

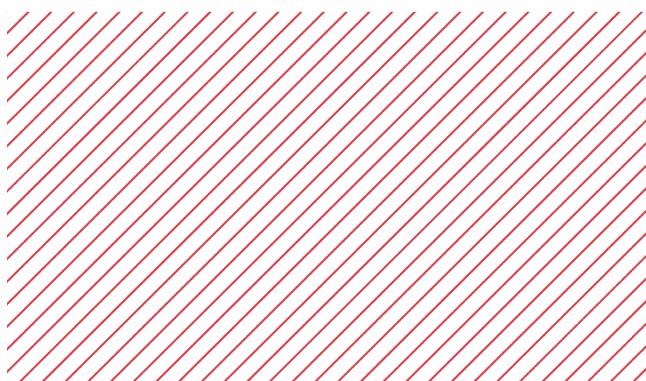
академия
больших
данных



Генеративно-состязательные нейронные сети. Продолжение

Фёдор Киташов

Программист-исследователь в команде
компьютерного зрения





План лекции

- ❖ Повторение прошлой лекции
- ❖ Метрики
- ❖ Статья Progressive Growing of GANs
- ❖ BigGAN
- ❖ StyleGAN и StyleGAN2



Повторение

- ❖ Статья “Generative Adversarial Nets” by I. Goodfellow
 - Функционал потерь и обучение
 - Проблема затухания градиентов
 - Mode collapse и JS дивергенция
- ❖ Wasserstein GAN
 - Earth Mover’s Distance
 - Непрерывность по Липшицу
 - Новая функция потерь с W_1 -дистанцией

Прогресс



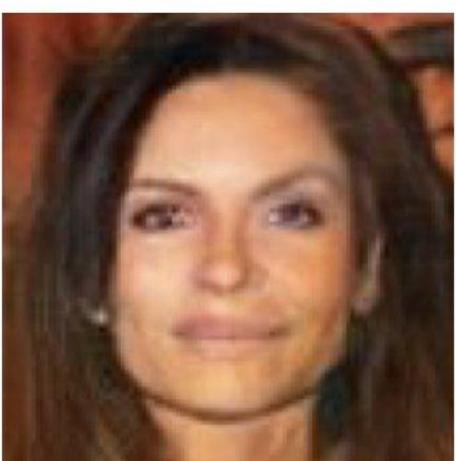
2014

GAN



2015

DCGAN



2016

CGAN



2017

PGGAN



2018

StyleGAN

Прогресс. На прошлой лекции



2014

GAN



2015

DCGAN



2016

CGAN



2017

PGGAN



2018

StyleGAN

Прогресс. На этой лекции



2014

GAN



2015

DCGAN



2016

CGAN



2017

PGGAN



2018

StyleGAN

Повторение

$$21 - (A + \eta)^2 + \kappa^2 \text{ and}$$



Архитектура: генератор и дискриминатор

- ❖ Генератор:
 - нейронная сеть
 - генерирует изображения из случайного шума



Архитектура: генератор и дискриминатор

- ❖ Генератор:
 - нейронная сеть
 - генерирует изображения из случайного шума

- ❖ Дискриминатор:
 - нейронная сеть
 - учится как бинарный классификатор
 - учится отличать настоящие изображения от сгенерированных



Обучение: minimax game

- ❖ Генератор G пытается сделать ❖ Дискриминатор D пытается сделать
 $D(G(z)) = 1$ $D(G(z)) = 0$



Обучение: minimax game

- ❖ Генератор G пытается сделать ❖ Дискриминатор D пытается сделать
 $D(G(z)) = 1$ $D(G(z)) = 0$

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

Обучение: minimax game

- ❖ Генератор G пытается сделать $D(G(z)) = 1$
- ❖ Дискриминатор D пытается сделать $D(G(z)) = 0$

Было:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

Стало:

$$D: \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

$$G: \max_G \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} \log(D(G(\mathbf{z})))$$

Обучение: minimax game

- ❖ Генератор G пытается сделать ❖ Дискриминатор D пытается сделать
 $D(G(z)) = 1$ $D(G(z)) = 0$

Было:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

Стало:

$$D: \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

$$G: \max_G \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} \log(D(G(\mathbf{z})))$$

Обучение: minimax game. Оптимальный D

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)}$$

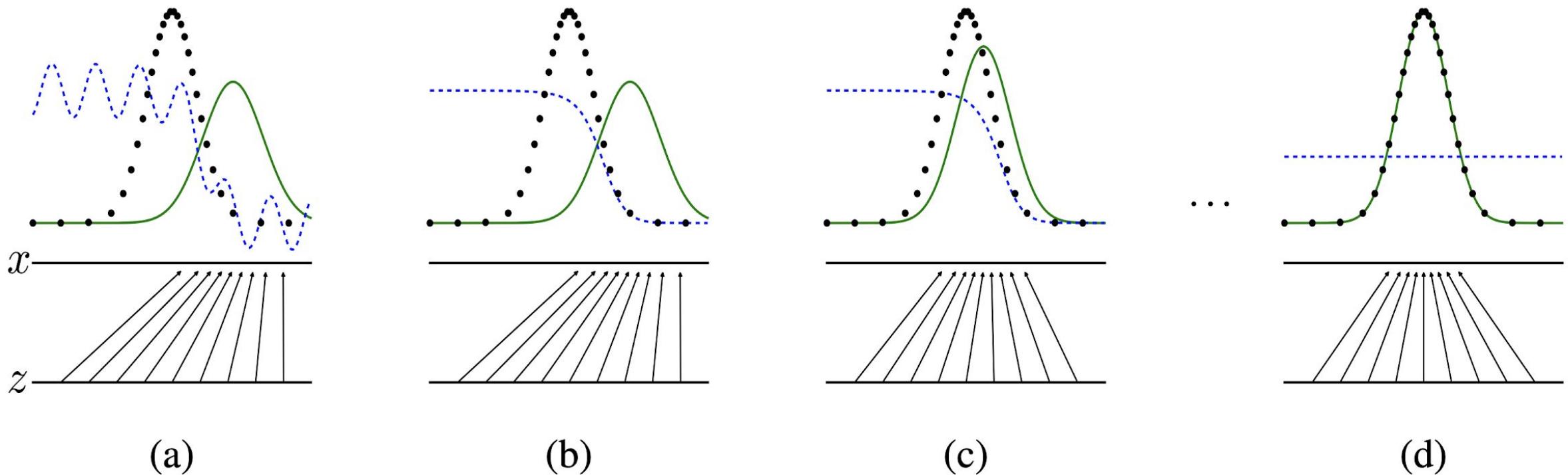
Обучение: minimax game. Оптимальный D

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)} = \frac{1}{2}$$

это происходит, когда $p_{\text{data}} = p_{\text{gen}}$

Обучение: minimax game.



$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)} = \frac{1}{2}$$

Оценка этого отношения с помощью обучения с учителем - главный механизм аппроксимации, который помогает генеративным сетям работать

DCGAN. Усовершенствование оригинальной статьи

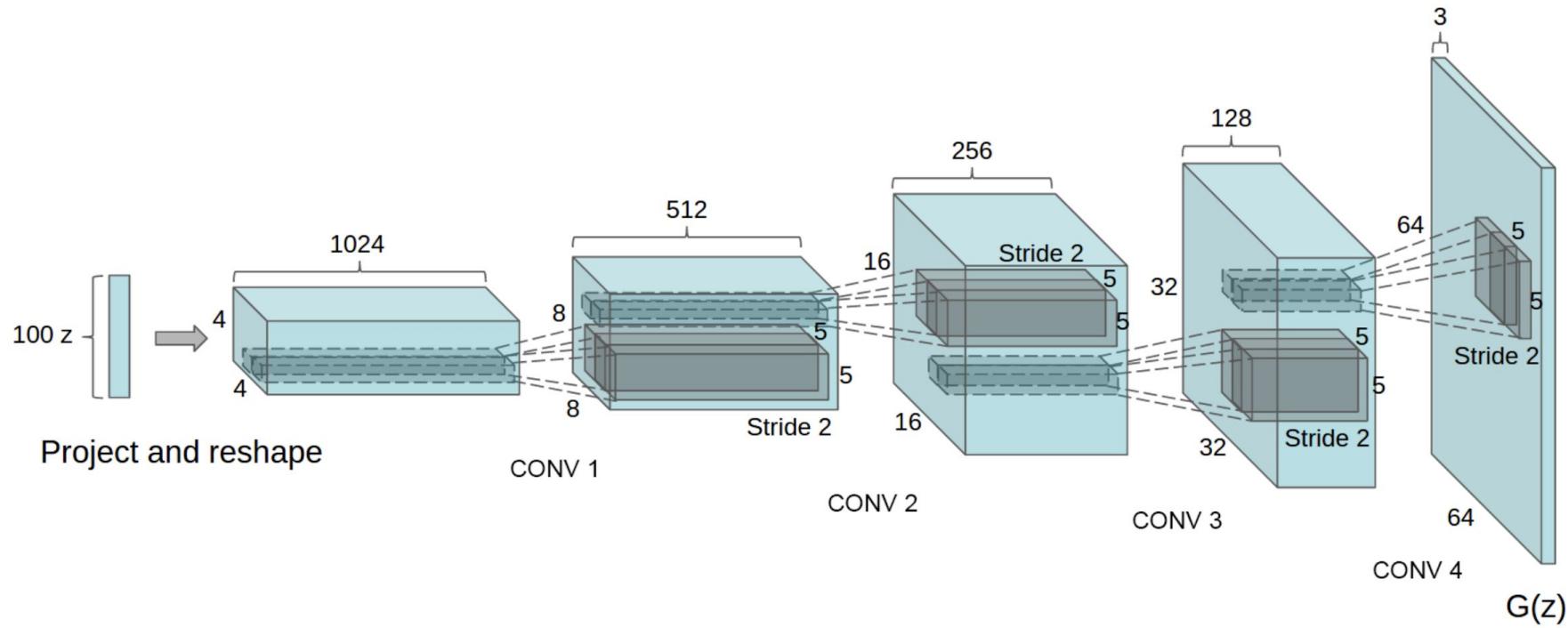


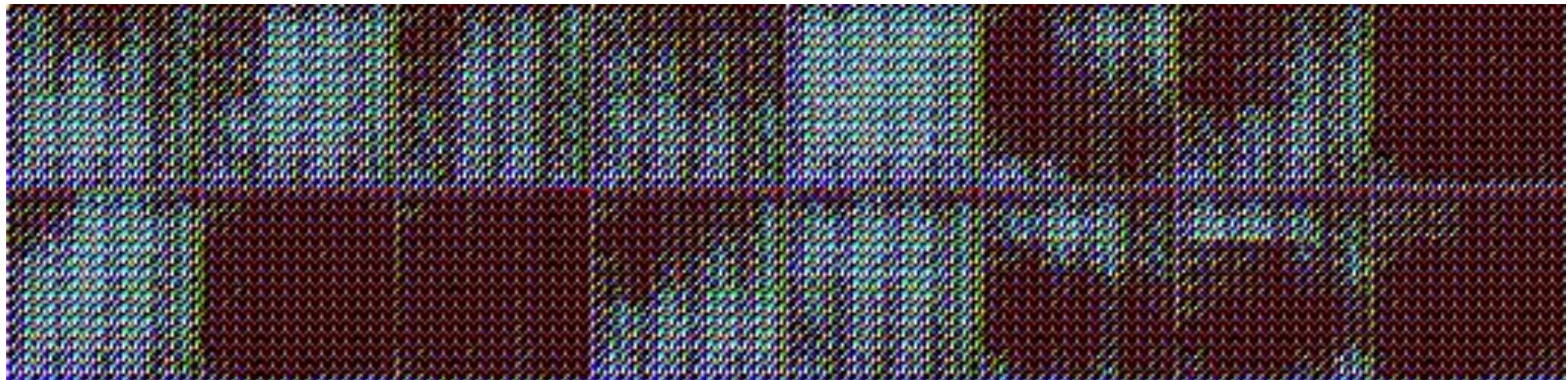
Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

DCGAN. Результаты работы на датасете LSUN

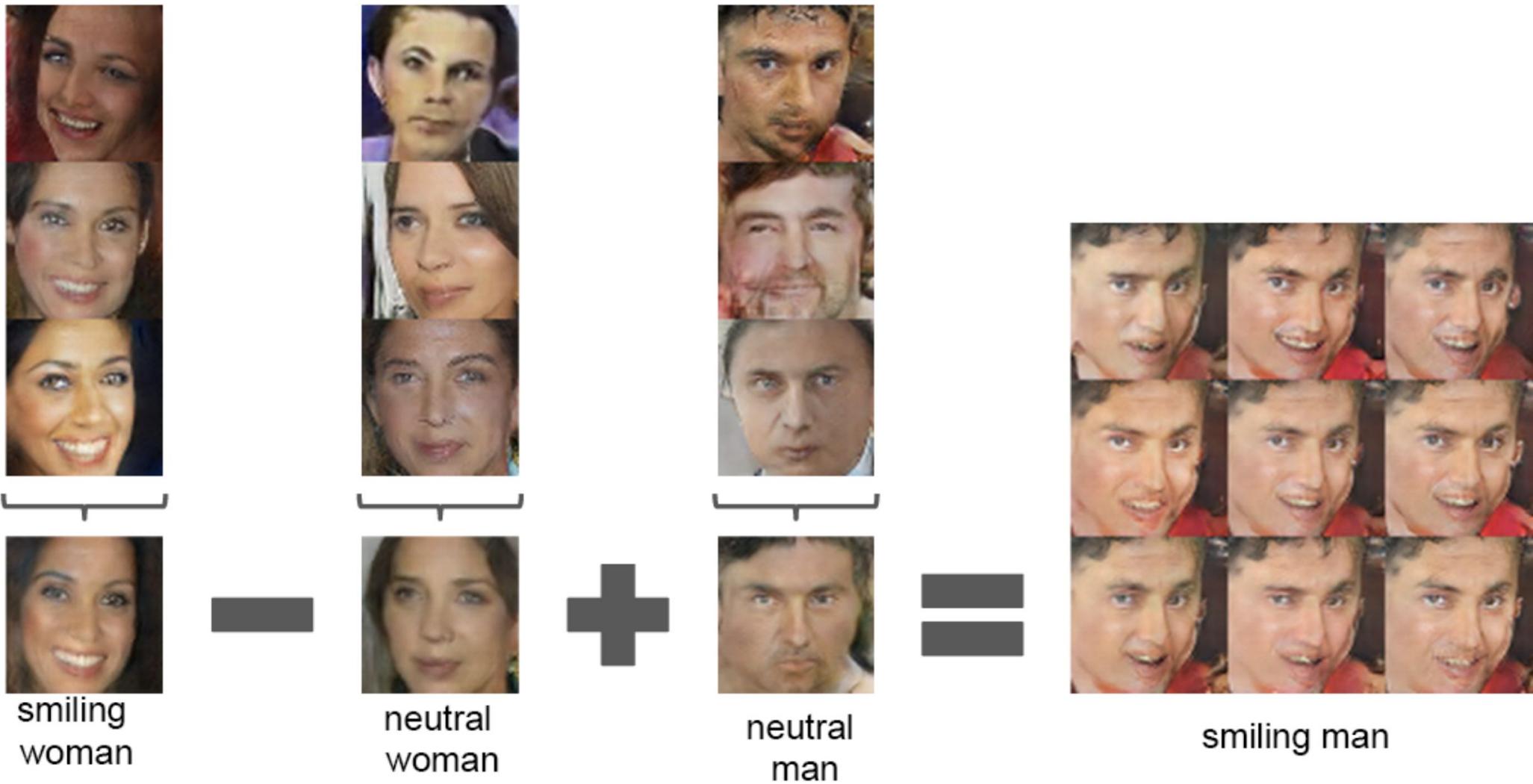




DCGAN. Результаты работы без BatchNorm



DCGAN. Векторная арифметика





Проблема идеального генератора

D: $\max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$

G: $\max_G \mathbb{E}_{z \sim p_z(z)} \log(D(G(z)))$

Если дискриминатор зафиксирован (не учится), то что нужно делать генератору, чтобы всегда обманывать дискриминатор?

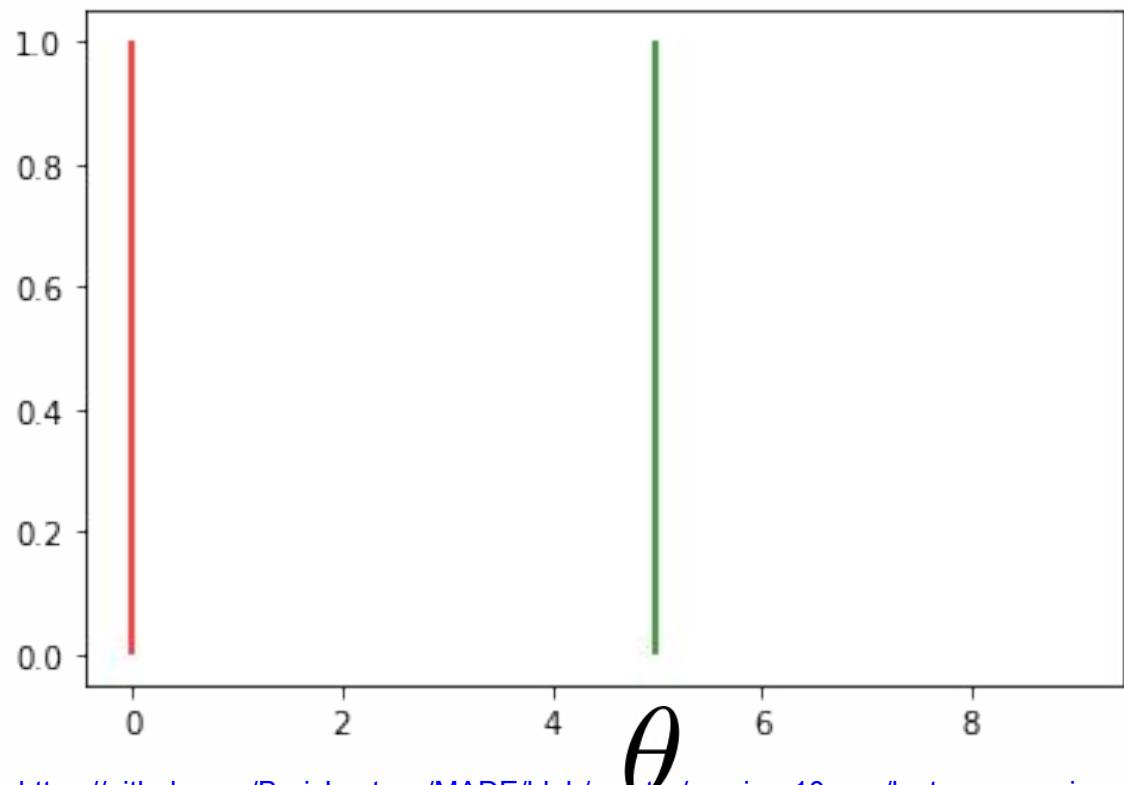
$$G(z) = \operatorname{argmax}_x D(x)$$





Проблемы KL и JS дивергенций. Пример

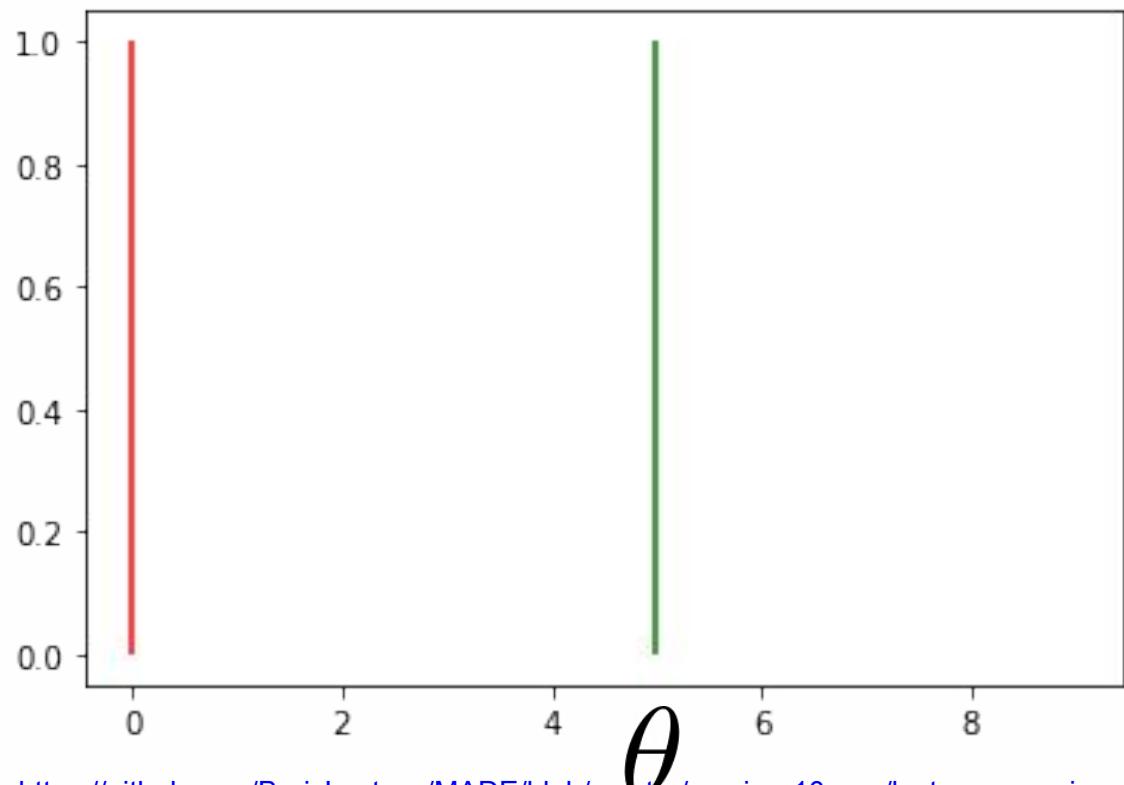
$$L(G, D) = -2 \log(2) + 2D_{JS}(p_r \parallel p_g)$$



$$D_{\text{KL}}(P \parallel Q) = \begin{cases} \infty & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$
$$JSD(P \parallel Q) = \begin{cases} \ln(2) & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

