)} D_{KL}(p(y|x) \| p(y))\right)$$

- + коррелирует с человеческим восприятием
 - но нет учёта mode collapse
 - нет учёта распределения данных
- классификатор и генератор должны быть обучены на одном домене

Как оценивать качество (сгенерированные картинки): Mode score (MS)

улучшенная версия IS

$$\text{MS}(\mathbb{P}_g) = e^{\mathbb{E}_{\mathbf{x} \sim \mathbb{P}_g} [KL(p_{\mathcal{M}}(y|\mathbf{x}) || p_{\mathcal{M}}(y))] - KL(p_{\mathcal{M}}(y) || p_{\mathcal{M}}(y^*))}$$

здесь p – плотность сгенерированных данных, p^* – исходных
борьба с mode collapse

T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, “Mode regularized generative adversarial networks,” in International Conference on Learning Representations, 2017 //
<https://arxiv.org/pdf/1612.02136.pdf>

Kernel MMD (Maximum Mean Discrepancy)

$$\text{MMD}^2(\mathbb{P}_r, \mathbb{P}_g) = \mathbb{E}_{\substack{\mathbf{x}_r, \mathbf{x}'_r \sim \mathbb{P}_r, \\ \mathbf{x}_g, \mathbf{x}'_g \sim \mathbb{P}_g}} \left[k(\mathbf{x}_r, \mathbf{x}'_r) - 2k(\mathbf{x}_r, \mathbf{x}_g) + k(\mathbf{x}_g, \mathbf{x}'_g) \right]$$

для какого-то ядра k

+ хороша на предобученном ResNet-е

The Wasserstein distance

$$\text{WD}(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Gamma(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(\mathbf{x}^r, \mathbf{x}^g) \sim \gamma} [d(\mathbf{x}^r, \mathbf{x}^g)]$$

– долго вычислять, не рекомендуется использовать

The 1-Nearest Neighbor classifier

Для двух равномощных сэмплов из разных распределений вычислить LOO-CV-NN1 – это мера сходства распределений и смотреть на разницу точности на данных и фейках – это мера mode collapse
 + практически идеальная метрика

Как оценивать качество (сгенерированные картинки): Fréchet Inception Distance (FID)

сравнение активаций предобученной сети

Считает распределение реальных объектов и фейков гауссианами
на свёрточных признаках Inception network:

$$\begin{aligned} FID(p_{data}, p_g) = & \|\mu_r - \mu_g\| \\ & + \text{tr} \left(C_r + C_g - 2(C_r C_g)^{1/2} \right) \end{aligned}$$

Это расстояние Фреше (Wasserstein-2 distance) между гауссианами
при некоторых предположениях равенство распределений \sim равенство моментов
– но тут их всего два

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, «Gans trained by a two time-scale update ruleconverge to a local nash equilibrium» NIPS, pp. 6626–6637, 2017.
<https://papers.nips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf>

Fréchet Inception Distance (FID)

