

ПОРОЖДАЮЩИЕ СОСТАЗАТЕЛЬНЫЕ СЕТИ II

Сергей Николенко

Академия MADE — Mail.Ru

27 ноября 2021 г.

Random facts:

- 27 ноября 1095 г. Урбан II на Клермонском соборе по просьбе византийского императора Алексея I провозгласил Первый крестовый поход
- 27 ноября 1895 г. Альфред Нобель подписал завещание, по которому большая часть его состояния поступала в фонд Нобелевской премии
- 27 ноября — день авиакатастроф: в 1962 г. Boeing 707 врезался в гору при заходе на посадку под Лимой (97 погибших), в 1970 Douglas DC-8 выкатился со взлётной полосы и загорелся (47 погибших), в 1983 Boeing 747 под Мадридом зацепил несколько холмов и разрушился (181 погибших), а в 1989 г. в окрестностях Боготы террористы по указанию Пабло Эскобара взорвали Boeing 727 (110 погибших)
- 27 ноября 1992 г. была создана Высшая школа экономики
- 27 ноября 2009 г. поезд «Невский Экспресс» сошёл с рельсов на границе Тверской и Новгородской областей; по официальной версии, это был теракт, и ответственность за подрыв взял на себя лидер «Кавказского эмирата» Доку Умаров

ЕЩЁ НЕМНОГО ФУНКЦИЙ ОШИБКИ

EBGAN и BEGAN

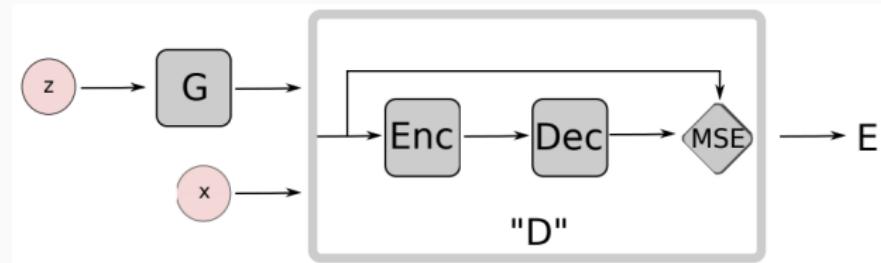
- (Zhao et al., 2016): Energy-Based Generative Adversarial Networks (EBGAN)
- Давайте рассмотрим дискриминатор как функцию энергии, которая присваивает низкие значения энергии областям вблизи распределения данных и высокие всем остальным.
- Генератор должен порождать контрастные сэмплы с минимальными значениями энергии.
- Такой взгляд позволяет в качестве дискриминатора использовать что угодно, не обязательно классификатор с логистическим сигмоидом; например, шарнирную функцию

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

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

EBGAN и BEGAN

- Вторая идея из (Zhao et al., 2016) – использовать в качестве дискриминатора автокодировщик, т.е.
 $D(\mathbf{x}) = \|\text{Dec}(\text{Enc}(\mathbf{x})) - \mathbf{x}\|:$



- Плюс ещё repelling regularizer – давайте потребуем, чтобы сэмплы в мини-батче были как можно дальше друг от друга, чтобы улучшить diversity.

EBGAN и BEGAN

- (Berthelot et al., 2017): Boundary Equilibrium Generative Adversarial Networks (BEGAN)
- Давайте добавим в EBGAN немножко Вассерштейна: будем минимизировать различие (расстояние Вассерштейна) между ошибками реконструкции для настоящих и порождённых примеров.
- Тоже становится лучше:

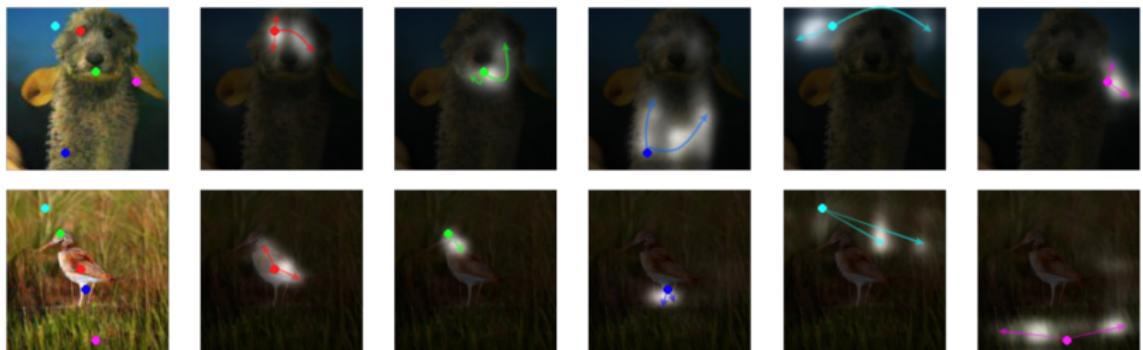


(a) EBGAN (64x64)

(b) Our results (128x128)

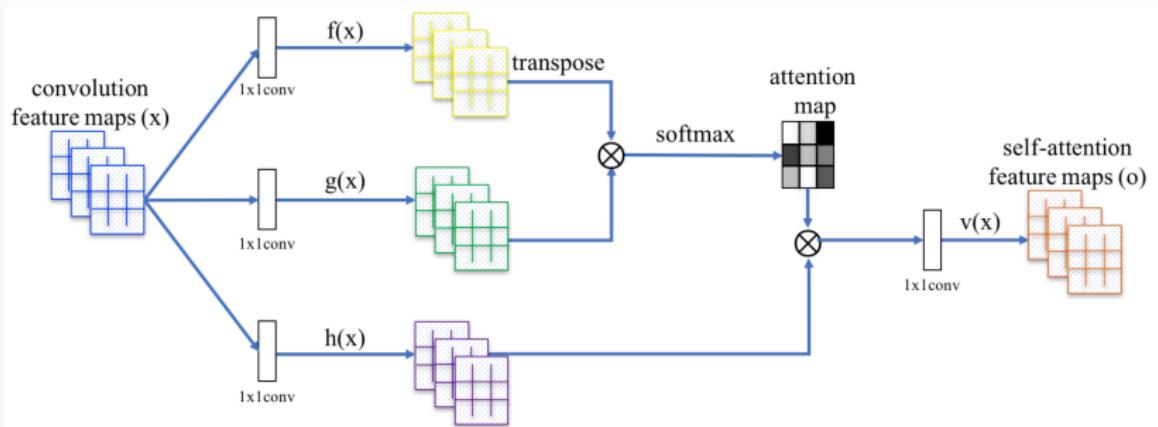
SAGAN

- (Zhang et al., 2019): Self-Attention Generative Adversarial Networks (SAGAN)
- Пытаются решить проблему с локальностью порождения свёрточными сетями

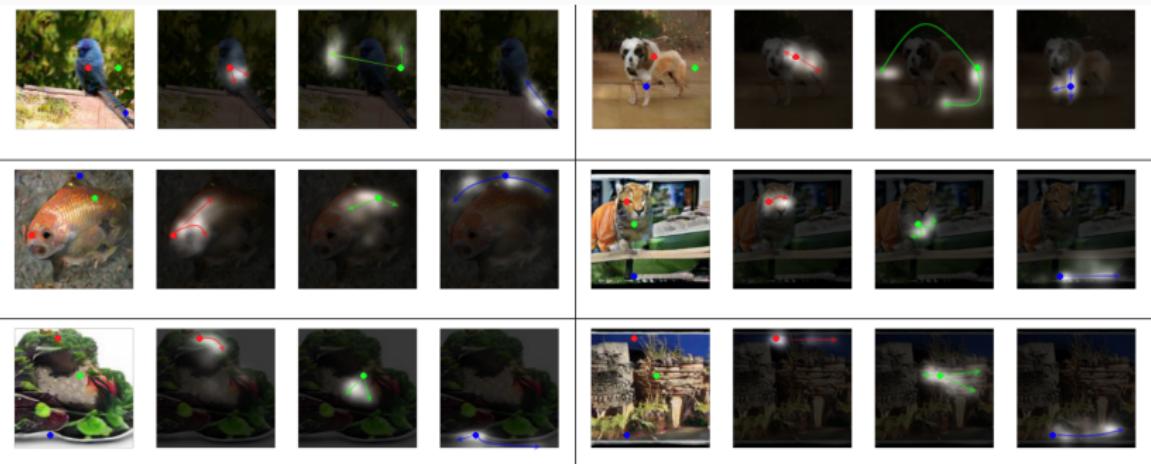


SAGAN

- Self-attention здесь примерно как в Transformer, только переложенный на картинки:

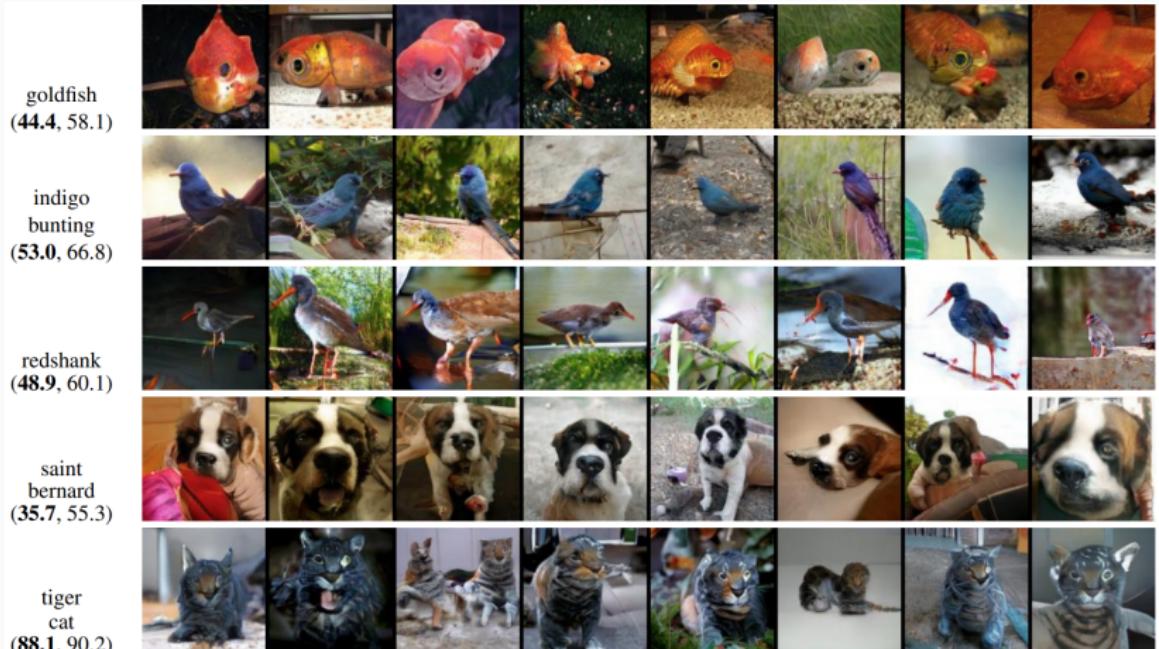


- И получается, что действительно генератор (здесь примеры с последнего слоя) может смотреть достаточно далеко, когда порождает картинку:

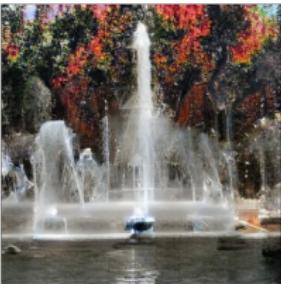


SAGAN

- В исходной статье (Zhang et al., 2019) уже получаются хорошие картинки:

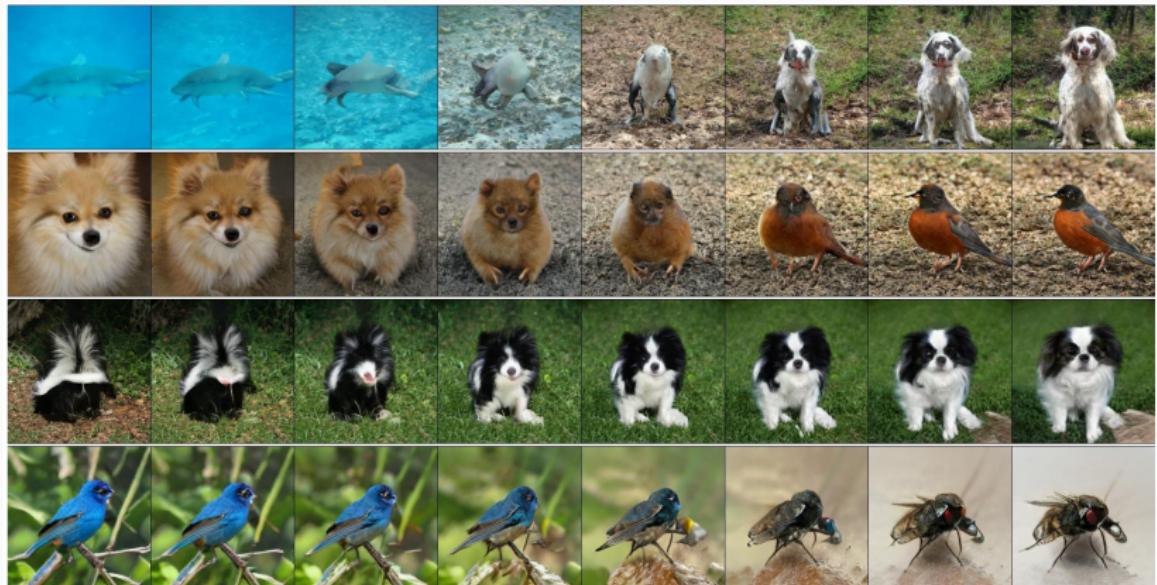


- Следующий этап – BigGAN (Brock et al., 2019), улучшил Inception score в три раза...



- Используют SA-GAN (GAN с self-attention) со всеми новыми трюками и специальными архитектурами

- Интерполяции:



- Ой, а что такое «лучше»?..

- Трудно оценивать качество картинок.
- Inception score (Salimans et al., 2016; вообще важная статья о GAN'ах):
 - берём порождённые картинки \mathbf{x} ;
 - применяем стандартный классификатор Inception;
 - хотим, чтобы у $p(y | \mathbf{x})$ была маленькая энтропия, но разнообразие чтобы было, т.е. чтобы у $p(y) = \int p(y | \mathbf{x} = G(\mathbf{z})) d\mathbf{z}$ была большая энтропия;
 - т.е. получаем метрику $\exp(\mathbb{E}_{\mathbf{x}} \text{KL}(p(y | \mathbf{x}) \| p(y)))$;
 - для обучения это трудно приспособить, а вот для оценки качества хорошо.

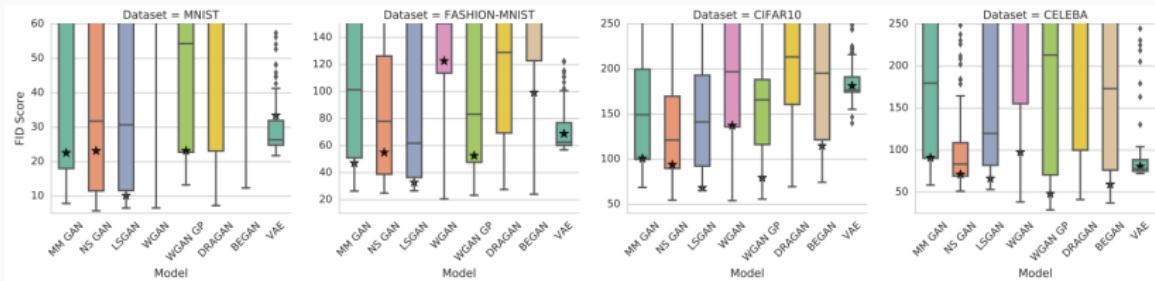
ARE GANs CREATED EQUAL?

- Впрочем, возможно, это всё ерунда.
- (Lucic et al., 2018): Are GANs Created Equal? Большое экспериментальное сравнение от Google Brain.