Проблемы KL и JS дивергенций. Пример

$$L(G, D) = -2 \log(2) + 2D_{JS}(p_r \parallel p_g)$$

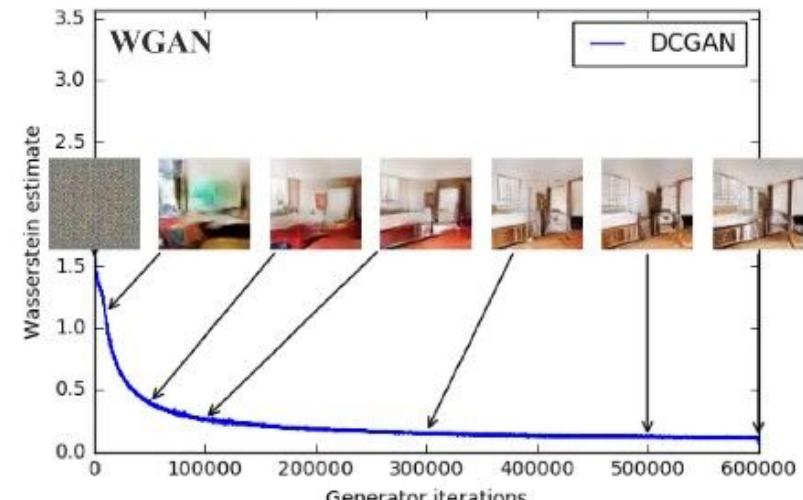
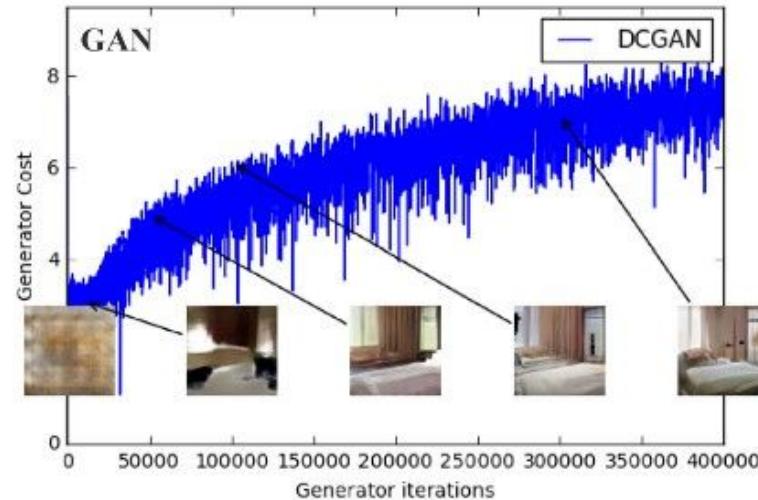
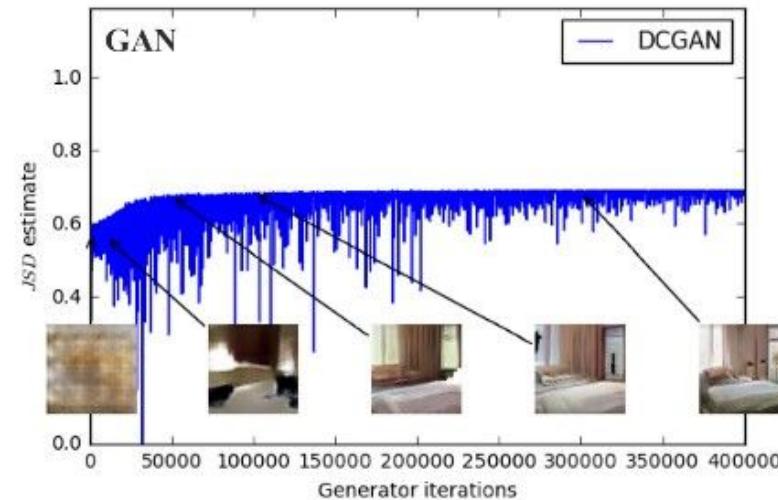


$$D_{\text{KL}}(P \parallel Q) = \begin{cases} \infty & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

$$JSD(P \parallel Q) = \begin{cases} \ln(2) & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

$$W_1(P \parallel Q) = \theta$$

Wasserstein GAN. Качество зависит от лосса





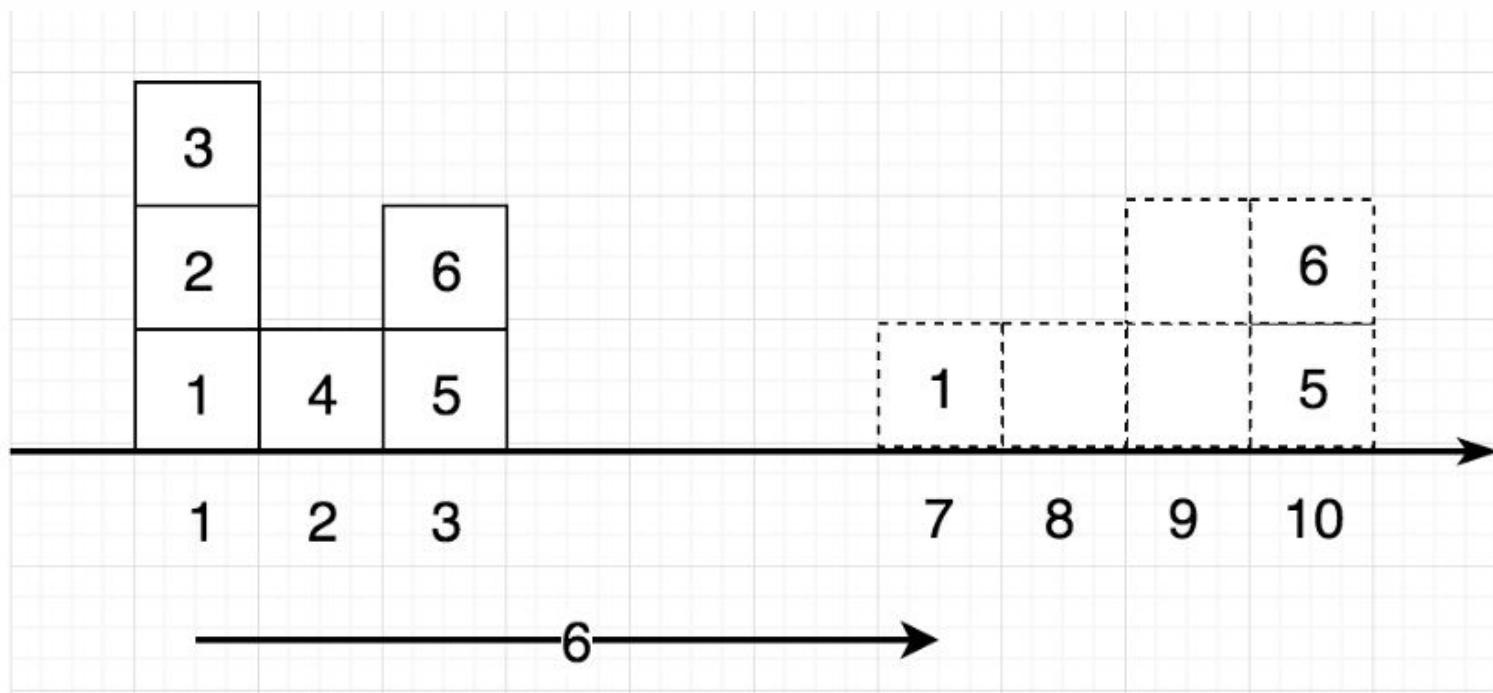
EMD: Earth mover's distance

- ❖ Метрика Вассерштейна (Wasserstein-1 distance, Earth Mover's Distance) - метрика, вычисляемая по формуле:

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

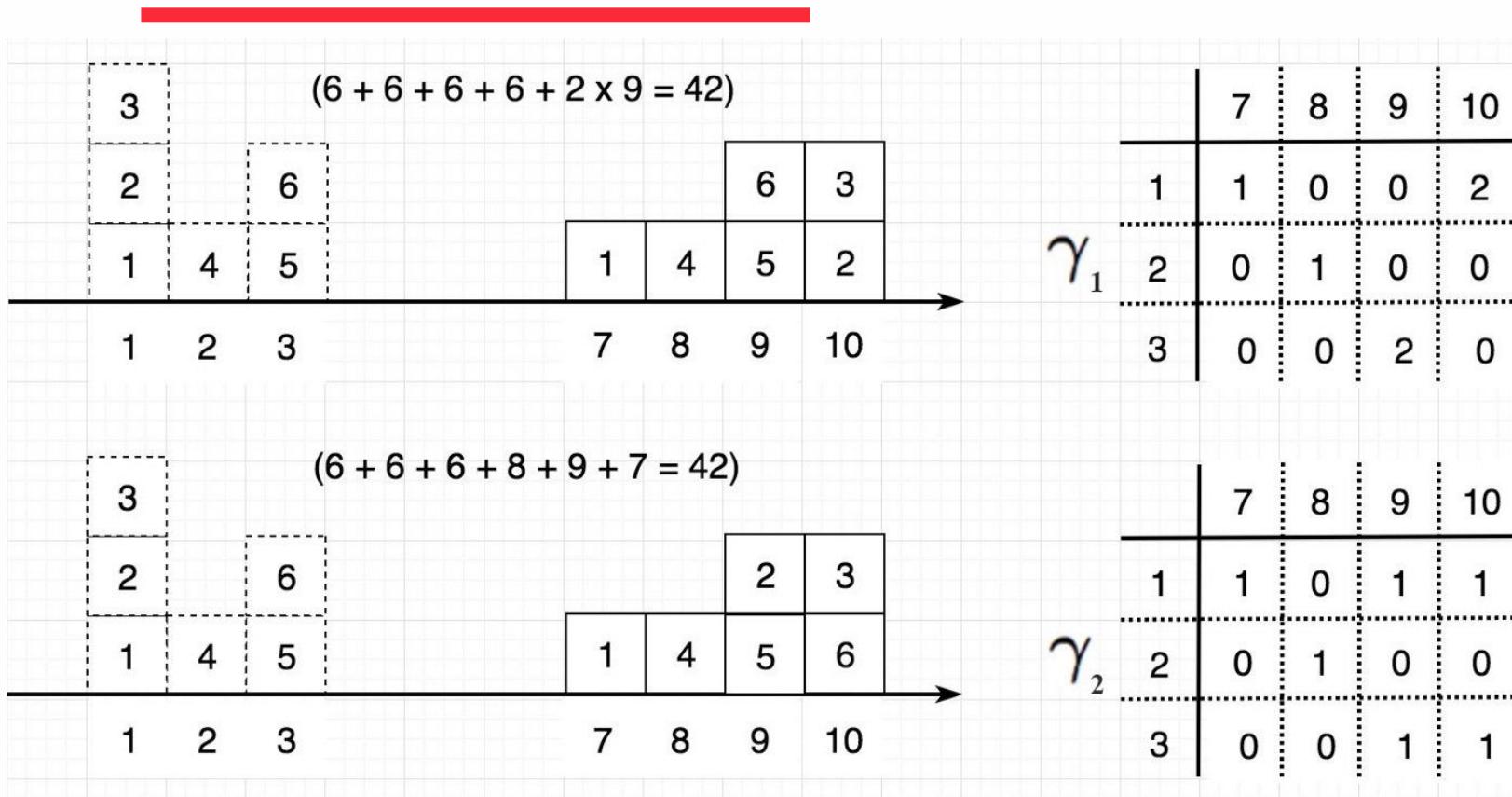
Также часто формулируется как минимальное количество энергии, которое нужно потратить, чтобы придать одной куче земли форму второй кучи земли

EMD: Earth mover's distance. Пример



- ❖ Хотим поменять форму кучи из шести ящиков оптимальным способом

EMD: Earth mover's distance. Пример



- ❖ Стратегий переноса может быть несколько.
- ❖ Обозначим γ_i i -ую стратегию переноса
- ❖ Дистанция Вассерштейна - это стоимость самой дешевой стратегии переноса



EMD: Earth mover's distance.

- ❖ Проблема: непонятно как считать градиенты по EMD
- ❖ Решение: можно переформулировать выражение EMD



EMD: Earth mover's distance.

- ❖ Проблема: непонятно как считать градиенты по EMD
- ❖ Решение: можно переформулировать выражение EMD

Было:

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

EMD: Earth mover's distance.

- ❖ Проблема: непонятно как считать градиенты по EMD
- ❖ Решение: можно переформулировать выражение EMD

Было:

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

Стало:

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r} [f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta} [f(x)]$$

Точная верхняя грань по всем функциям $f : \mathcal{X} \rightarrow \mathbb{R}$, непрерывных по Лишнику с константой меньшей или равной единице (Используется дуализм Канторовича-Рубинштейна)
<https://arxiv.org/pdf/1701.07875.pdf>

Идея: перейти от JS-дивергенции к W-1

Было: $L(G, D) = -2 \log(2) + 2D_{JS}(p_r \parallel p_g)$

Стало: $W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$

	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://arxiv.org/abs/1701.07875>

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

<https://www.alexirpan.com/2017/02/22/wasserstein-gan.html>

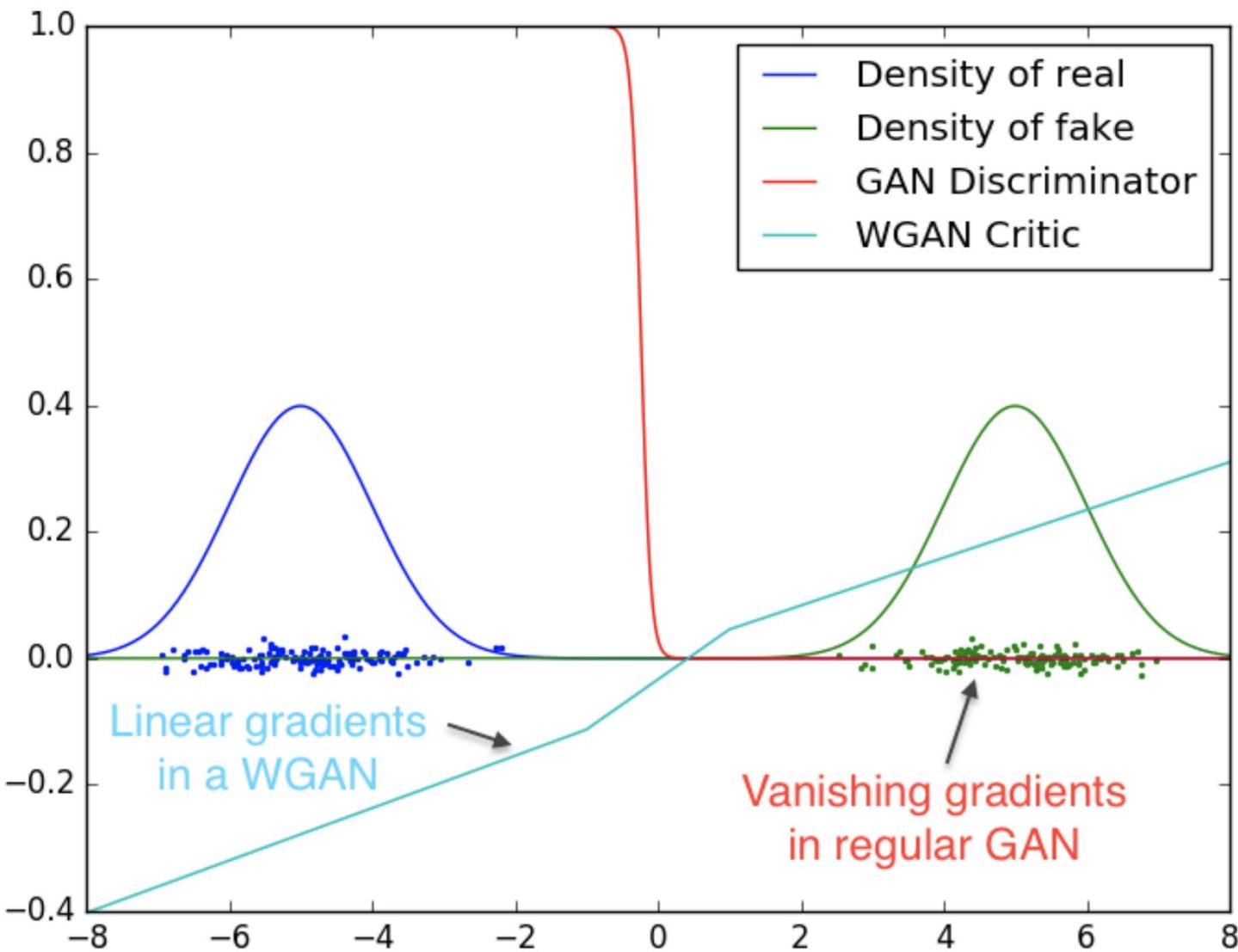


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians.