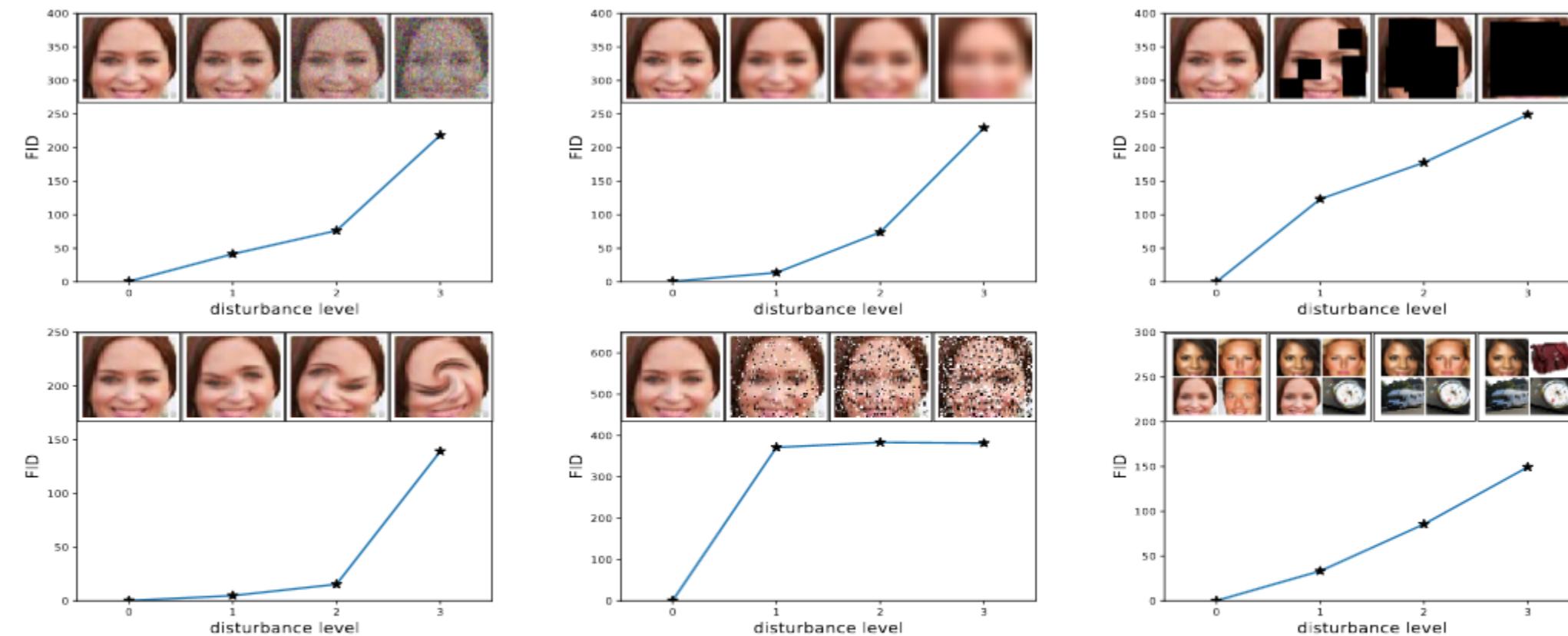


Figure 3: FID is evaluated for **upper left**: Gaussian noise, **upper middle**: Gaussian blur, **upper right**: implanted black rectangles, **lower left**: swirled images, **lower middle**: salt and pepper noise, and **lower right**: CelebA dataset contaminated by ImageNet images. The disturbance level rises from zero and increases to the highest level. The FID captures the disturbance level very well by monotonically increasing.

FID хорошо себя ведёт, а IS нет

Structural similarity (SSIM)

Для двух сигналов одинаковой длины

$$l(\mathbf{x}, \mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1},$$

$$c(\mathbf{x}, \mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2},$$

$$s(\mathbf{x}, \mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3},$$

1. symmetry: $\text{SSIM}(\mathbf{x}, \mathbf{y}) = \text{SSIM}(\mathbf{y}, \mathbf{x});$
2. boundedness: $\text{SSIM}(\mathbf{x}, \mathbf{y}) \leq 1;$
3. unique maximum: $\text{SSIM}(\mathbf{x}, \mathbf{y}) = 1$ if and only if $\mathbf{x} = \mathbf{y}$

C – маленькие константы

$$\text{SSIM}(\mathbf{x}, \mathbf{y}) = [l(\mathbf{x}, \mathbf{y})]^\alpha \cdot [c(\mathbf{x}, \mathbf{y})]^\beta \cdot [s(\mathbf{x}, \mathbf{y})]^\gamma$$

<https://ru.wikipedia.org/wiki/SSIM>

Multi-scale structural similarity (MSSIM)