GAN	DISCRIMINATOR LOSS	GENERATOR LOSS
MM GAN	$\mathcal{L}_D^{GAN} = -\mathbb{E}_{x \sim p_d} [\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$	$\mathcal{L}_G^{GAN} = \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$
NS GAN	$\mathcal{L}_D^{NSGAN} = -\mathbb{E}_{x \sim p_d} [\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$	$\mathcal{L}_G^{NSGAN} = -\mathbb{E}_{\hat{x} \sim p_g} [\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_D^{WGAN} = -\mathbb{E}_{x \sim p_d} [D(x)] + \mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})]$	$\mathcal{L}_G^{WGAN} = -\mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})]$
WGAN GP	$\mathcal{L}_D^{WGANGP} = \mathcal{L}_D^{WGAN} + \lambda \mathbb{E}_{\hat{x} \sim p_g} [(\nabla D(\alpha x + (1 - \alpha)\hat{x}) _2 - 1)^2]$	$\mathcal{L}_G^{WGANGP} = -\mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})]$
LS GAN	$\mathcal{L}_D^{LSGAN} = -\mathbb{E}_{x \sim p_d} [(D(x) - 1)^2] + \mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})^2]$	$\mathcal{L}_G^{LSGAN} = -\mathbb{E}_{\hat{x} \sim p_g} [(D(\hat{x} - 1))^2]$
DRAGAN	$\mathcal{L}_D^{DRAGAN} = \mathcal{L}_D^{GAN} + \lambda \mathbb{E}_{\hat{x} \sim p_d + \mathcal{N}(0, c)} [(\nabla D(\hat{x}) _2 - 1)^2]$	$\mathcal{L}_G^{DRAGAN} = \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$
BEGAN	$\mathcal{L}_D^{BEGAN} = \mathbb{E}_{x \sim p_d} [x - AE(x) _1] - k_t \mathbb{E}_{\hat{x} \sim p_g} [\hat{x} - AE(\hat{x}) _1]$	$\mathcal{L}_G^{BEGAN} = \mathbb{E}_{\hat{x} \sim p_g} [\hat{x} - AE(\hat{x}) _1]$

ARE GANs CREATED EQUAL?

- Все GAN'ы очень чувствительны к гиперпараметрам:

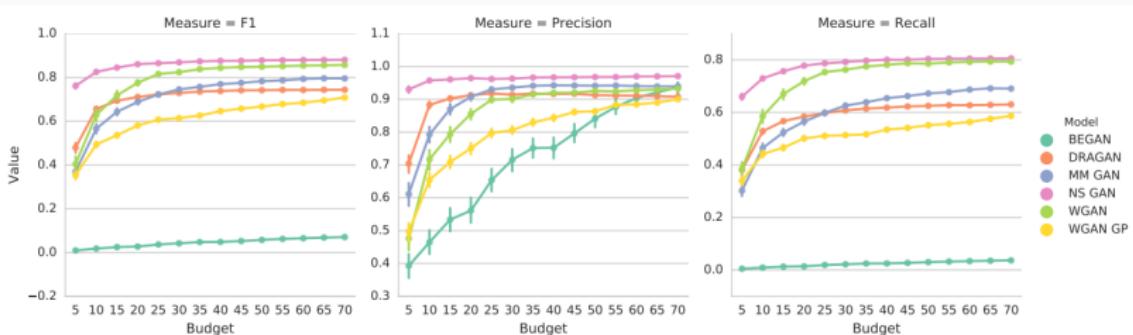
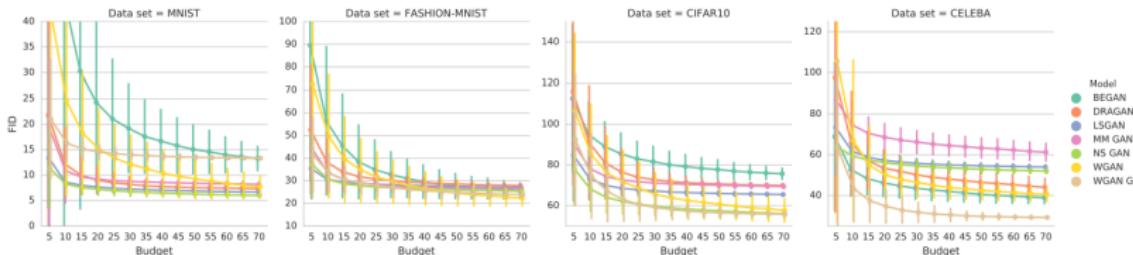