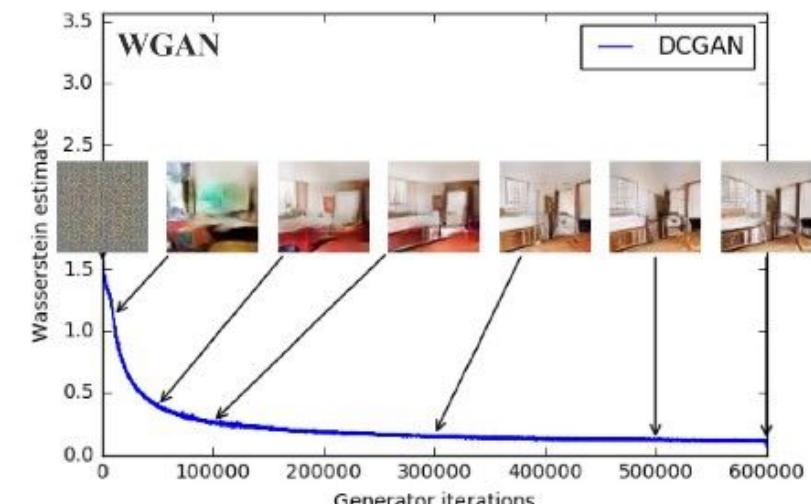
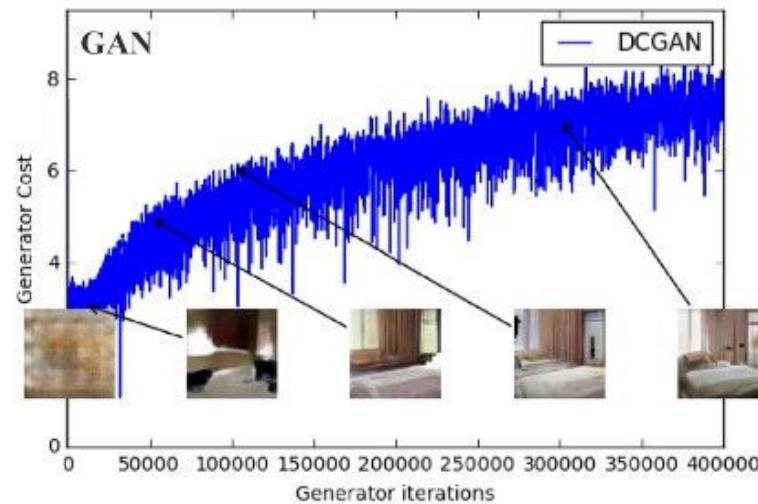
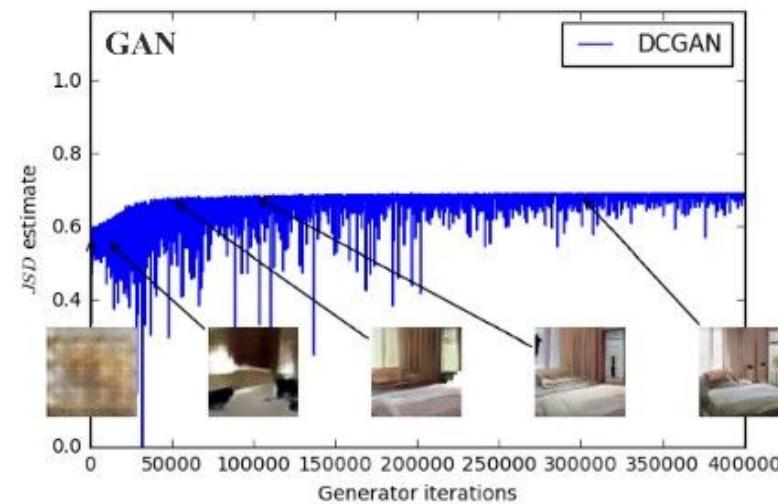
<https://arxiv.c> As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.



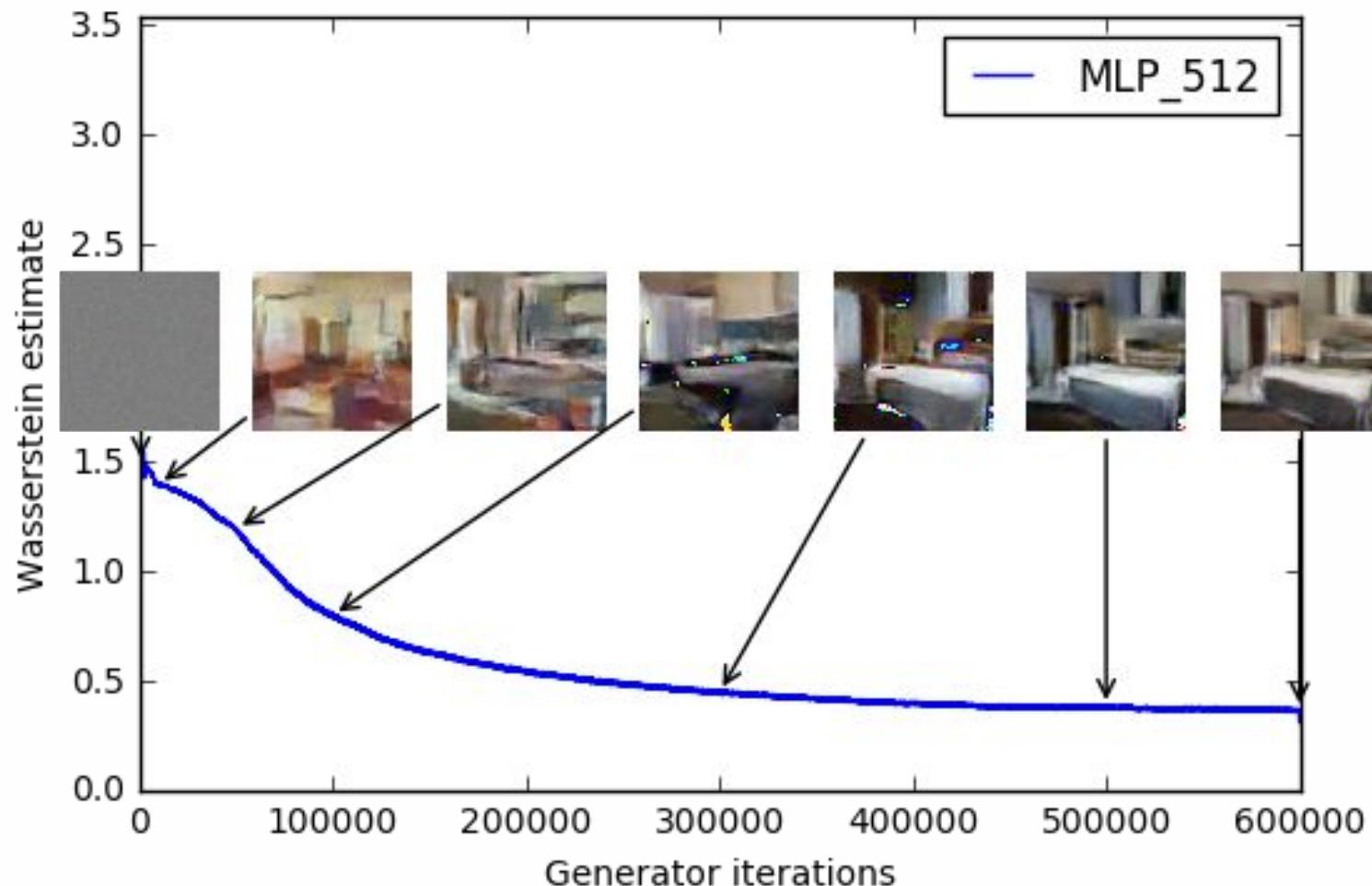
Способы достижения Липшиц-непрерывности

- ❖ Weight clipping - на каждой итерации обрезать веса до отрезка $[-c; c]$
$$w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$$
$$w \leftarrow \text{clip}(w, -c, c)$$
- ❖ Gradient Penalty (WGAN-GP) - в лосс добавляется отличие нормы градиента от единицы
- ❖ Спектральная нормализация - делим каждый элемент матрицы весов на её спектральную норму
 - Спектральная норма - максимальное сингулярное число матрицы весов.

Wasserstein GAN. Качество зависит от лосса



Wasserstein GAN. Качество зависит от лосса



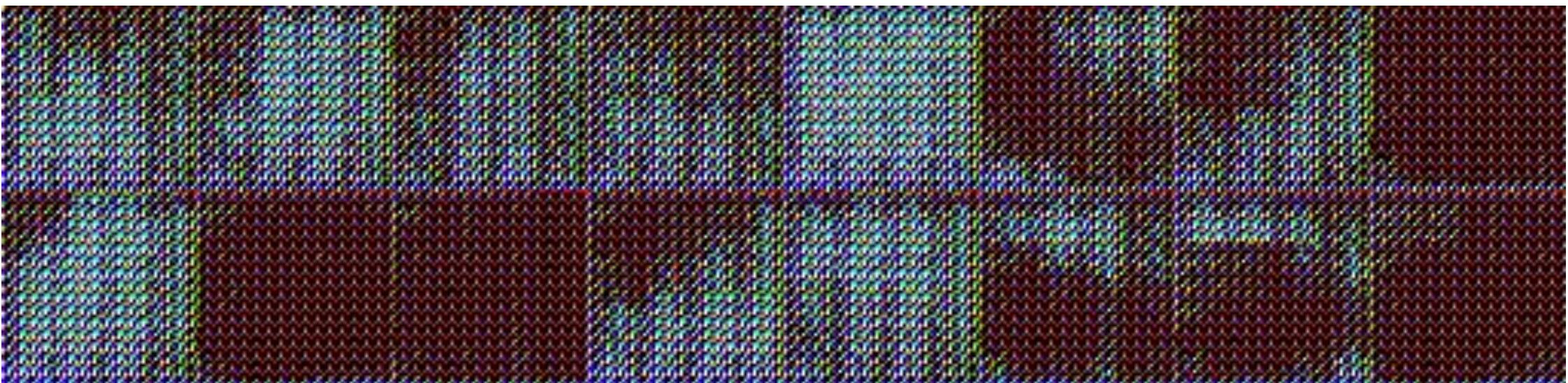
Wasserstein GAN. Сравнение с DCGAN



Верхняя картинка: WGAN с архитектурой DCGAN. Нижняя - DCGAN

<https://arxiv.org/abs/1701.07875>

Wasserstein GAN без BatchNorm



Верхняя картинка: WGAN с архитектурой DCGAN без BN. Нижняя - DCGAN без BN

<https://arxiv.org/abs/1701.07875>



Что будет дальше

- ❖ Метрики
- ❖ Progressive Growing of GANs:
 - люди научились генерировать большие реалистичные изображения
- ❖ BigGAN
 - Соберём все удачные хаки вместе и улучшим результат
- ❖ StyleGAN и StyleGAN2
 - посмотрим на архитектурные решения и результаты



Метрики

- ❖ Проблема: хотим измерить, насколько реалистичны изображения



Метрики

- ❖ Проблема: хотим измерить, насколько реалистичны изображения
- ❖ Решение:
 - Показывать людям результаты
 - Но это долго, дорого, ...



Метрики

- ❖ Проблема: хотим измерить, насколько реалистичны изображения
- ❖ Решение:
 - Показывать людям результаты
 - Но это долго, дорого, ...
 - Inception score
 - Frechet Inception Distance



Inception score

- ❖ Оценивает реалистичность сгенерированной выборки
- ❖ Хорошо коррелирует с оценкой людей



Inception score

- ❖ Оценивает реалистичность сгенерированной выборки
- ❖ Хорошо коррелирует с оценкой людей
- ❖ Предположение:
 - Пусть на каждом фото есть только один объект
 - Выборка сбалансирована



Inception score

- ❖ Оценивает реалистичность сгенерированной выборки
- ❖ Хорошо коррелирует с оценкой людей
- ❖ Предположение:
 - Пусть на каждом фото есть только один объект
 - Выборка сбалансирована
- ❖ Метрика одновременно измеряет, насколько
 - Изображения разнообразны
 - На каждом изображении объект различим

Inception score

- ❖ Оценивает реалистичность сгенерированной выборки
- ❖ Хорошо коррелирует с оценкой людей
- ❖ Предположение:
 - Пусть на каждом фото есть только один объект
 - Выборка сбалансирована
- ❖ Метрика одновременно измеряет, насколько
 - Изображения разнообразны
 - На каждом изображении объект различим





Метрики. Inception score

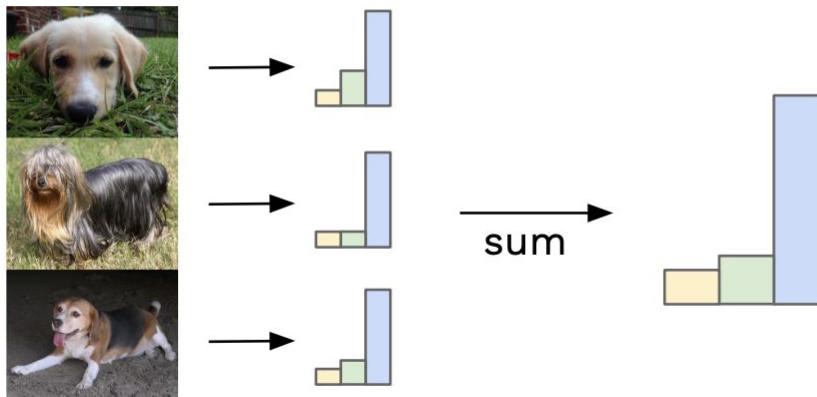
- ❖ Алгоритм:
 - Генерируем K изображений N классов.
 - Выборка сбалансирована: каждый класс - K/N изображений

Метрики. Inception score

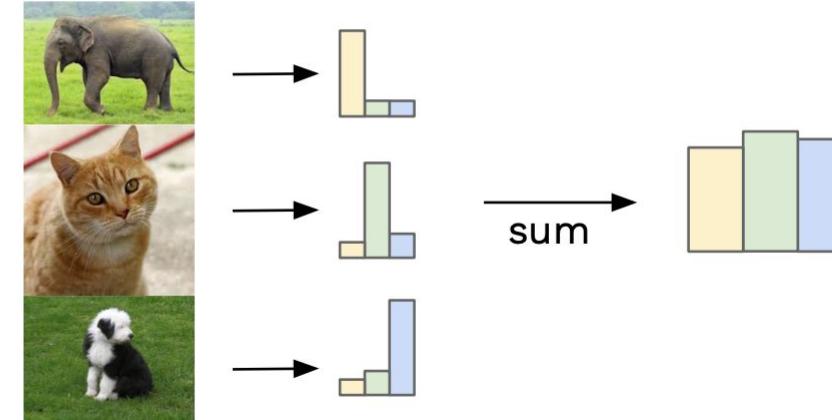
❖ Алгоритм:

- Генерируем K изображений N классов.
 - Выборка сбалансирована: каждый класс - K/N изображений
- Хотим, чтобы было вот так:

Similar labels sum to give focussed distribution



Different labels sum to give uniform distribution

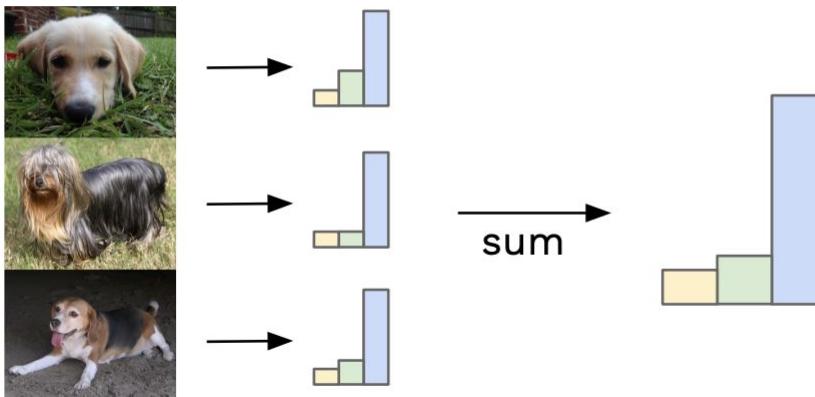


Метрики. Inception score

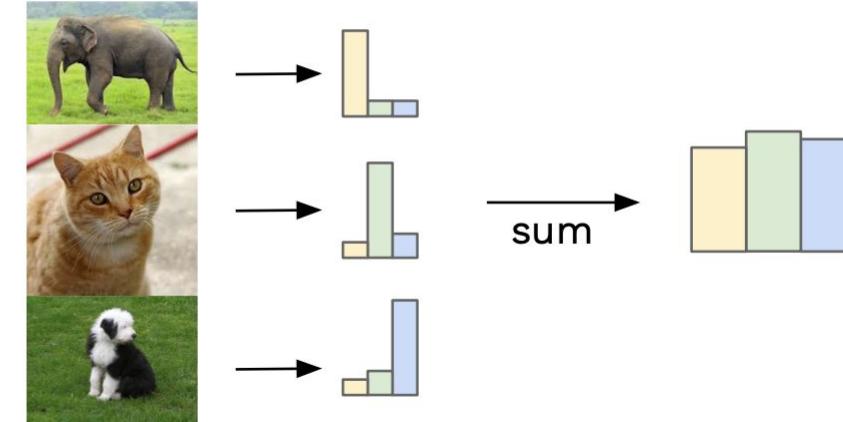
❖ Алгоритм:

- Генерируем K изображений N классов.
 - Выборка сбалансирована: каждый класс - K/N изображений
- Хотим, чтобы было вот так:

Similar labels sum to give focussed distribution



Different labels sum to give uniform distribution



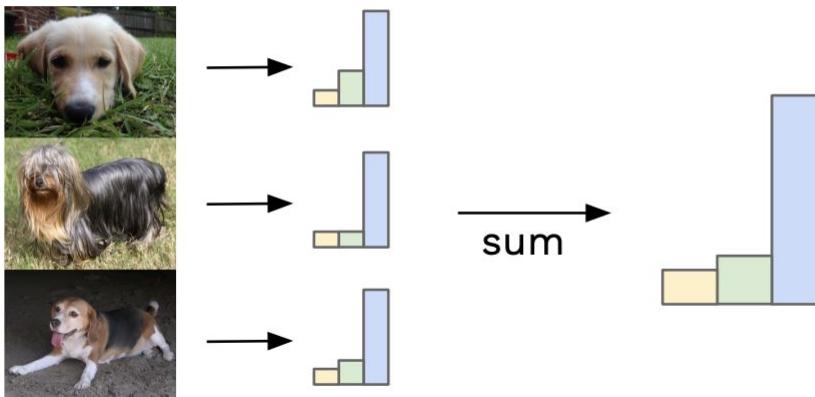
- Замеряем KL-дивергенцию между распределениями

Метрики. Inception score

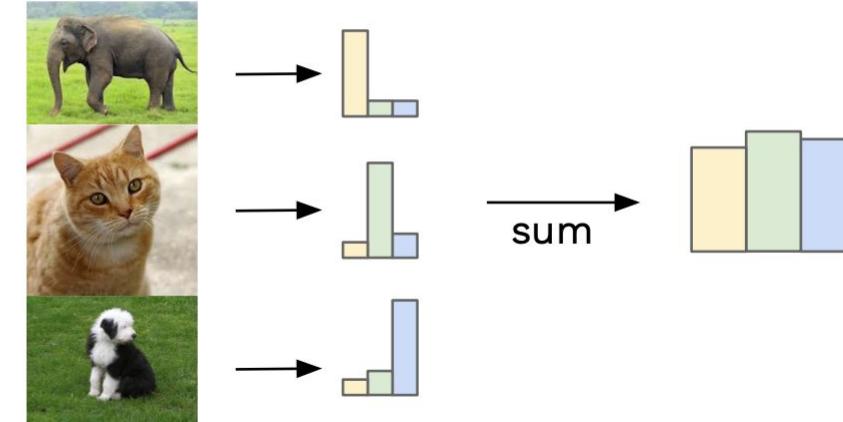
❖ Алгоритм:

- Генерируем K изображений N классов.
 - Выборка сбалансирована: каждый класс - K/N изображений
- Хотим, чтобы было вот так:

Similar labels sum to give focussed distribution



Different labels sum to give uniform distribution



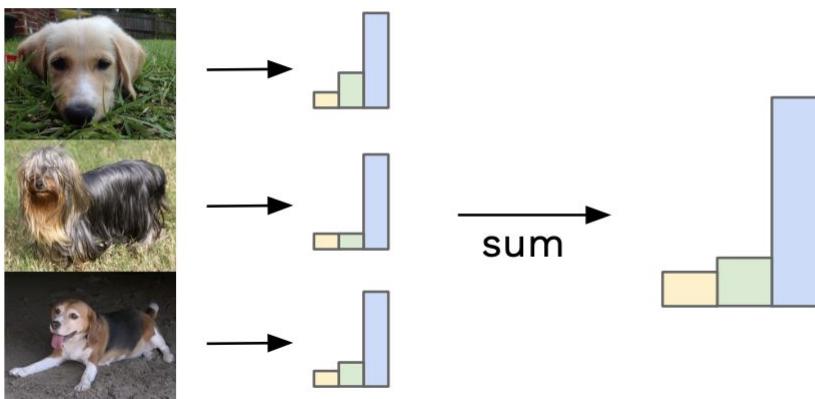
- Замеряем KL-дивергенцию между распределениями
- Берём экспоненту

Метрики. Inception score

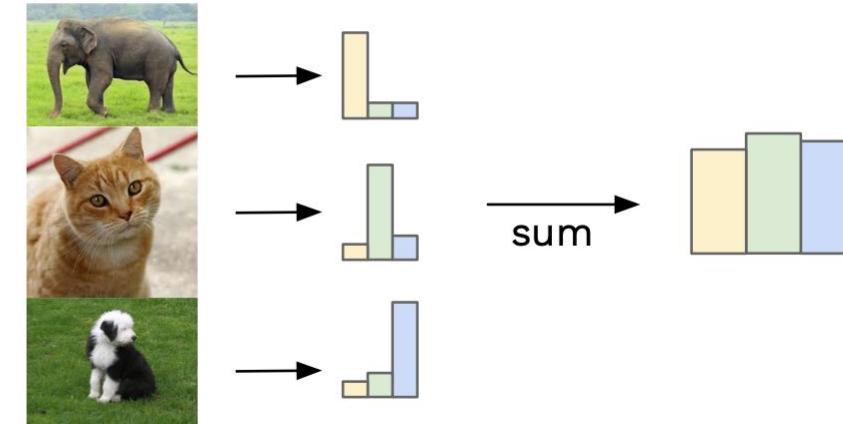
❖ Алгоритм:

- Генерируем K изображений N классов.
 - Выборка сбалансирована: каждый класс - K/N изображений
- Хотим, чтобы было вот так:

Similar labels sum to give focussed distribution



Different labels sum to give uniform distribution



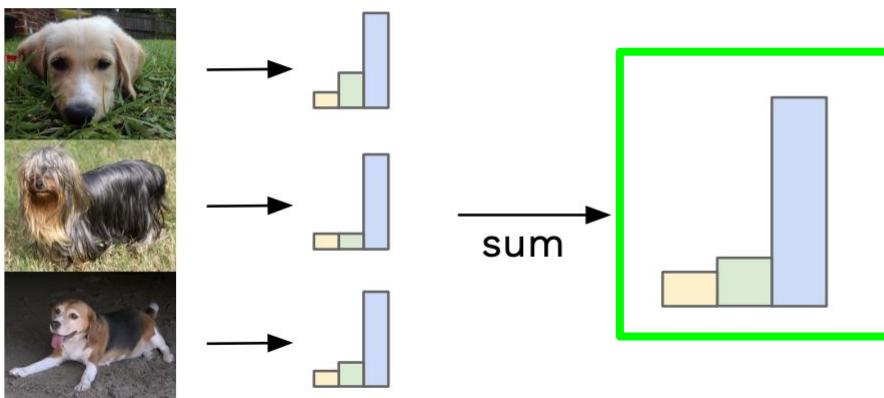
- Замеряем KL-дивергенцию между распределениями
- Берём экспоненту
- $\text{IS} = \exp(\mathbb{E}_x \text{KL}(p(y|x) || p(y)))$

Метрики. Inception score

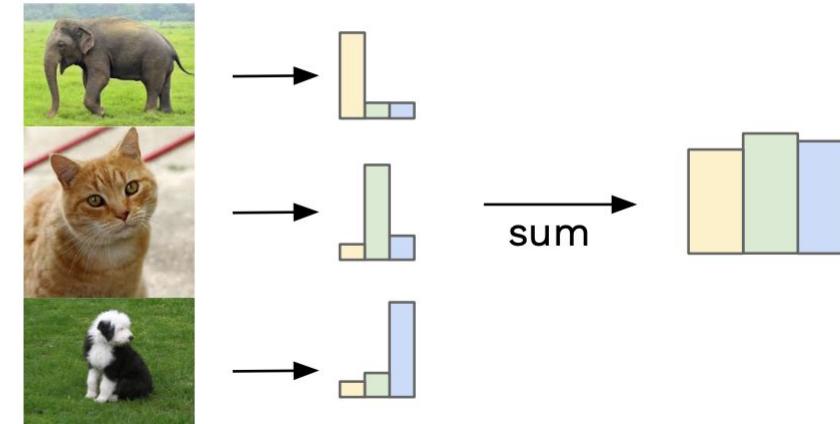
❖ Алгоритм:

- Генерируем K изображений N классов.
 - Выборка сбалансирована: каждый класс - K/N изображений
- Хотим, чтобы было вот так:

Similar labels sum to give focussed distribution



Different labels sum to give uniform distribution



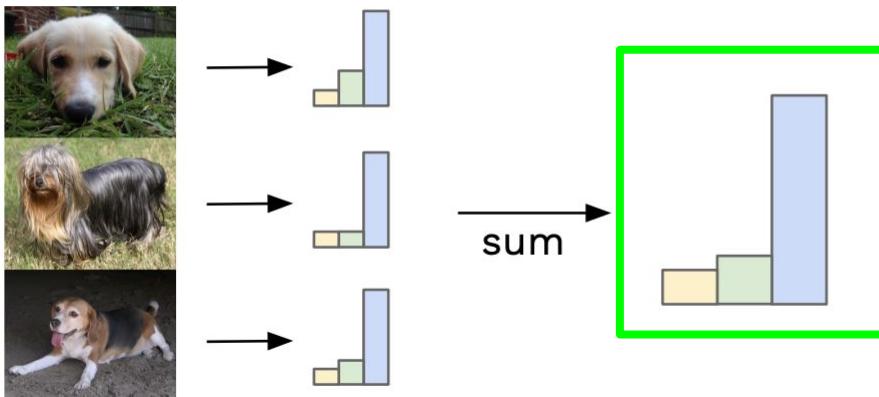
- Замеряем KL-дивергенцию между распределениями
- Берём экспоненту
- $\text{IS} = \exp(\mathbb{E}_x \text{KL}(p(y|x) | p(y)))$

Метрики. Inception score

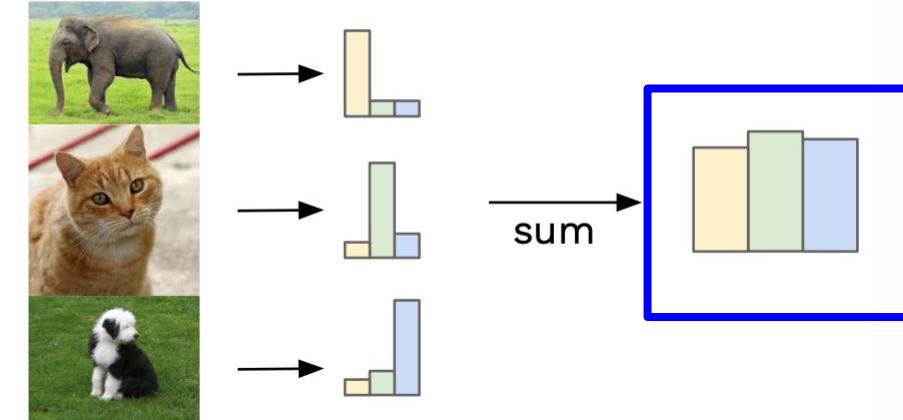
❖ Алгоритм:

- Генерируем K изображений N классов.
 - Выборка сбалансирована: каждый класс - K/N изображений
- Хотим, чтобы было вот так:

Similar labels sum to give focussed distribution



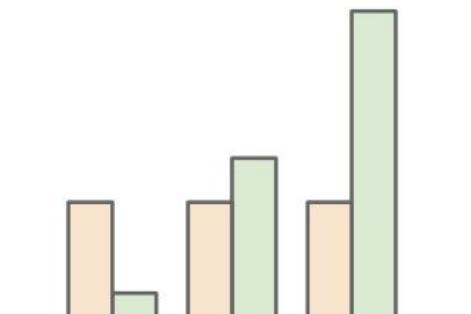
Different labels sum to give uniform distribution



- Замеряем KL-дивергенцию между распределениями
- Берём экспоненту
- $\text{IS} = \exp(\mathbb{E}_x \text{KL}(p(y|x) || p(y)))$

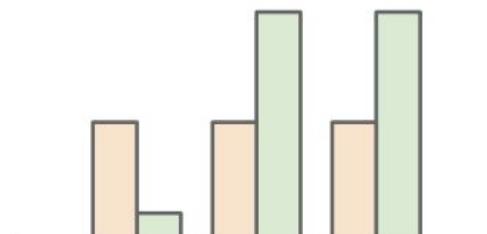
Метрики. Inception score

High KL divergence



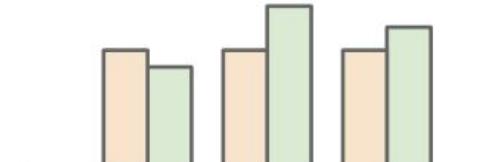
Ideal situation

Medium KL divergence



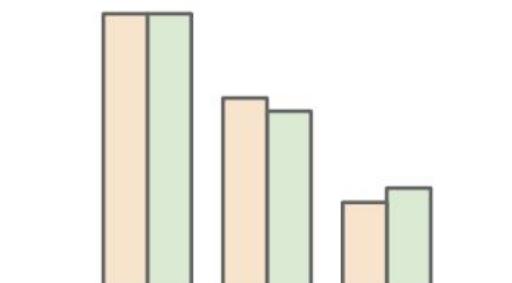
Generated images are
not distinctly one
label

Low KL divergence



Generated images are
not distinctly one
label

Low KL divergence



Generator lacks
diversity

Label distribution

Marginal distribution

$$\text{IS} = \exp(\mathbb{E}_x \text{KL}(p(y|x) || p(y)))$$

Inception score. Минусы

- ❖ Невозможно замерять генерацию объектов, не представленных в обучающей выборке
- ❖ Ограничение на один объект на фотографии
- ❖ Мы не учитываем датасет, на которым мы учились
- ❖ Можно сломать:



Figure 1. Sample of generated images achieving an Inception Score of 900.15. The maximum achievable Inception Score is 1000, and the highest achieved in the literature is on the order of 10.



Frechet Inception Distance (FID)

- ❖ Идея: сравнивать средние значения и разброс активаций предобученной сети на реальных и сгенерированных данных



Frechet Inception Distance (FID)

- ❖ Идея: сравнивать средние значения и разброс активаций предобученной сети на реальных и сгенерированных данных

$$FID = \|\mu_r - \mu_g\|^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}), \text{ где}$$

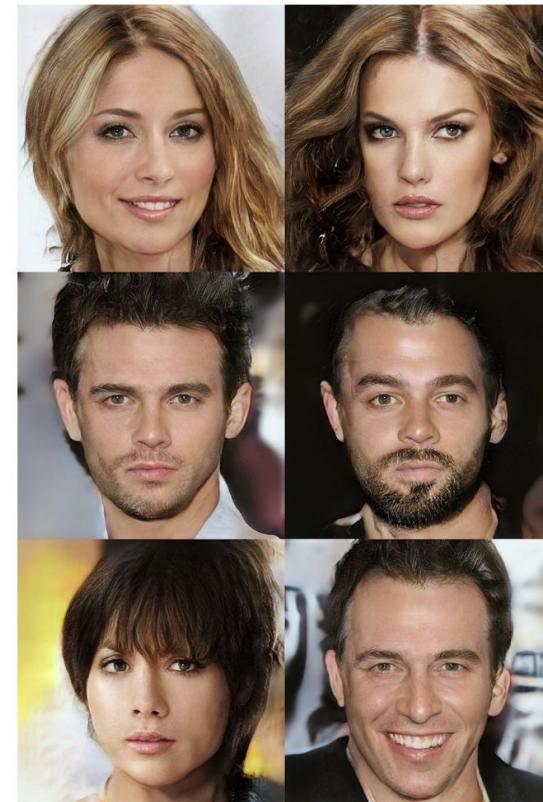
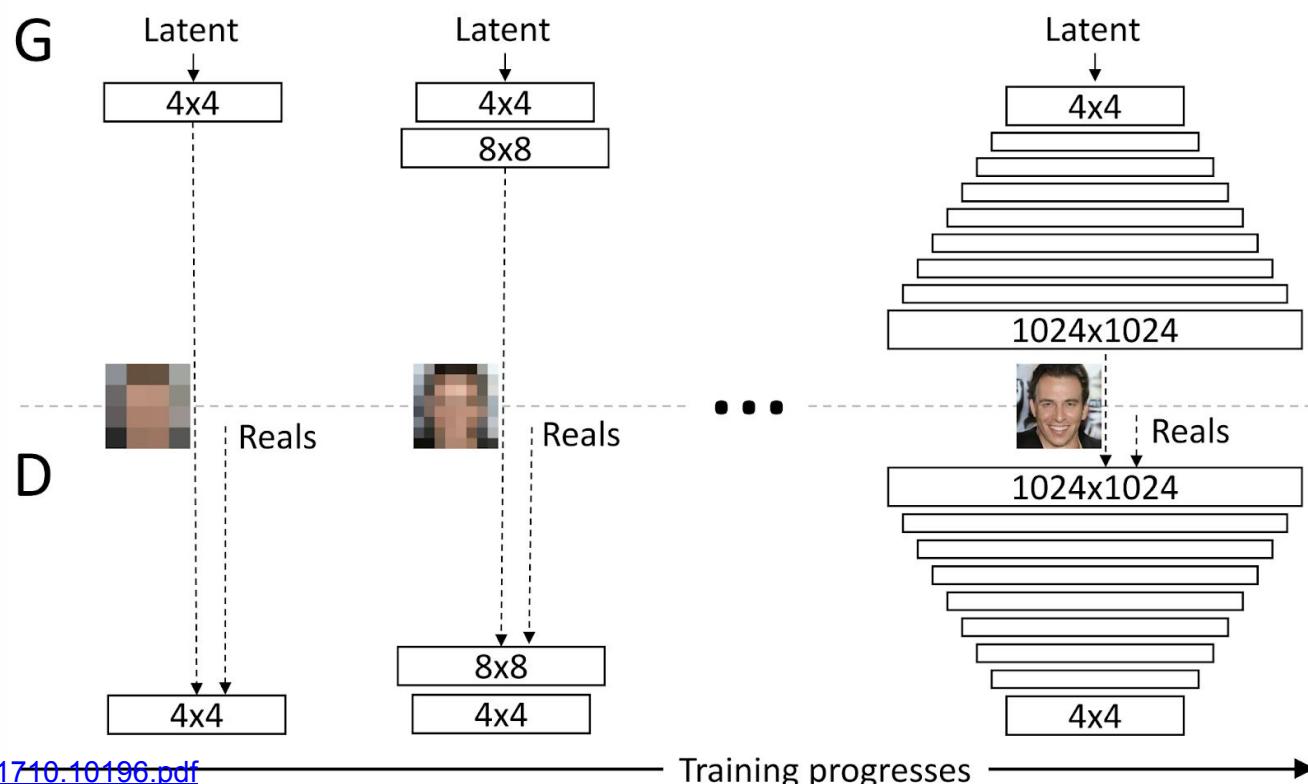
$X_r \sim \mathcal{N}(\mu_r, \Sigma_r)$ и $X_g \sim \mathcal{N}(\mu_g, \Sigma_g)$ - вектора длины 2048, полученные из pool3-слоя сети Inception V3

Progressive Growing of GANs



Progressive Growing of GANs

- ❖ Идея: учить генератор и дискриминатор, постепенно увеличивая разрешение изображений





Progressive Growing of GANs

- ❖ Идея: учить генератор и дискриминатор, постепенно увеличивая разрешение изображений



Progressive Growing of GANs

- ❖ Идея: учить генератор и дискриминатор, постепенно увеличивая разрешение изображений
- ❖ Функционал потерь: W-1 distance with Gradient Penalty

Discriminator/Critic

$$\nabla_w \frac{1}{m} \sum_{i=1}^m [f(x^{(i)}) - f(G(z^{(i)}))]$$

Generator

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^m f(G(z^{(i)}))$$

Progressive Growing of GANs

- ❖ Идея: учить генератор и дискриминатор, постепенно увеличивая разрешение изображений
- ❖ Функционал потерь: W-1 distance with Gradient Penalty

Discriminator/Critic

$$\nabla_w \frac{1}{m} \sum_{i=1}^m [f(x^{(i)}) - f(G(z^{(i)}))]$$

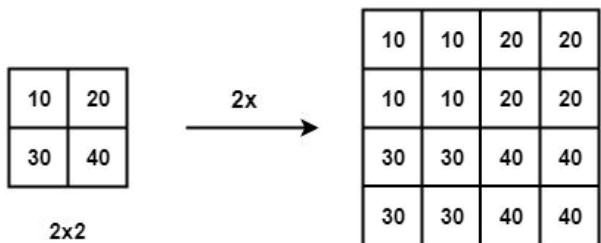
Generator

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^m f(G(z^{(i)}))$$

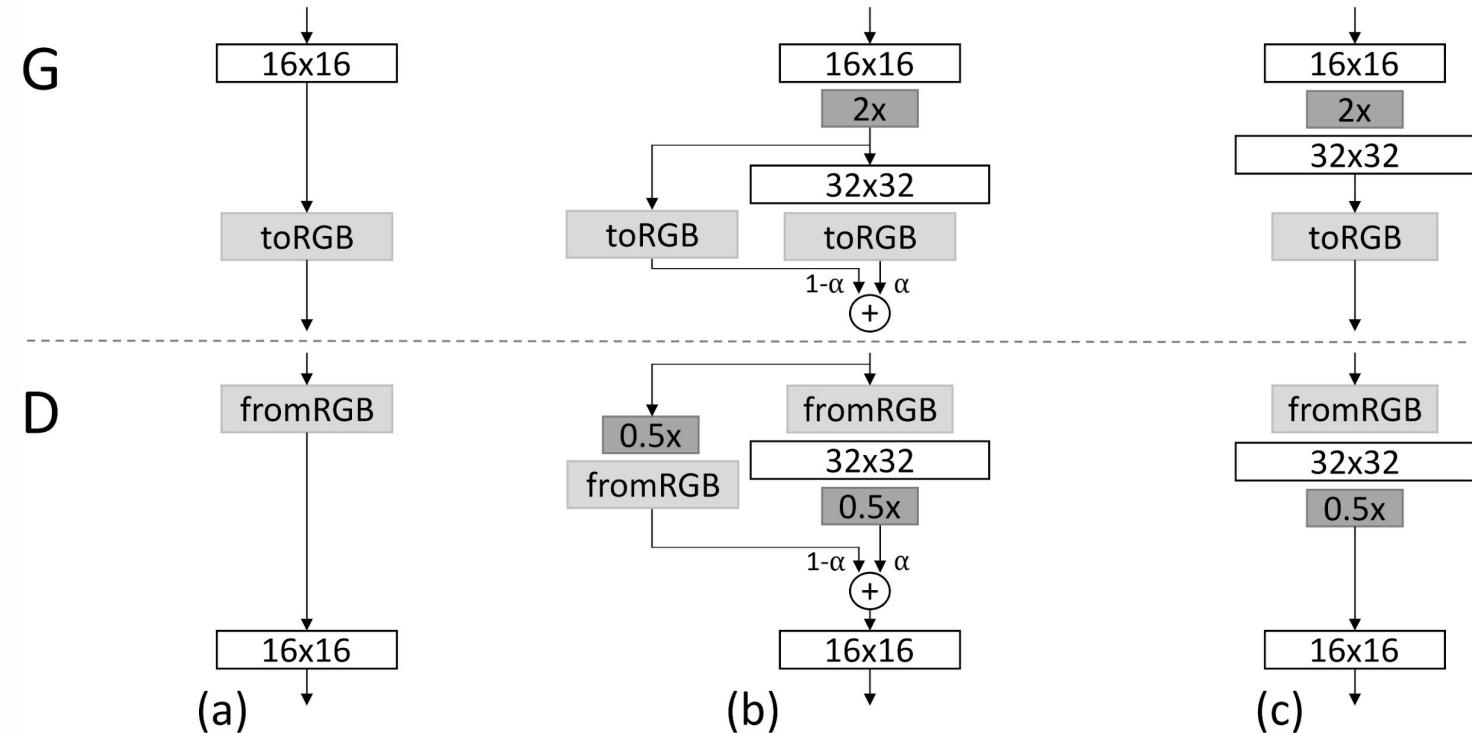
- ❖ Gradient Penalty - в лосс добавляется отличие нормы градиента от единицы