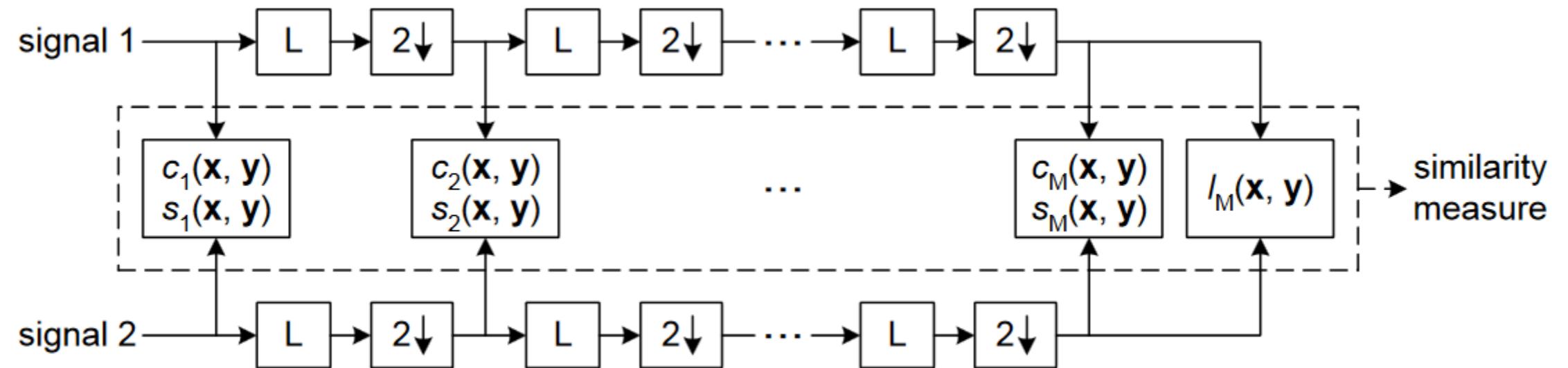


Fig. 1. Multi-scale structural similarity measurement system. L: low-pass filtering; $2 \downarrow$: downsampling by 2.

Z. Wang, E. P. Simoncelli, and A. C. Bovik, «Multiscale structural similarity for image quality assessment», in Asilomar Conference on Signals, Systems & Computers, vol. 2, pp. 1398–1402, 2003 <https://www.researchgate.net/publication/4071876 Multiscale structural similarity for image quality assessment>

Сравнение метрик качества

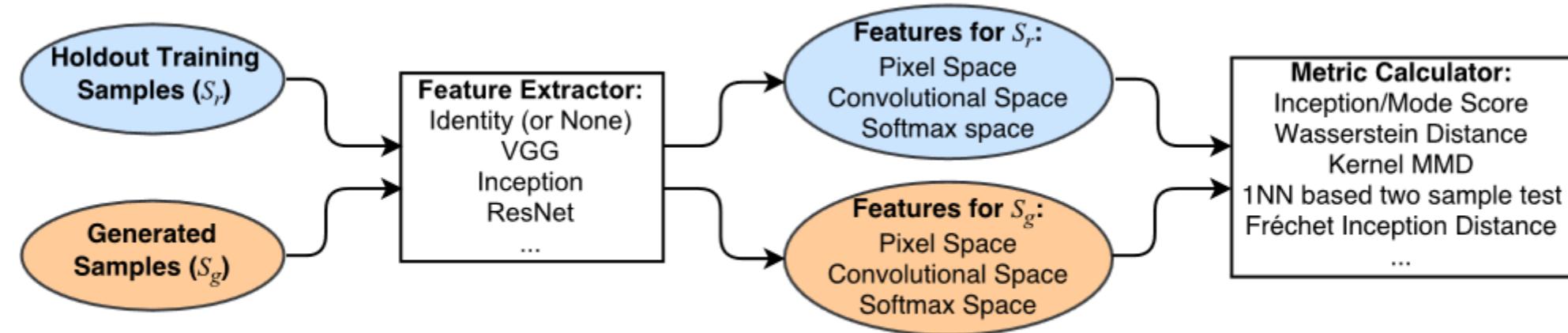


Figure 1: Typical sample based GAN evaluation methods.

Q. Xu, G. Huang, Y. Yuan, C. Guo, Y. Sun, F. Wu, and K. Weinberger «An empirical study on evaluation metrics of generative adversarial networks» // <https://arxiv.org/pdf/1806.07755.pdf>