- Кстати, метрика здесь – Frechet Inception Distance (FID): считаем средние и матрицы ковариаций кодов настоящих и фейковых данных и считаем между ними расстояние Фреше

$$d^2((\mathbf{m}, \mathbf{C}), (\mathbf{m}', \mathbf{C}')) = \|\mathbf{m} - \mathbf{m}'\|_2^2 + \text{Tr} \left(\mathbf{C} + \mathbf{C}' - 2(\mathbf{C}\mathbf{C}')^{1/2} \right).$$

ARE GANs CREATED EQUAL?

- Различия есть, но они стираются по мере того, как мы увеличиваем computational budget:



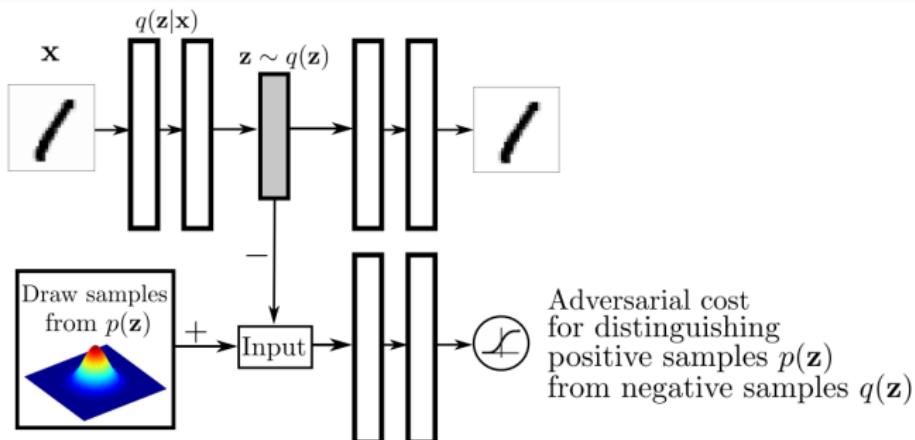
Что сейчас с GAN'ами делают

- Как видите, GAN'ы – очень интересная наука, много математики, настоящие теоремы надо доказывать...
- Но, конечно, тут же выяснилось, что часто достаточно просто хорошо придумать loss functions из общих соображений. Чем все и занимаются.

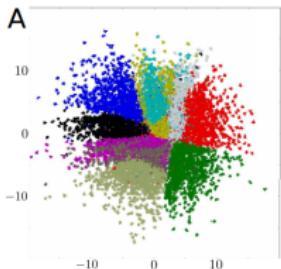


АРХИТЕКТУРЫ, ОСНОВАННЫЕ НА GAN

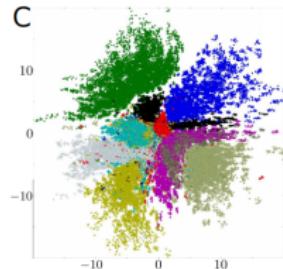
- Соперничающие автокодировщики (adversarial autoencoders; Makhzani et al., 2015): дискриминатор приводит распределение признаков (на скрытом слое) к заданному $p(\mathbf{z})$.



Adversarial Autoencoder

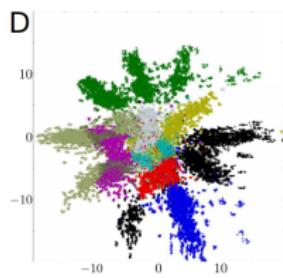
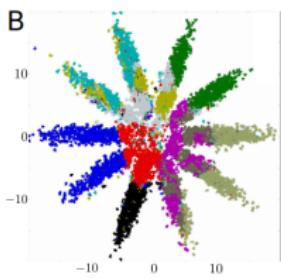