Progressive Growing of GANs

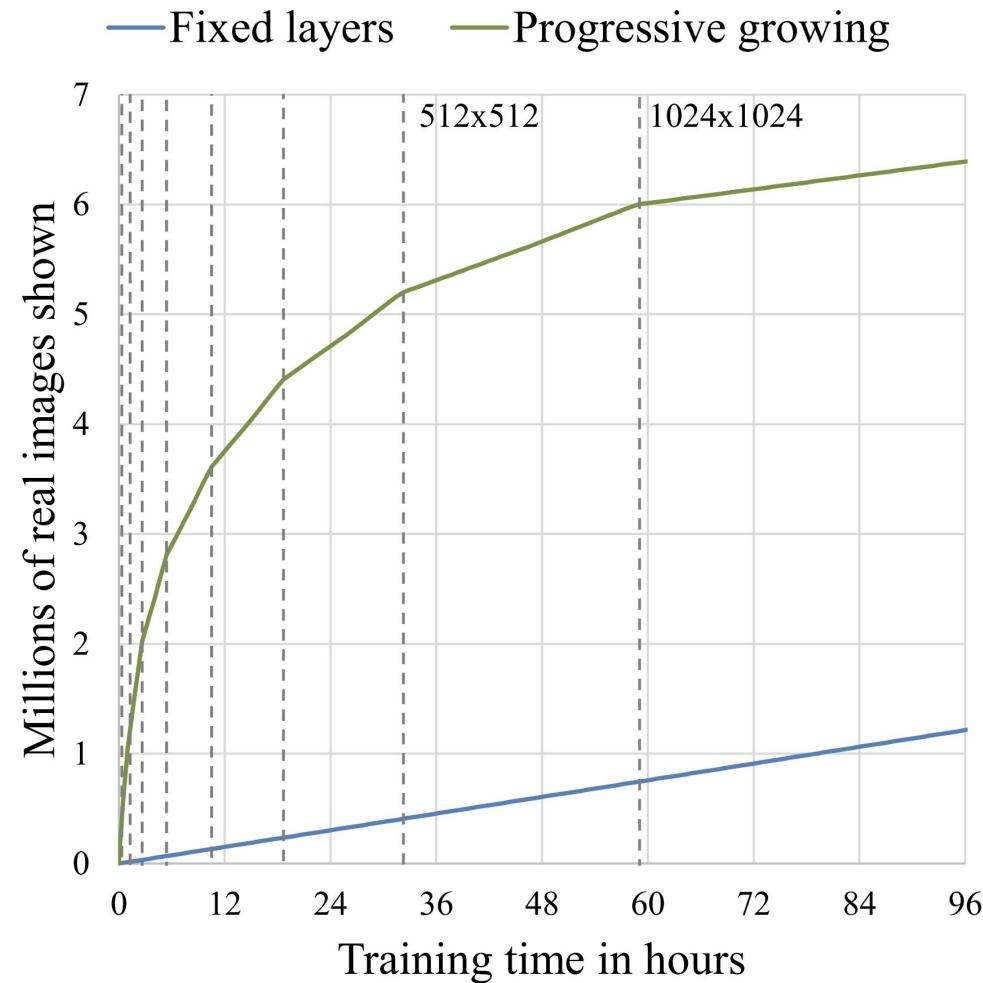
toRGB: свёртка 1x1
2x: nearest neighbour interpolation



α меняется линейно от 0 до 1 во время обучения



Progressive Growing of GANs



Progressive Growing of GANs. Результаты



Progressive Growing of GANs. Результаты



Mao et al. (2016b) (128×128)

Gulrajani et al. (2017) (128×128)

Our (256×256)

Figure 6: Visual quality comparison in LSUN BEDROOM; pictures copied from the cited articles.

BigGAN

- ❖ Размер изображений: 128x128, 256x256 и 512x512
- ❖ Не использует идею progressive growing
- ❖ Получили хорошие результаты

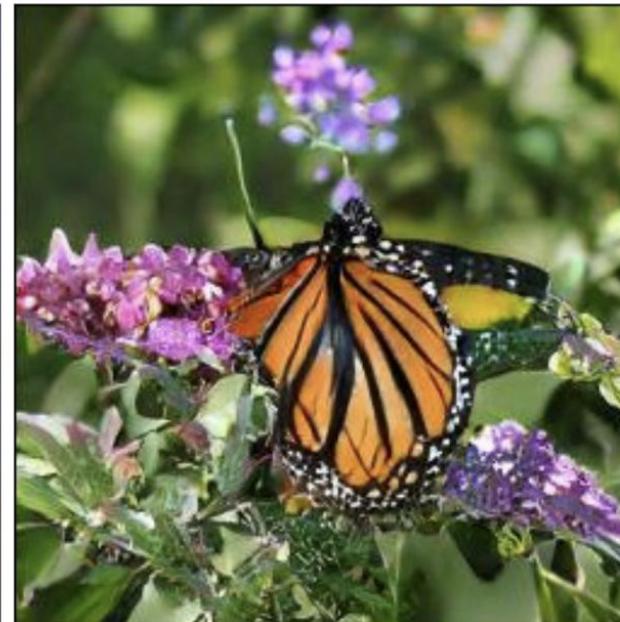
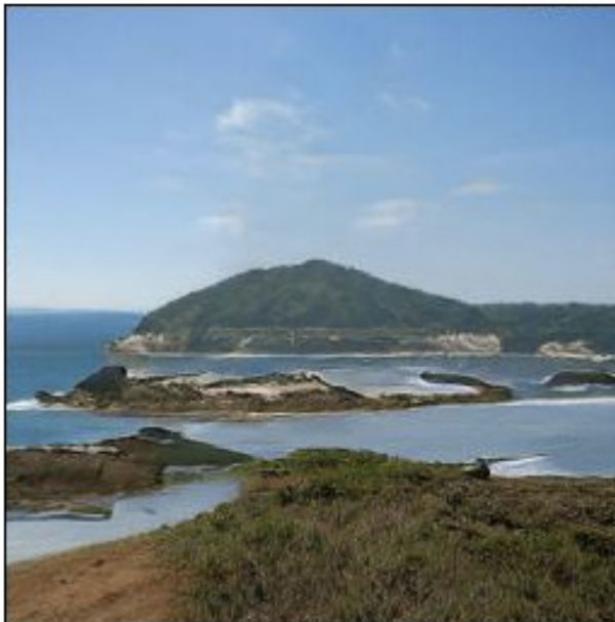


Figure 1: Class-conditional samples generated by our model.



BigGAN. Трюки

- ❖ Self-Attention
- ❖ Hinge Loss
- ❖ Использование метки класса в дискриминаторе
- ❖ Спектральная нормализация
- ❖ Чаще обновлять дискриминатор
- ❖ Экспоненциальное усреднение весов генератора
- ❖ Ортогональная инициализация весов
- ❖ Большоооооой размер батча
- ❖ Больше параметров сети
- ❖ Skip-Z connections
- ❖ Truncation trick
- ❖ Ортогональная регуляризация



BigGAN. Трюки

- ❖ **Self-Attention**
- ❖ **Hinge Loss**
- ❖ Использование метки класса в дискриминаторе
- ❖ **Спектральная нормализация**
- ❖ Чаще обновлять дискриминатор
- ❖ Экспоненциальное усреднение весов генератора
- ❖ Ортогональная инициализация весов
- ❖ **Большооооой размер батча**
- ❖ Больше параметров сети
- ❖ Skip-Z connections
- ❖ **Truncation trick**
- ❖ **Ортогональная регуляризация**

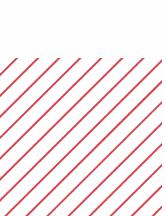


Hinge Loss Function

Было:

$$D: \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

$$G: \max_G \mathbb{E}_{z \sim p_z(z)} \log(D(G(z)))$$



Hinge Loss Function

Было:

$$D: \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

$$G: \max_G \mathbb{E}_{z \sim p_z(z)} \log(D(G(z)))$$

Стало:

$$L_D = -\mathbb{E}_{(x,y) \sim p_{\text{data}}} [\min(0, -1 + D(x, y))] - \mathbb{E}_{z \sim p_z, y \sim p_{\text{data}}} [\min(0, -1 - D(G(z), y))],$$

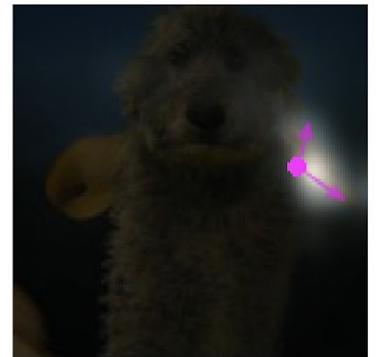
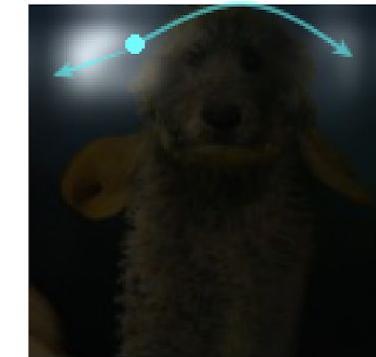
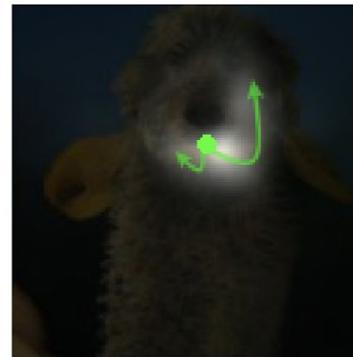
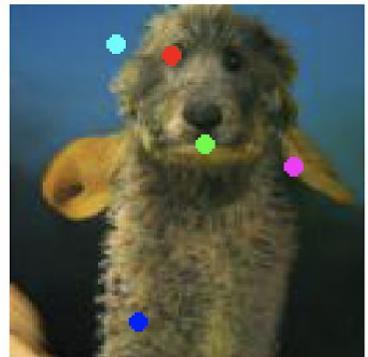
$$L_G = -\mathbb{E}_{z \sim p_z, y \sim p_{\text{data}}} D(G(z), y),$$



Self-attention

- ❖ Идея:
 - приучить сеть использовать информацию о разных частях изображения
- ❖ Пример:
 - у собаки должно быть 2 глаза, 4 лапы и т.д.