Сравнение метрик качества

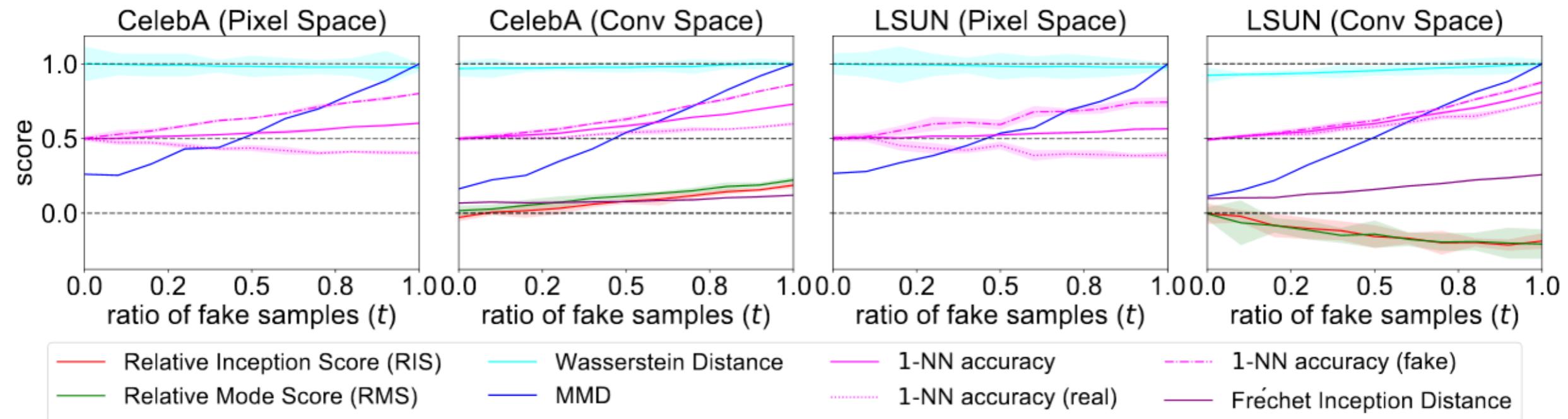


Figure 2: Distinguishing a set of real images from a mixed set of real images and GAN generated images. For the metric to be discriminative, its score should increase as the fraction of generated samples in the mix increases. RIS and RMS fail as they decrease with the fraction of generated samples in S_g on LSUN. Wasserstein and 1-NN accuracy (real) fail in pixel space as they do not increase.

здесь на разных пространствах: пикселях и свёрточных признаках

Сравнение метрик качества

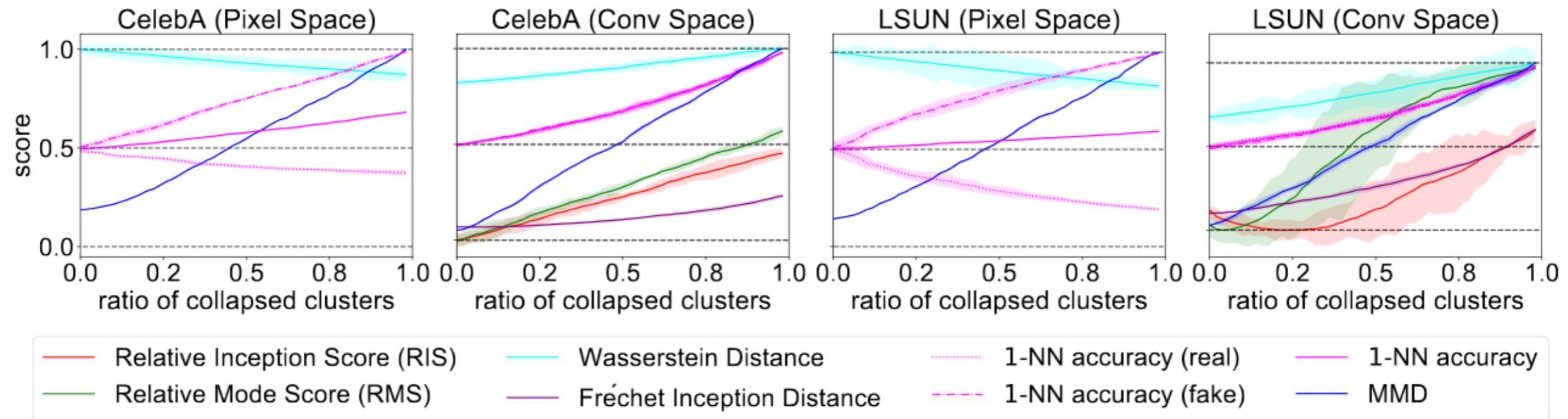


Figure 3: Experiment on simulated mode collapsing. A metric score should increase to reflect the mismatch between true distribution and generated distribution as more modes are collapsed towards their cluster center. All metrics respond correctly in convolutional space. In pixel space, both Wasserstein distance and 1-NN accuracy (real) fail as they decrease in response to more collapsed clusters.