Self-attention

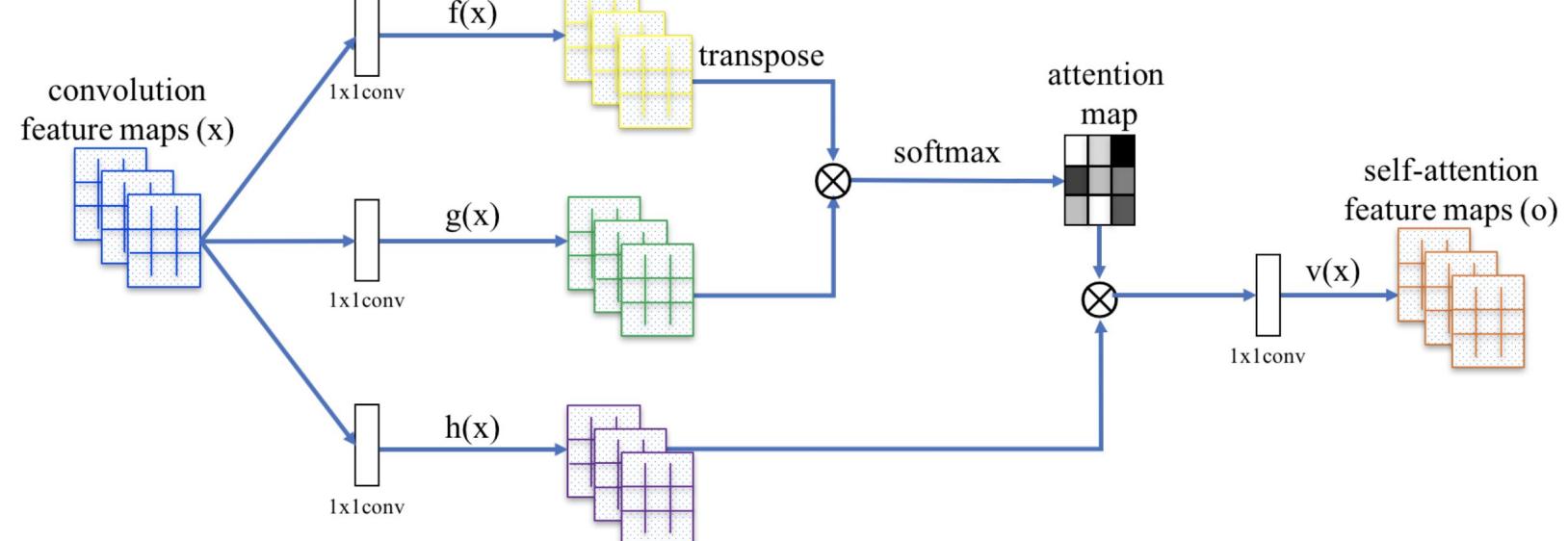




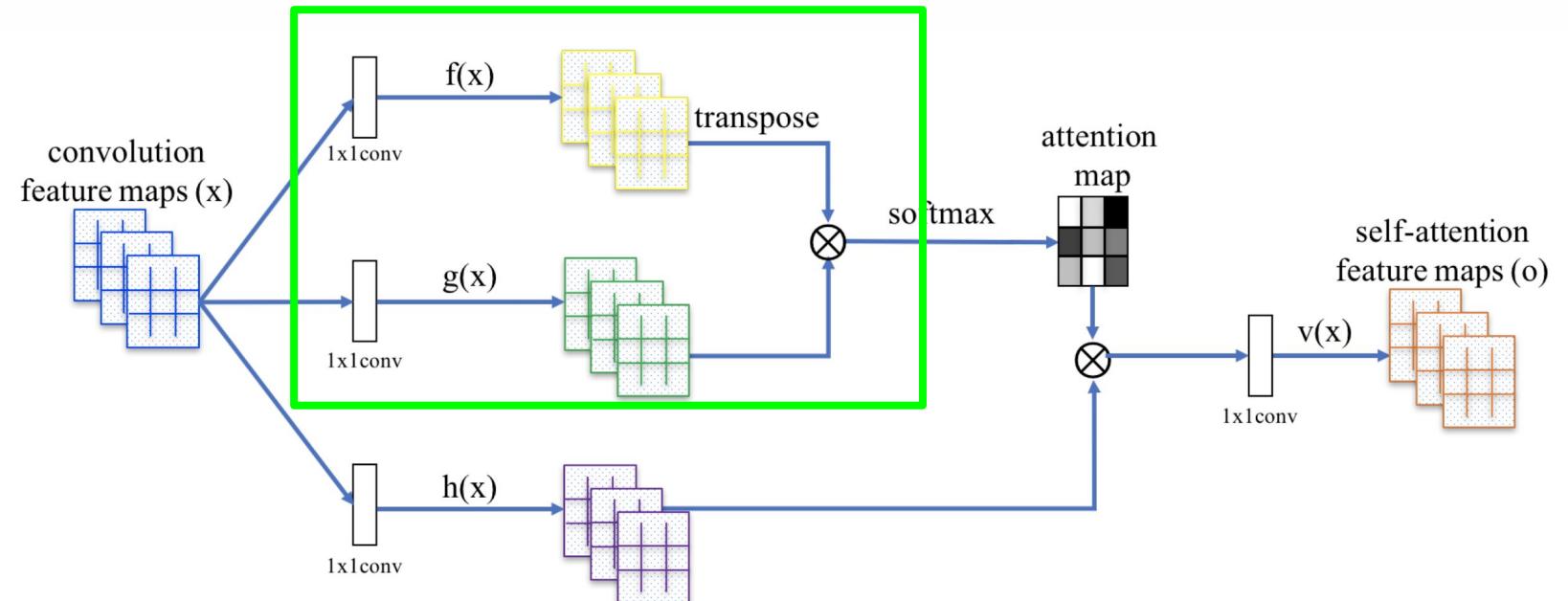
Self-attention



Self-attention



Self-attention



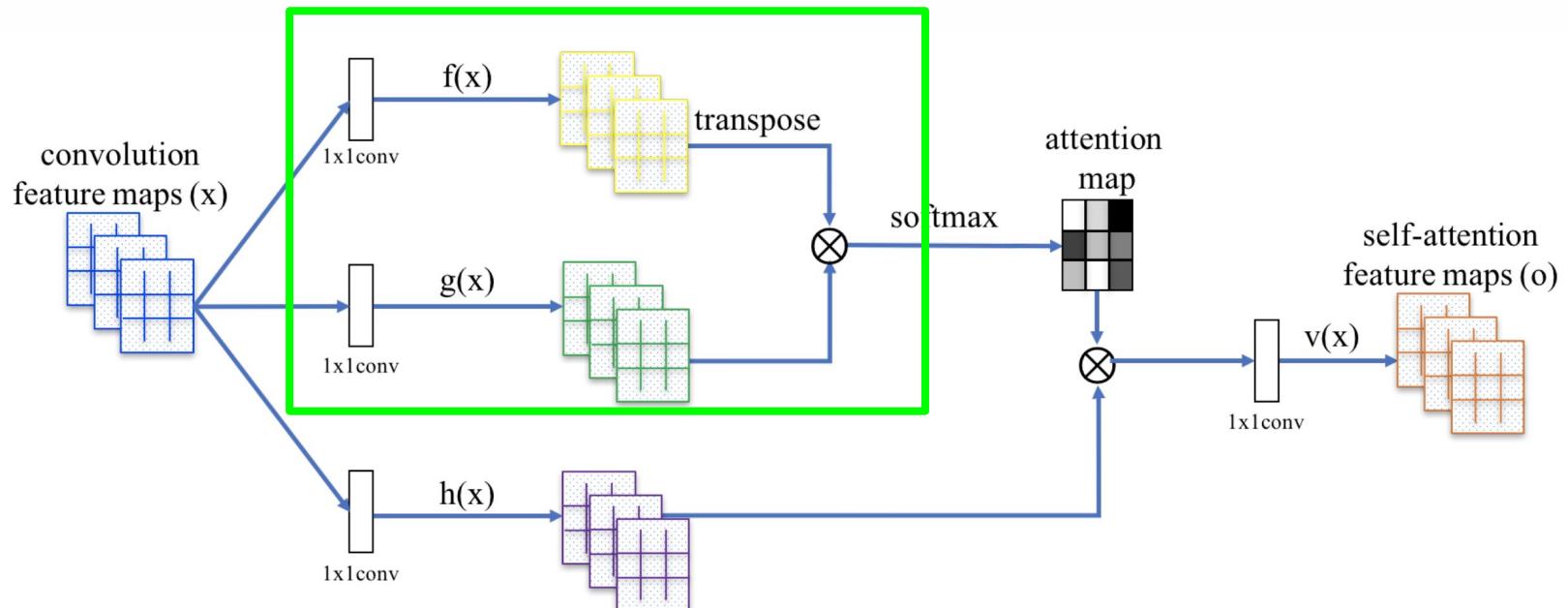
Self-attention

$$g(x) = W_g x$$

$$f(x) = W_f x$$

$$s_{ij} = f(x_i)^T g(x_j)$$

$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^N \exp(s_{ij})}$$



Self-attention

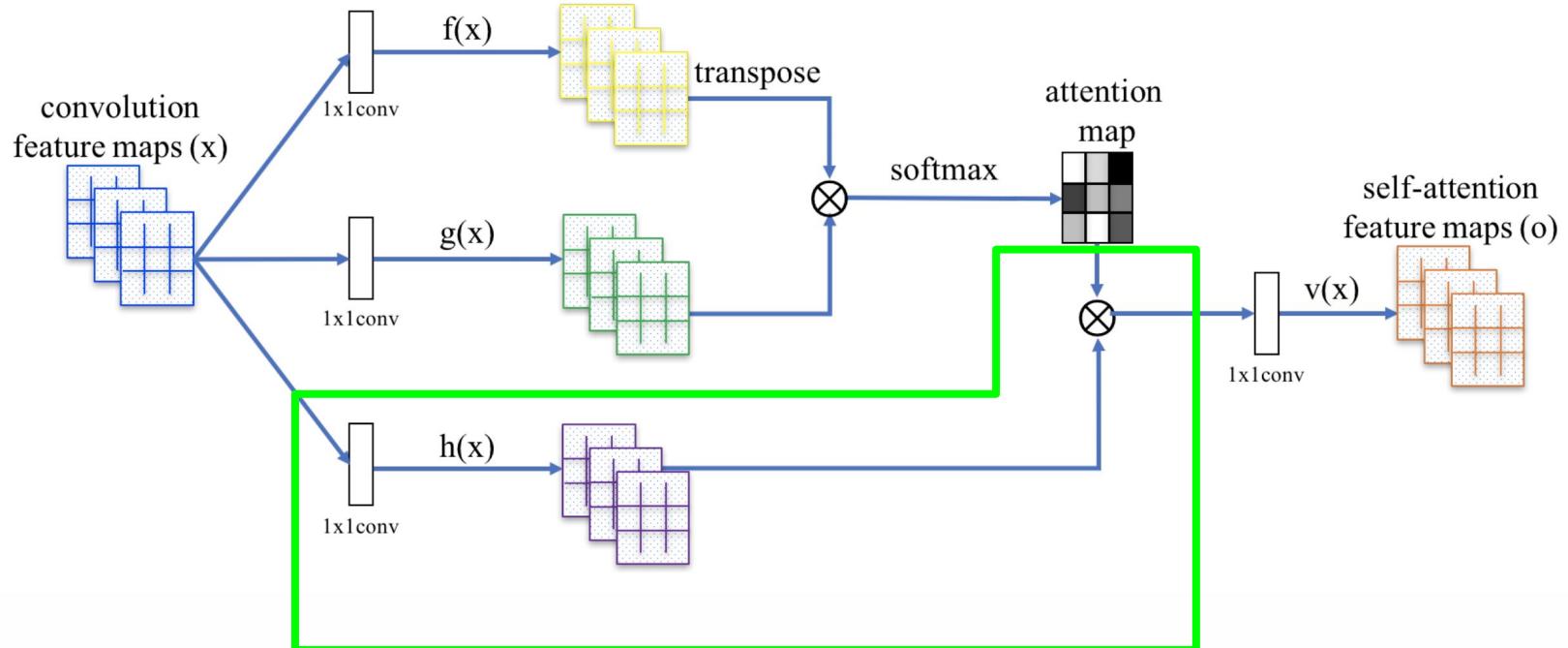
$$\mathbf{h}(\mathbf{x}_i) = \mathbf{W}_h \mathbf{x}_i$$

$$\mathbf{o}_j = \sum_{i=1}^N \beta_{j,i} \mathbf{h}(\mathbf{x}_i)$$

$$\mathbf{W}_h \in \mathbb{R}^{C \times C}$$

$$\mathbf{o} \in \mathbb{R}^{C \times N}$$

$$\mathbf{y}_i = \gamma \mathbf{o}_i + \mathbf{x}_i$$



Self-attention

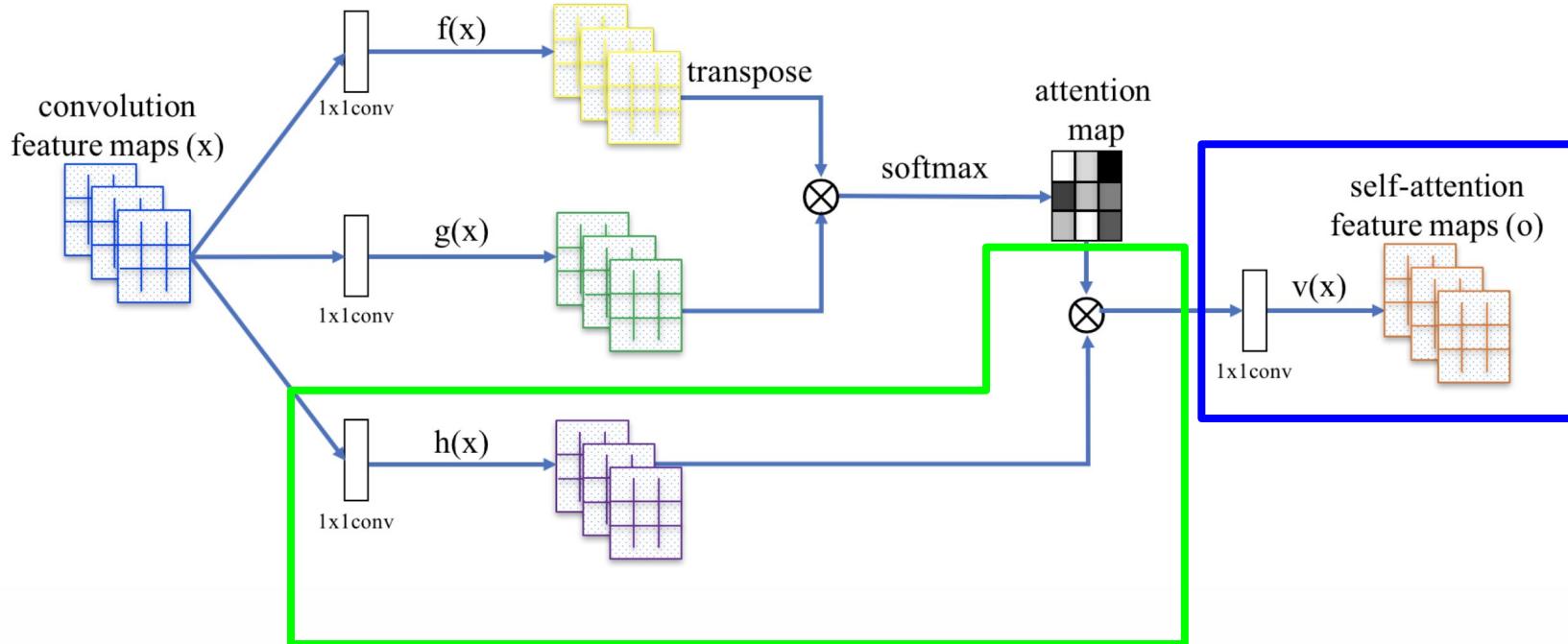
$$\mathbf{h}(\mathbf{x}_i) = \mathbf{W}_h \mathbf{x}_i$$

$$\mathbf{o}_j = \sum_{i=1}^N \beta_{j,i} \mathbf{h}(\mathbf{x}_i)$$

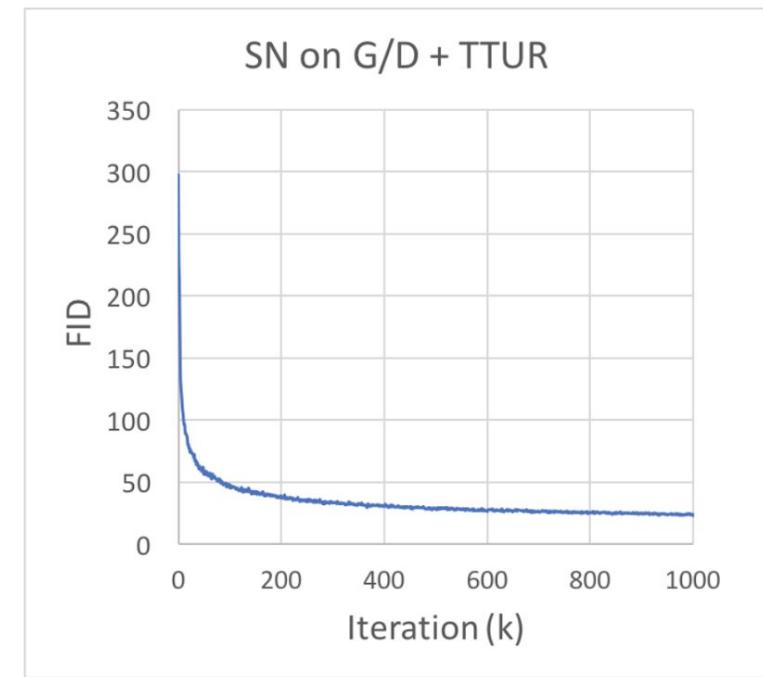
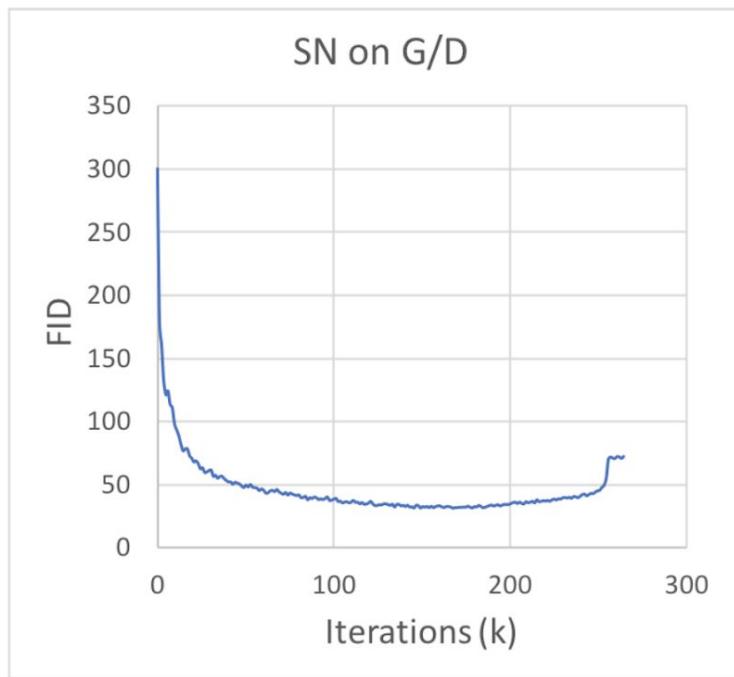
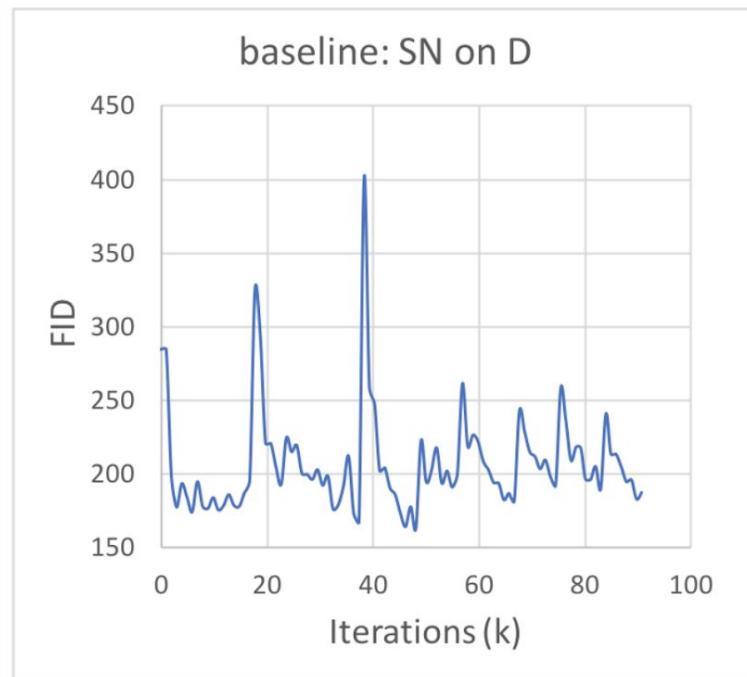
$$\mathbf{W}_h \in \mathbb{R}^{C \times C}$$

$$\mathbf{o} \in \mathbb{R}^{C \times N}$$

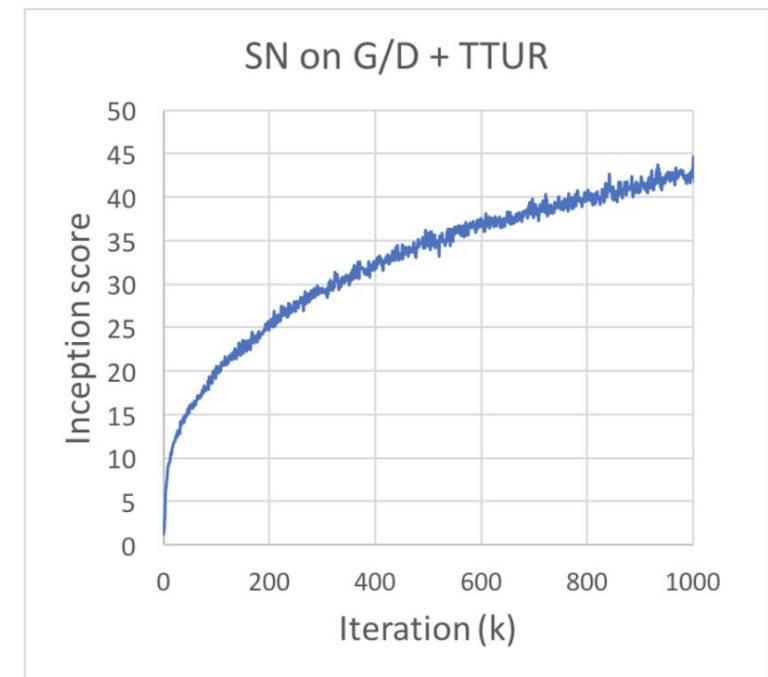
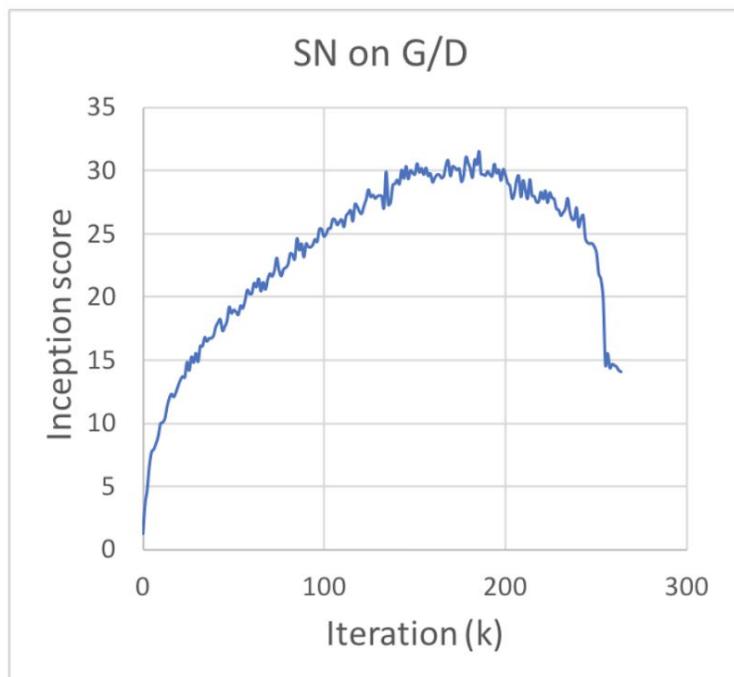
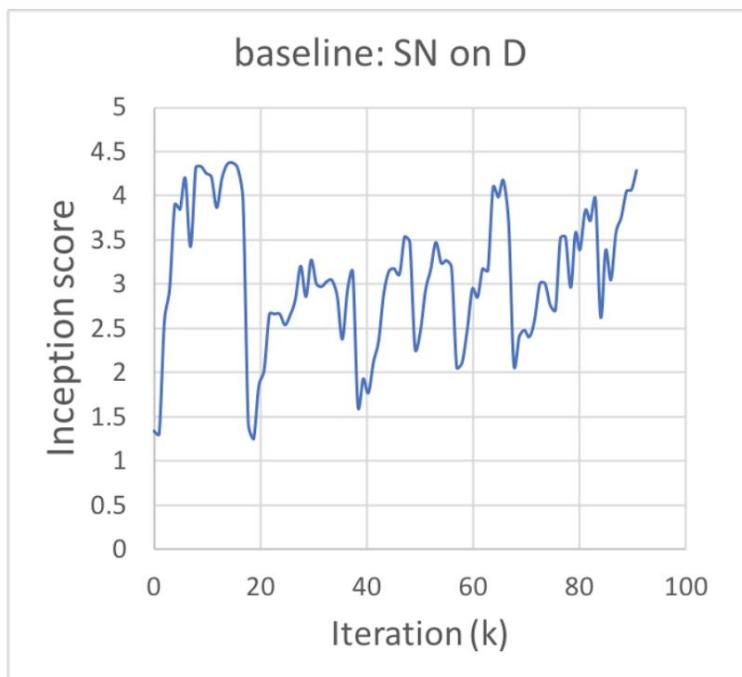
$$\mathbf{y}_i = \gamma \mathbf{o}_i + \mathbf{x}_i$$



Self-attention. Frechet Inception Distance



Self-attention. Inception score





Спектральная нормализация

- ❖ Идея:
 - добиться непрерывности по Липшицу каждого слоя нейросети, поделив каждый вес на его спектральную норму
 - Спектральная норма - максимальное сингулярное число матрицы весов



Размер батча

- ❖ Размер батча при обучении имеет большое значение

Размер батча

- ❖ Размер батча при обучении имеет большое значение
- ❖ Увеличение батча в 8 раз дало улучшение качества на 46%

Batch	Ch.	Param (M)	Shared	Skip- z	Ortho.	Itr $\times 10^3$	FID	IS
256	64	81.5		SA-GAN Baseline		1000	18.65	52.52
512	64	81.5	\times	\times	\times	1000	15.30	58.77(± 1.18)
1024	64	81.5	\times	\times	\times	1000	14.88	63.03(± 1.42)
2048	64	81.5	\times	\times	\times	732	12.39	76.85(± 3.83)

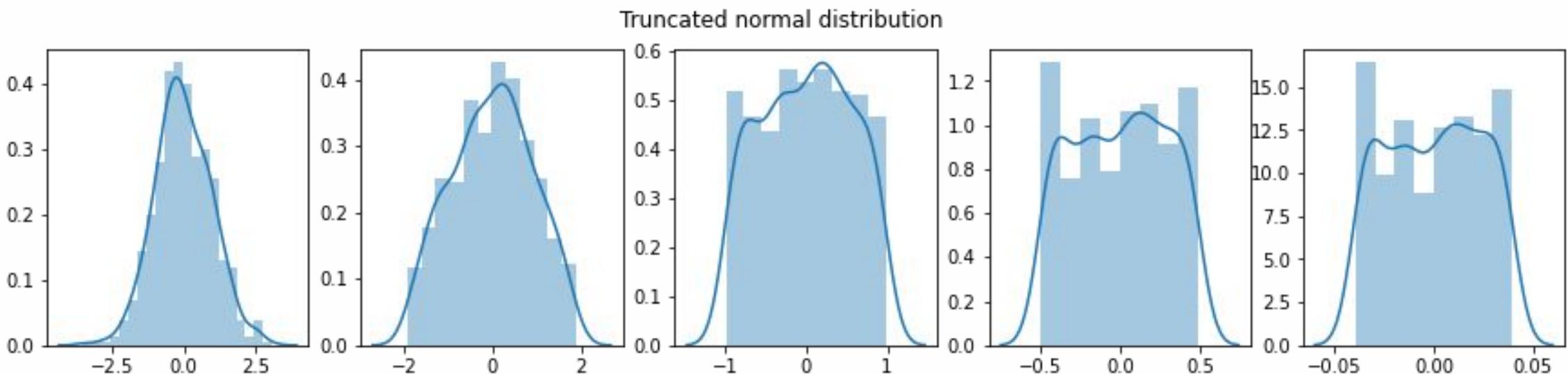


Truncation trick

- ❖ Обучение: входной шум - нормальное распределение $z \sim \mathcal{N}(0, I)$
- ❖ Инференс: входной шум - обрезанное (*trunkated*) нормальное распределение

Truncation trick

- ❖ Обучение: входной шум - нормальное распределение $z \sim \mathcal{N}(0, I)$
- ❖ Инференс: входной шум - обрезанное (*truncated*) нормальное распределение



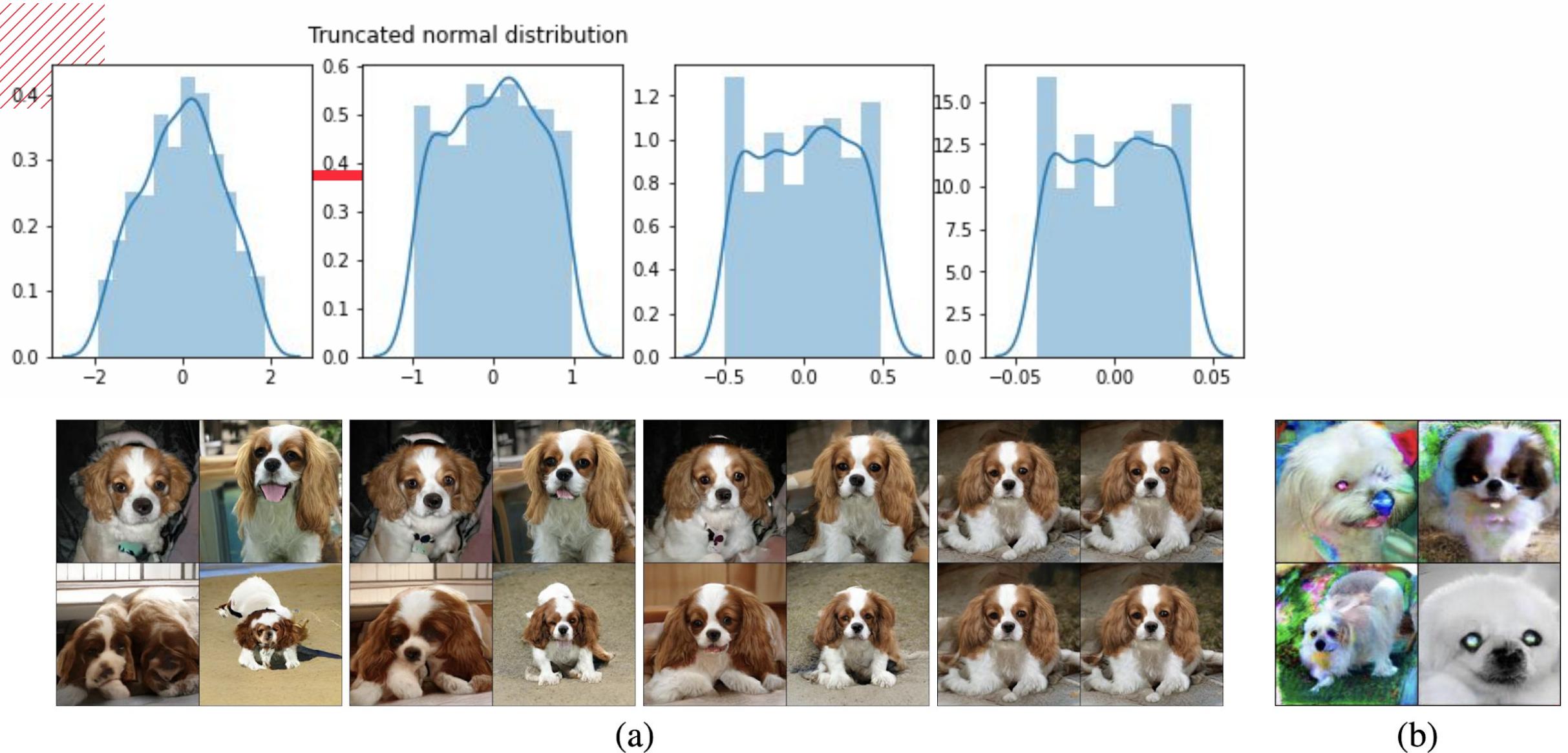


Figure 2: (a) The effects of increasing truncation. From left to right, the threshold is set to 2, 1, 0.5, 0.04. (b) Saturation artifacts from applying truncation to a poorly conditioned model.

Ортогональная регуляризация

- ❖ Truncation trick помогает не всем моделям
- ❖ Ортогональная матрица: $AA^T = A^T A = E$ или $A^{-1} = A^T$
- ❖ Идея:
 - поддерживать ортогональность весов, чтобы при умножении на них норма матрицы не изменялась

$$\mathcal{L}_{ortho} = \Sigma(|WW^T - I|)$$

Ортогональная регуляризация

- ❖ Идея:
 - поддерживать ортогональность весов, чтобы при умножении на них норма матрицы не изменялась

$$R_\beta(W) = \beta \|W^\top W - I\|_F^2$$

- Можно сделать требование более мягким:

$$R_\beta(W) = \beta \|W^\top W \odot (1 - I)\|_F^2$$



Ортогональная регуляризация

- ❖ Было: 16% моделей улучшились от truncation trick
- ❖ Стало 60%

Ортогональная регуляризация

- ❖ Было: 16% моделей улучшались от truncation trick
- ❖ Стало 60%

Batch	Ch.	Param (M)	Shared	Skip- z	Ortho.	Itr $\times 10^3$	FID	IS
256	64	81.5		SA-GAN Baseline			1000	18.65
512	64	81.5	✗	✗	✗	1000	15.30	58.77(± 1.18)
1024	64	81.5	✗	✗	✗	1000	14.88	63.03(± 1.42)
2048	64	81.5	✗	✗	✗	732	12.39	76.85(± 3.83)
2048	96	173.5	✗	✗	✗	295(± 18)	9.54(± 0.62)	92.98(± 4.27)
2048	96	160.6	✓	✗	✗	185(± 11)	9.18(± 0.13)	94.94(± 1.32)
2048	96	158.3	✓	✓	✗	152(± 7)	8.73(± 0.45)	98.76(± 2.84)
2048	96	158.3	✓	✓	✓	165(± 13)	8.51(± 0.32)	99.31(± 2.10)
2048	64	71.3	✓	✓	✓	371(± 7)	10.48(± 0.10)	86.90(± 0.61)

BigGAN





BigGAN

- ❖ Собрали и проверили большое количество хаков
- ❖ Многие используются до сих пор

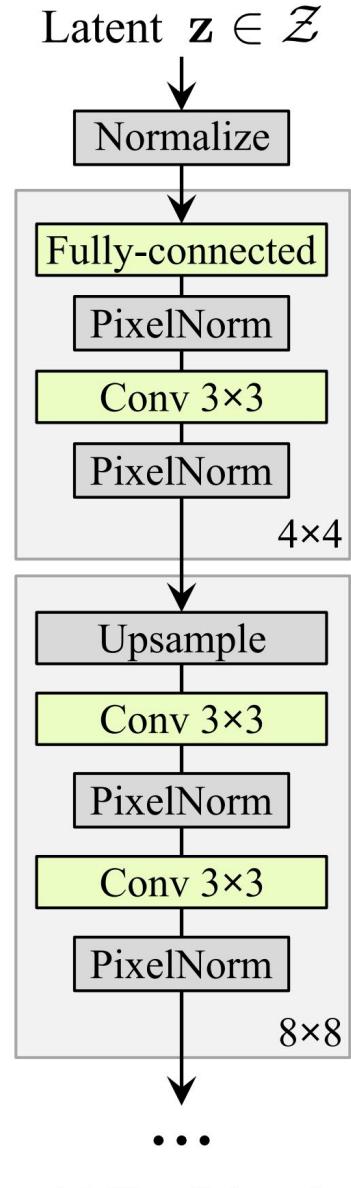
BigGAN

- ❖ Собрали и проверили большое количество хаков
- ❖ Многие используются до сих пор



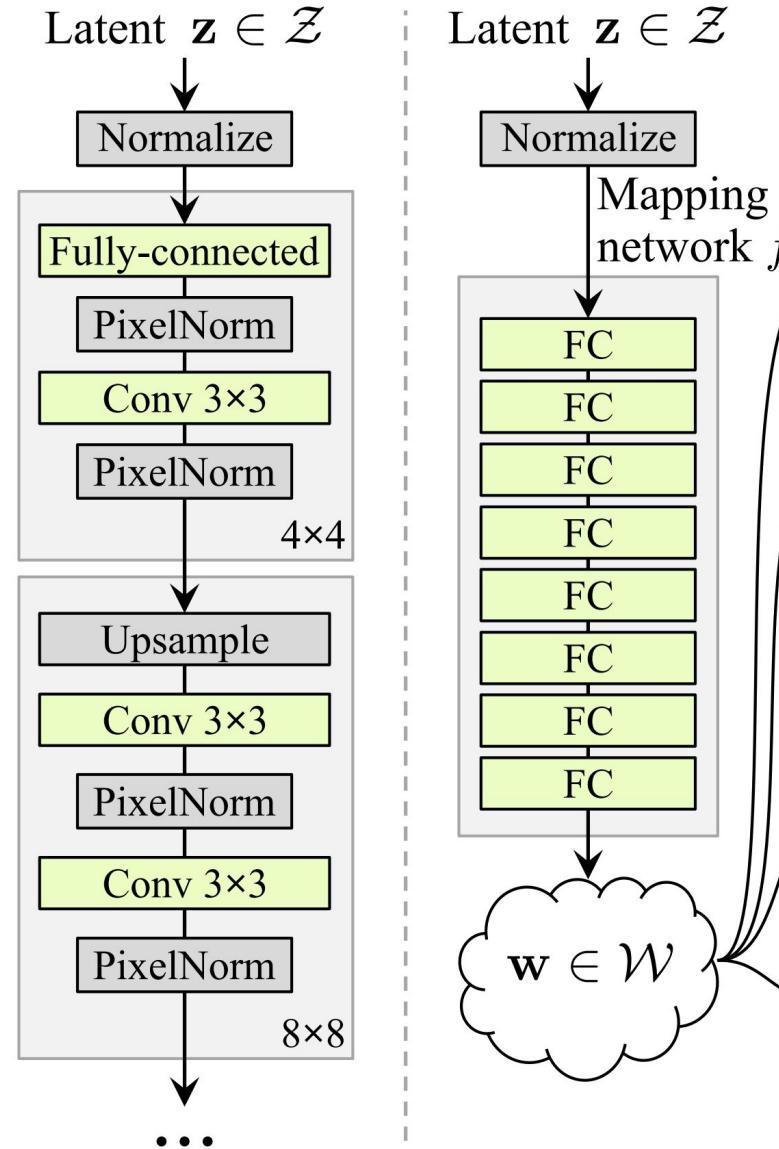
StyleGAN

- ❖ Было: вектор \mathbf{z} из нормального распределения



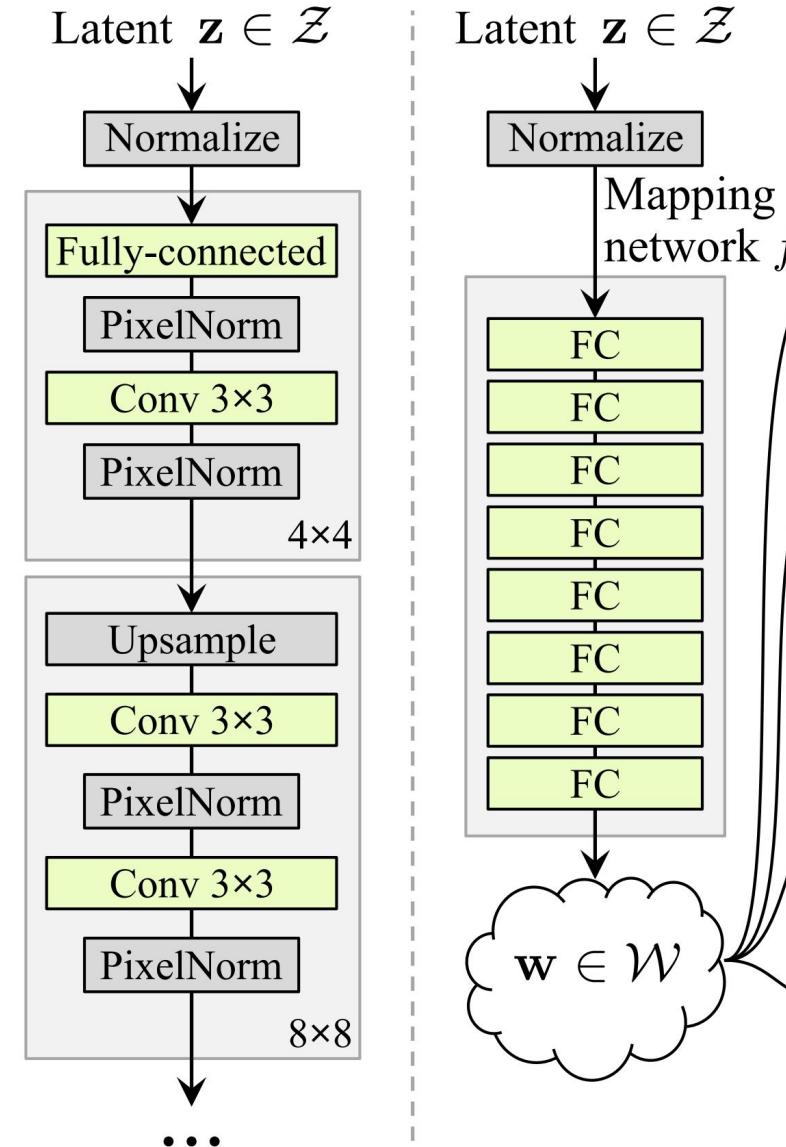
StyleGAN

- ❖ Было: вектор \mathbf{z} из нормального распределения
- ❖ Стало: вытягиваем фичи из \mathbf{z} в \mathbf{w}



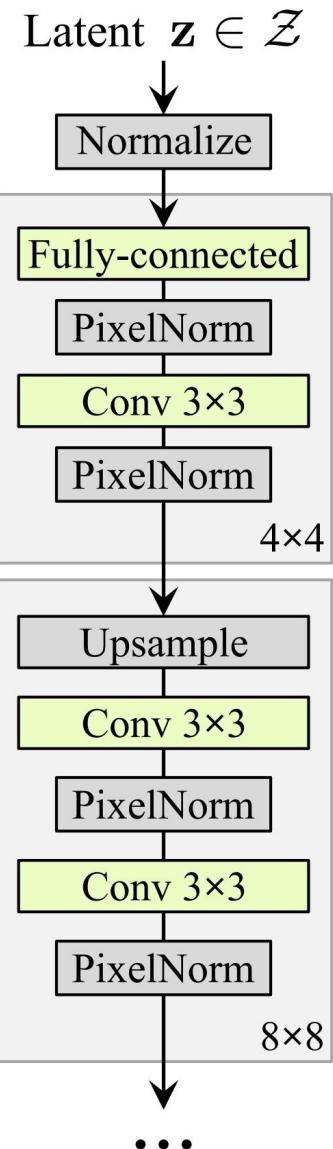
StyleGAN

- ❖ Было: вектор \mathbf{z} из нормального распределения
- ❖ Стало: вытягиваем фичи из \mathbf{z} в \mathbf{w}
- ❖ \mathbf{w} лучше описывает разнообразие

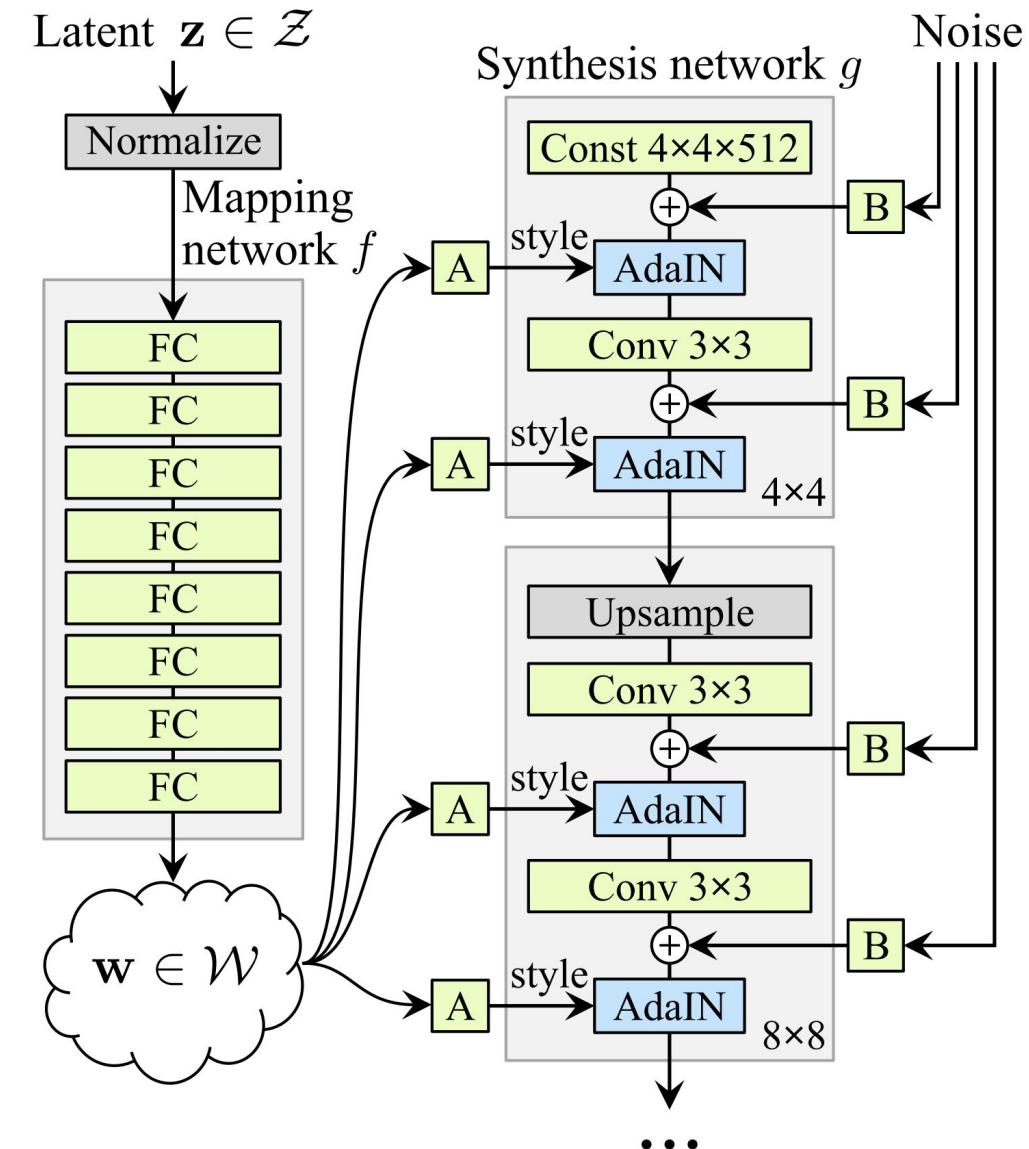


StyleGAN

- ❖ Было: вектор \mathbf{z} из нормального распределения
- ❖ Стало: вытягиваем фичи из \mathbf{z} в \mathbf{w}
- ❖ \mathbf{w} лучше описывает разнообразие