Сравнение метрик качества

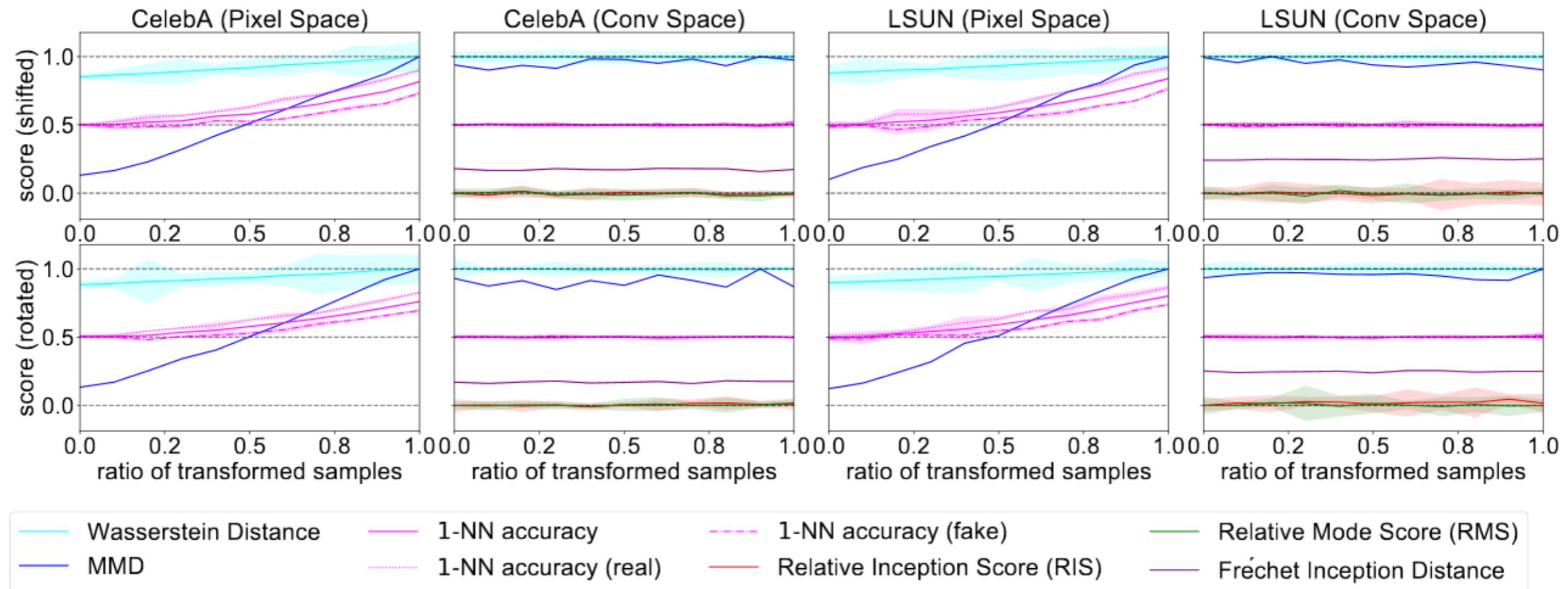


Figure 5: Experiment on robustness of each metric to small transformations (rotations and translations). All metrics should remain constant across all mixes of real and transformed real samples, since the transformations do not alter image semantics. All metrics respond correctly in convolutional space, but not in pixel space. This experiment illustrates the unsuitability of distances in pixel space.

Особенности GAN

**генерацию можно распараллелить
нельзя в авторегрессии**

визуально кажется лучше других методов

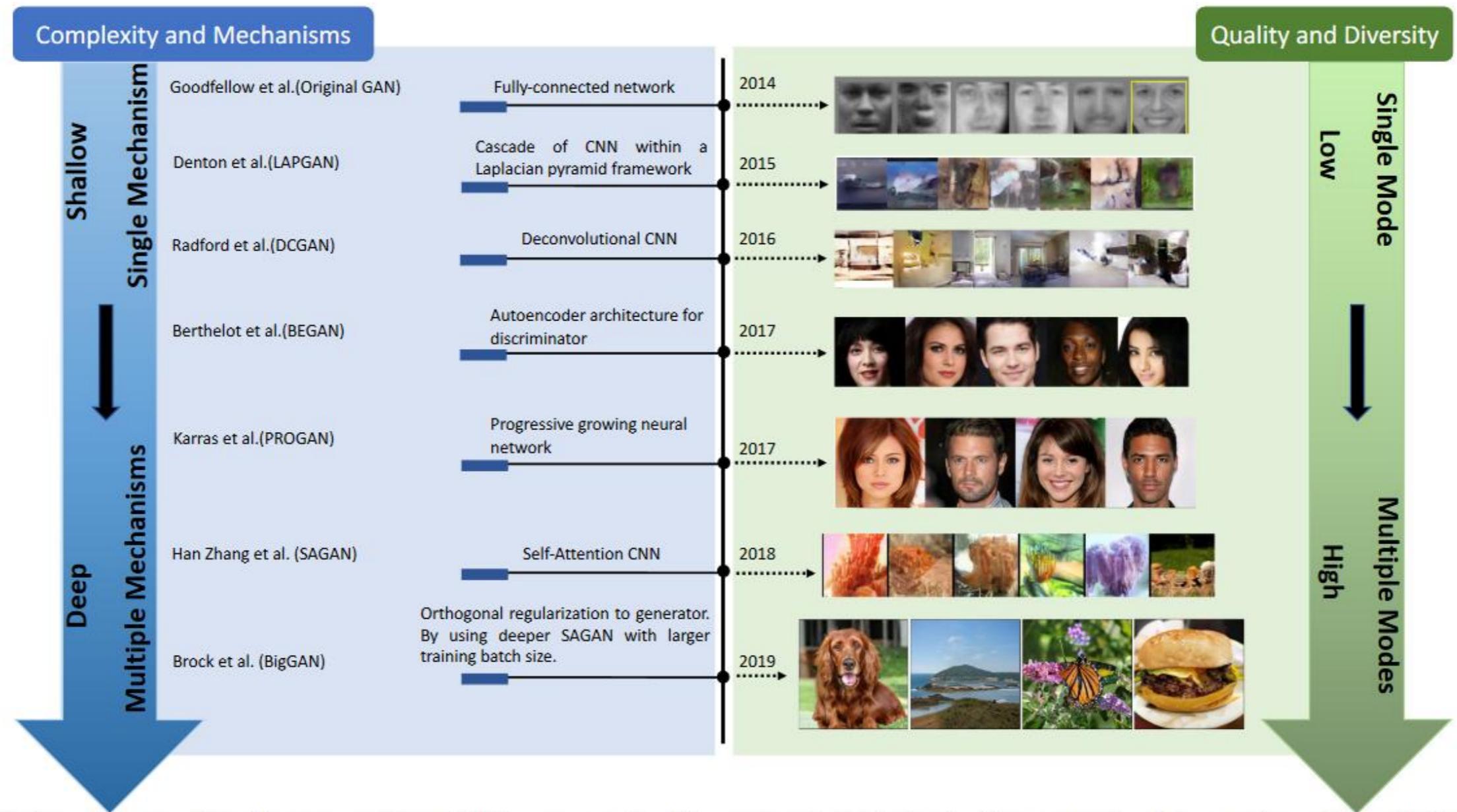


Fig. 3. Timeline of part of architecture-variant GANs present in this paper. Complexity in blue stream refers to size of the architecture and computational cost such as batch size. Mechanisms refer to the number of types of models used in the architecture (e.g., BEGAN uses an autoencoder architecture for its discriminator while a deconvolutional neural network is used for the generator. In this case, two mechanisms are used).