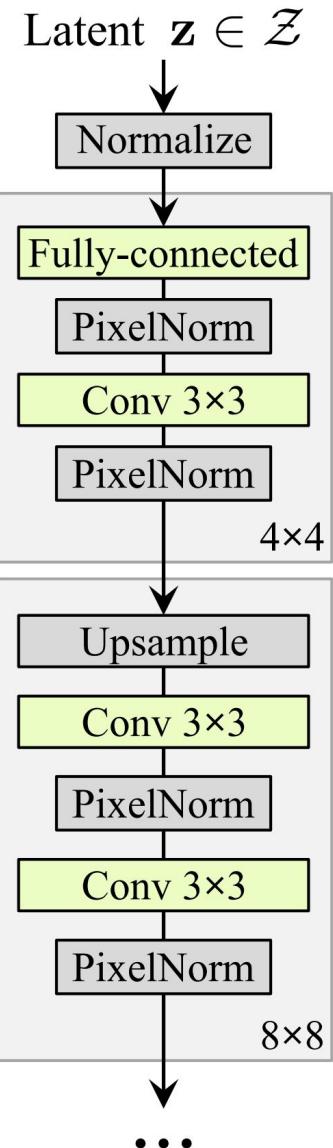
(a) Traditional



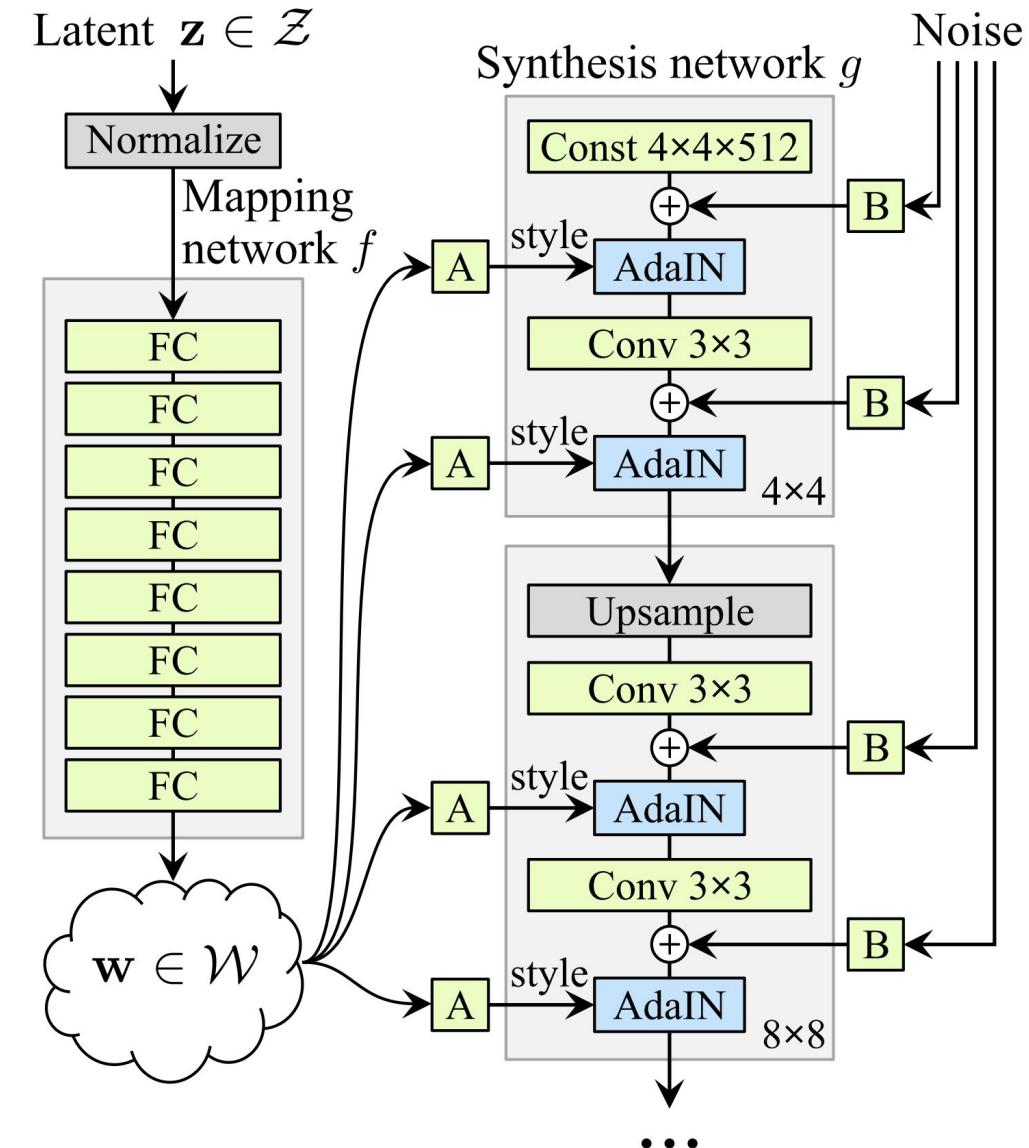
(b) Style-based generator

StyleGAN

- ❖ Было: вектор \mathbf{z} из нормального распределения
- ❖ Стало: вытягиваем фичи из \mathbf{z} в \mathbf{w}
- ❖ \mathbf{w} лучше описывает разнообразие
- ❖ AdaIN слои помогают учитывать \mathbf{w} во время генерации



(a) Traditional

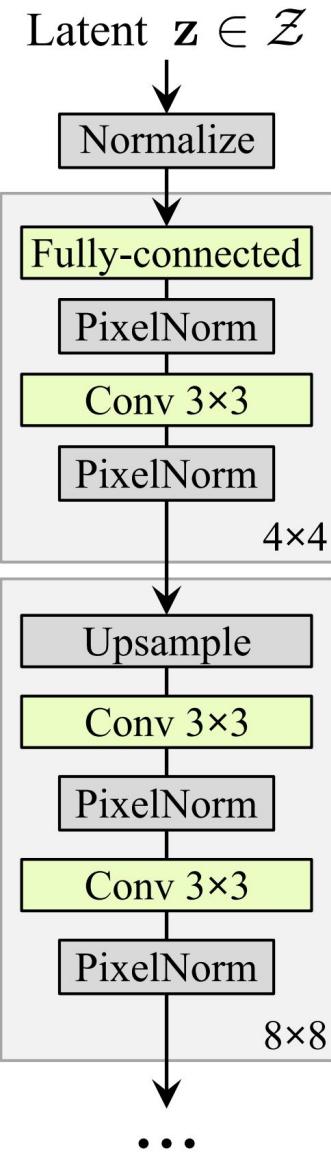


(b) Style-based generator

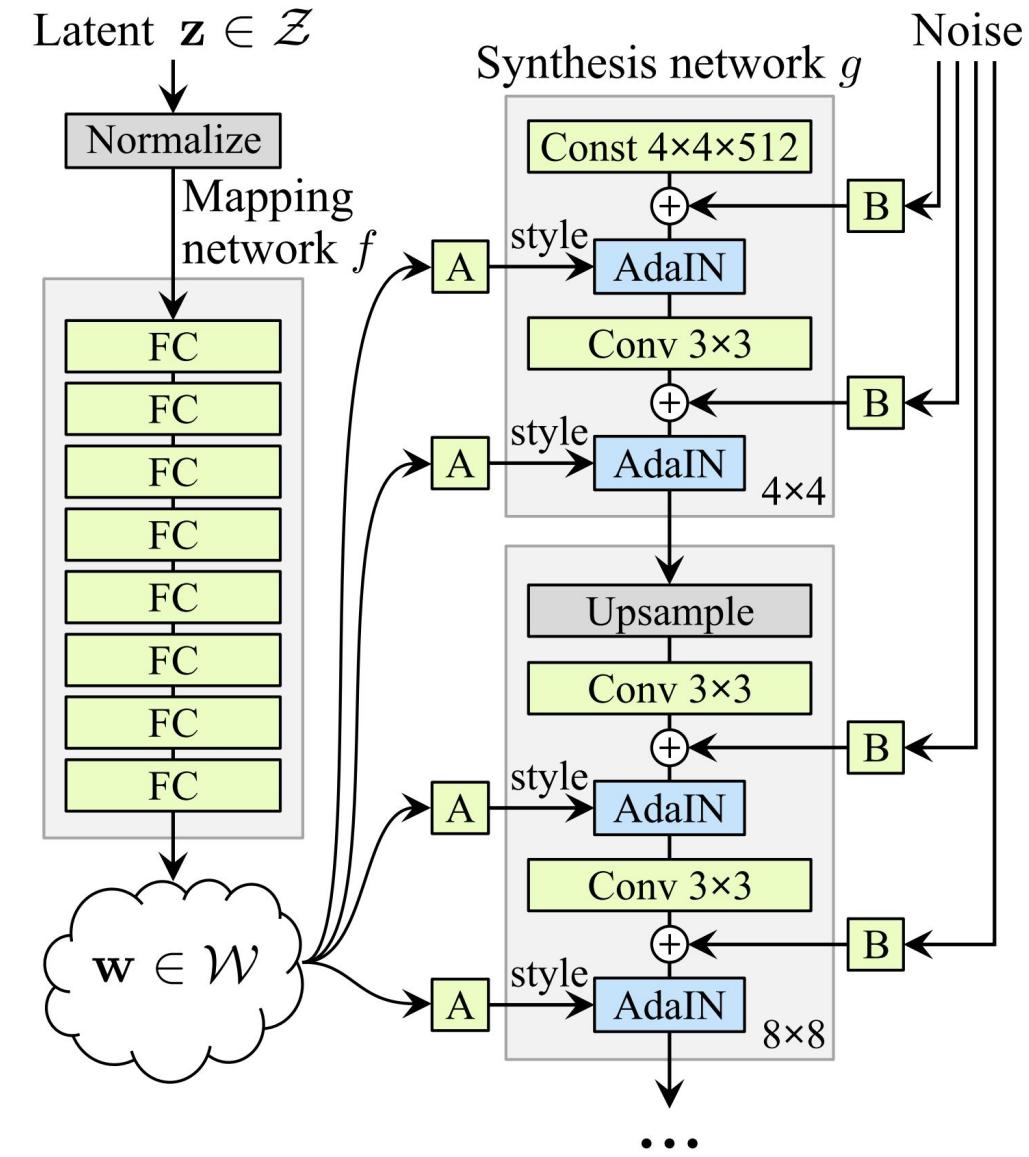
StyleGAN

- ❖ Было: вектор \mathbf{z} из нормального распределения
- ❖ Стало: вытягиваем фичи из \mathbf{z} в \mathbf{w}
- ❖ \mathbf{w} лучше описывает разнообразие
- ❖ AdaIN слои помогают учитывать \mathbf{w} во время генерации

$$\text{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$



(a) Traditional



(b) Style-based generator

Coarse styles from source B

Source A

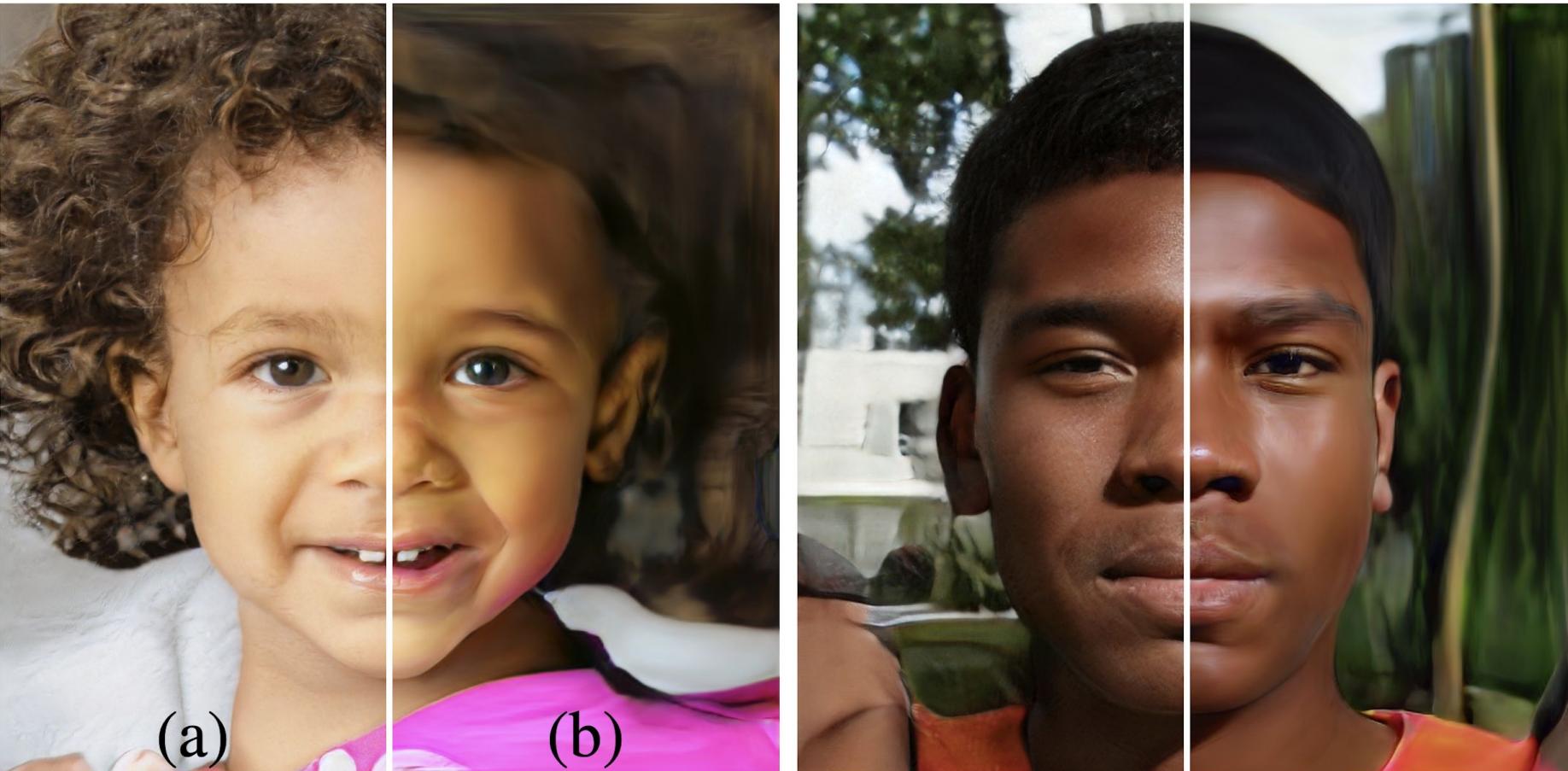


Source B



StyleGAN. Влияние шума

- ❖ Без шума результаты менее детализированные

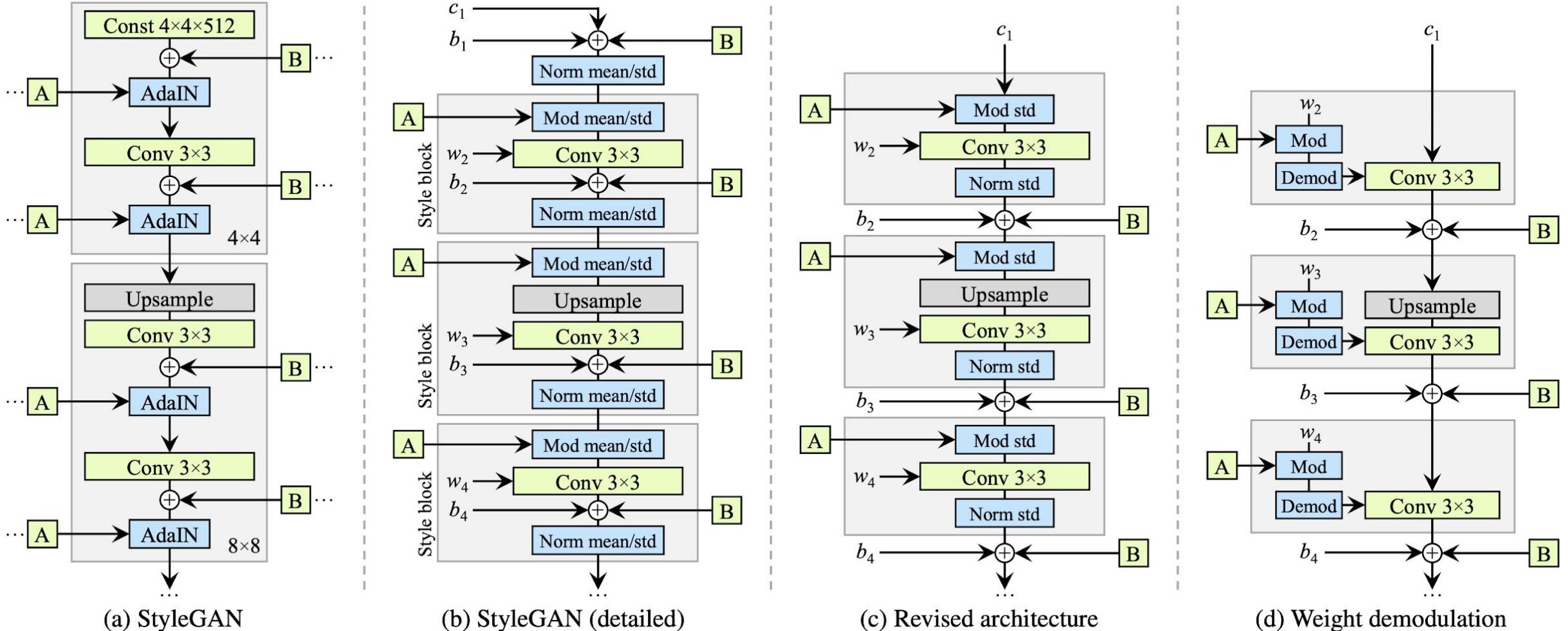


StyleGAN2

- ❖ Исправляет часть проблем StyleGAN
- ❖ Отказывается от progressive growing и AdaIN
- ❖ Результаты лучше



StyleGAN2. Новая архитектура



StyleGAN2. Отказ от progressive growing



Figure 6. Progressive growing leads to “phase” artifacts. In this example the teeth do not follow the pose but stay aligned to the camera, as indicated by the blue line.

Ссылки

- ❖ [Generative Adversarial Nets. Goodfellow et al.](#)
- ❖ [NIPS 2016 Tutorial: Generative Adversarial Networks. I. Goodfellow](#)
- ❖ [Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. Radford et al.](#)
- ❖ [Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Ioffe et al.](#)
- ❖ [Unrolled Generative Adversarial Networks. Metz et al.](#)
- ❖ <https://youtu.be/T4UuL7U5asA>
- ❖ <https://github.com/bayesgroup/deepbayes-2019/blob/master/lectures/day3/1.%20Egor%20Zakharov%20-%20GANs%20.pdf>
- ❖ <https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html#kullbackleibler-and-jensenshannon-divergence>
- ❖ <https://vincentrerrmann.github.io/blog/wasserstein/>
- ❖ Progressive Growing of GANs: <https://www.youtube.com/watch?v=t640zZzIRBY>
- ❖ <https://arxiv.org/abs/1710.10196>
- ❖ <https://towardsdatascience.com/explained-a-style-based-generator-architecture-for-gans-generating-and-tuning-realistic-6cb2be0f431>
- ❖ <https://arxiv.org/pdf/1706.08224.pdf>
- ❖ Полезный и красивый семинар Саши Панчина из курса ШАДа по глубинному обучению:
https://colab.research.google.com/github/yandexdataschool/Practical_DL/blob/spring20/seminar08-generative/simple_1d_gan_pytorch.ipynb
- ❖ <https://arxiv.org/abs/1606.03498>