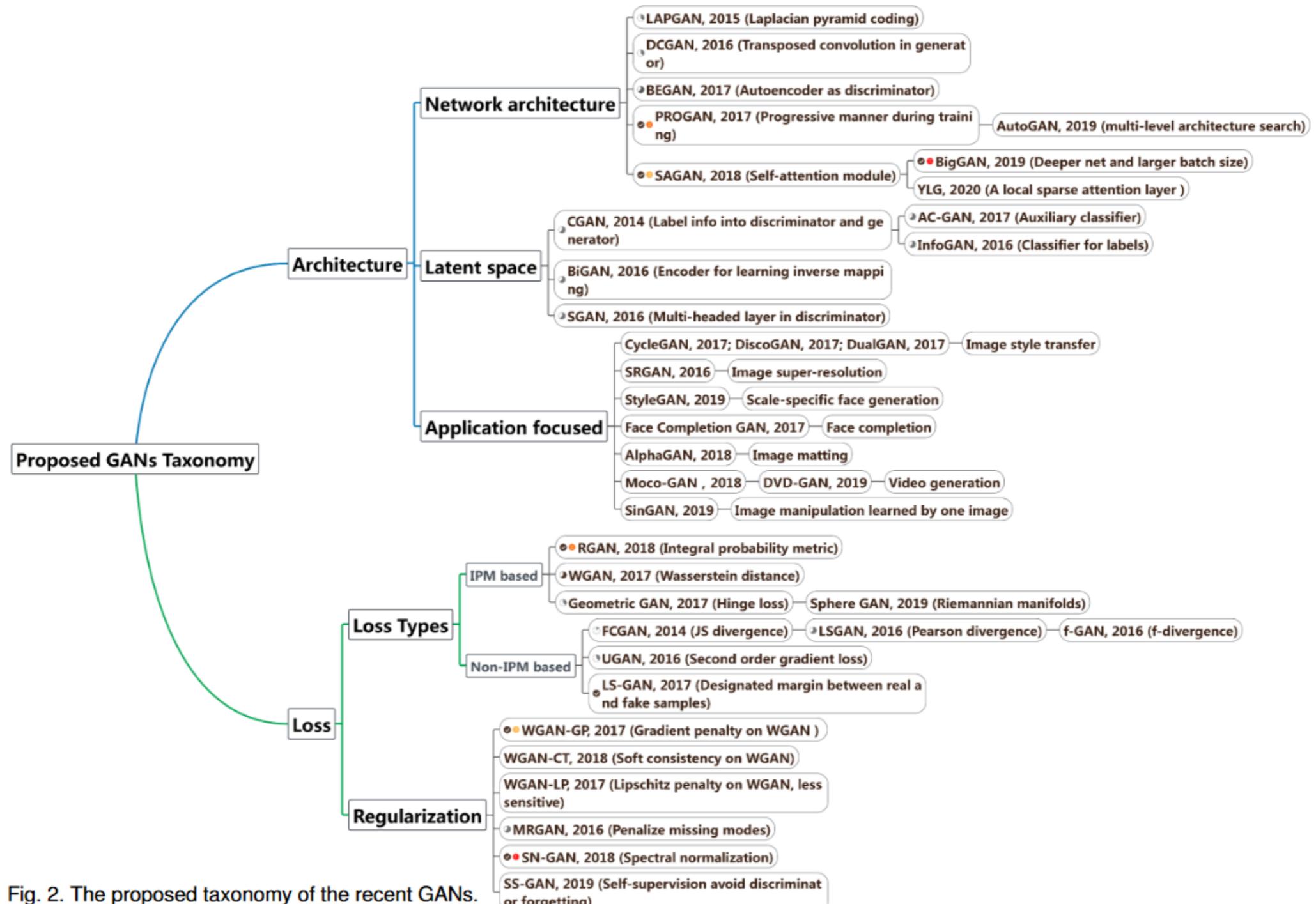


Fig. 2. The proposed taxonomy of the recent GANs.

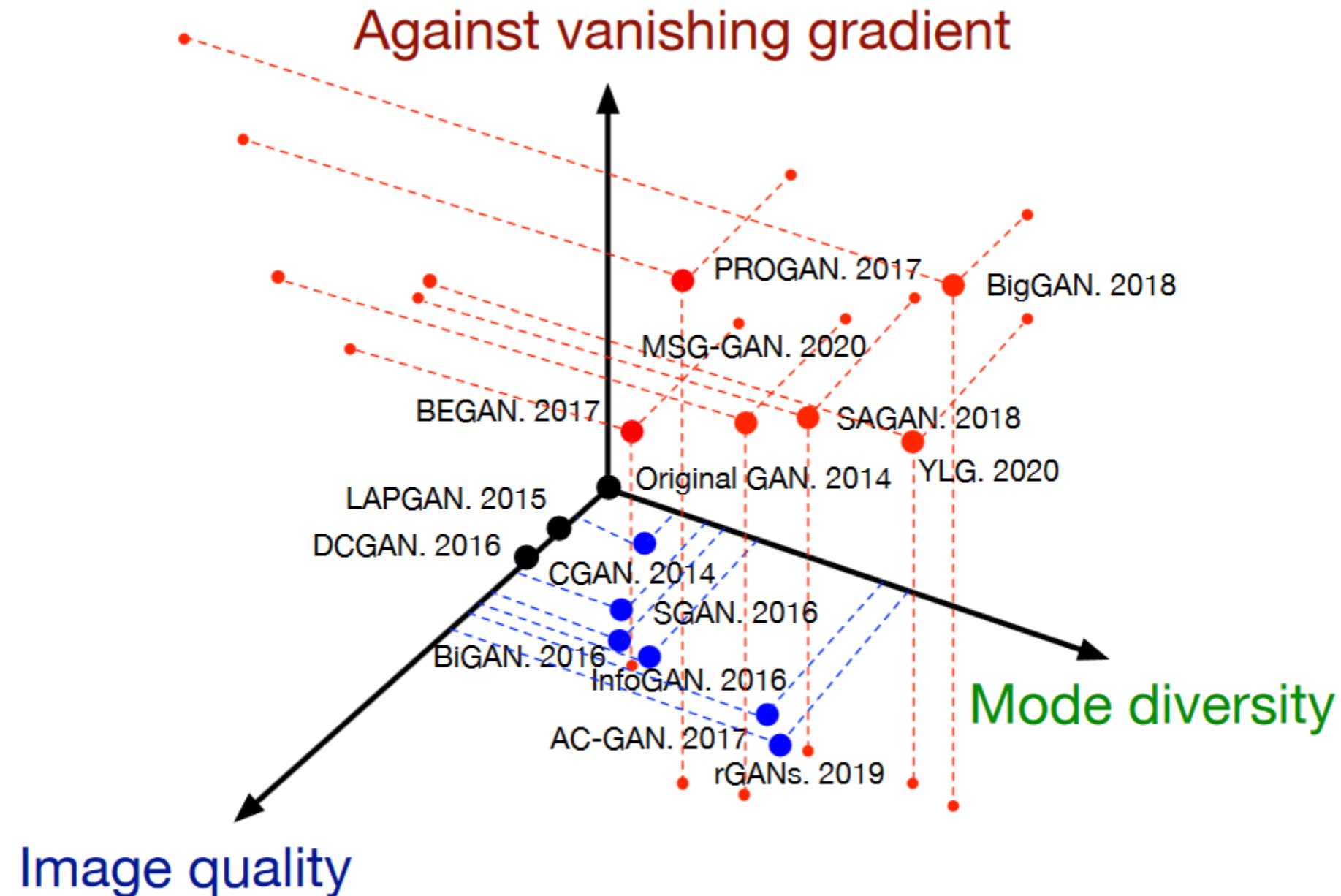


Fig. 18. Summary of recent architecture-variant GANs for solving the three challenges. The challenges are categorized by three orthogonal axes. A larger value for each axis indicates better performance. Red points indicate GAN-variants which cover all three challenges, blue points cover two, and black points cover only one challenge. Quantitative results can be referred to Table 5 in section 8.

Различные виды GAN

перечень видов со ссылками на первоисточники

<https://github.com/hindupuravinash/the-gan-zoo>

<https://github.com/wiseodd/generative-models>

<https://github.com/hwalsuklee/tensorflow-generative-model-collections>

Ian Goodfellow «NIPS 2016 Tutorial:Generative Adversarial Networks»

<https://arxiv.org/pdf/1701.00160.pdf>

Поиграться с GAN

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

Jie Gui et al «A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications» // <https://arxiv.org/pdf/2001.06937.pdf>