Variational Autoencoder

Manifold of
Adversarial Autoencoder

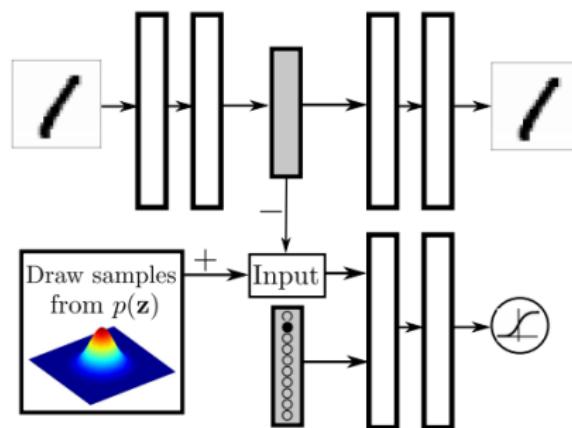
E

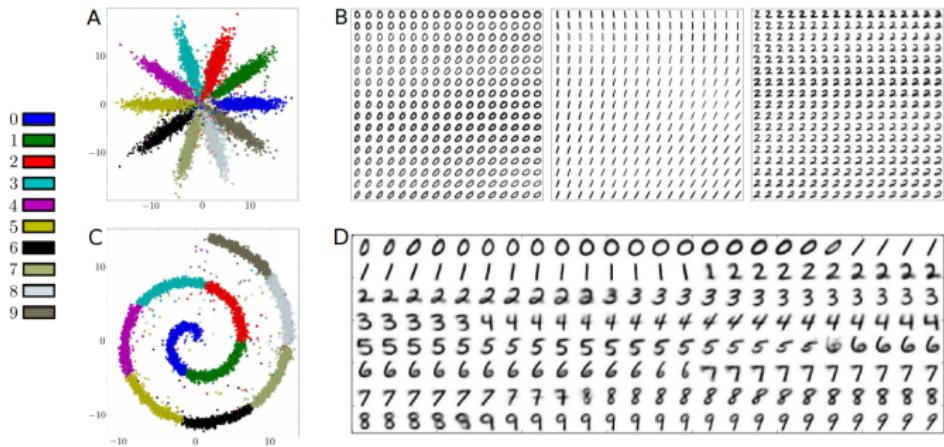
0	0	0	0	5	5	3	3	3	3	3	5	5	5	8	8	8	2
0	0	0	0	5	5	3	3	3	3	3	5	5	5	8	8	8	2
0	0	0	0	0	5	3	3	3	3	3	3	8	8	8	2	2	
0	0	0	0	0	5	3	3	3	3	3	3	8	8	8	2	2	
6	6	6	0	0	3	3	3	3	3	3	5	5	8	8	8	2	
6	6	6	6	6	6	5	5	5	5	5	8	8	8	2	2		
6	6	6	6	6	6	5	5	5	5	5	8	8	8	2	2		
6	6	6	6	6	6	5	5	5	5	5	8	8	8	2	2		
6	6	6	6	6	6	5	5	5	5	5	8	8	8	2	2		
4	4	4	4	4	4	4	4	4	4	4	5	5	8	8	8	2	
4	4	4	4	4	4	4	4	4	4	4	7	7	7	2	2		
4	4	4	4	4	4	4	4	4	4	4	7	7	7	2	2		
4	4	4	4	4	4	4	4	4	4	4	7	7	7	2	2		
4	4	4	4	4	4	4	4	4	4	4	7	7	7	2	2		
4	4	4	4	4	4	4	4	4	4	4	7	7	7	2	2		
4	4	4	4	4	4	4	4	4	4	4	7	7	7	2	2		
4	4	4	4	4	4	4	4	4	4	4	7	7	7	2	2		
4	4	4	4	4	4	4	4	4	4	4	7	7	7	2	2		
4	4	4	4	4	4	4	4	4	4	4	7	7	7	2	2		
7	7	7	7	7	7	7	7	7	7	7	7	7	7	1	1	1	
7	7	7	7	7	7	7	7	7	7	7	7	7	7	1	1	1	
7	7	7	7	7	7	7	7	7	7	7	7	7	7	1	1	1	
7	7	7	7	7	7	7	7	7	7	7	7	7	7	1	1	1	
7	7	7	7	7	7	7	7	7	7	7	7	7	7	1	1	1	



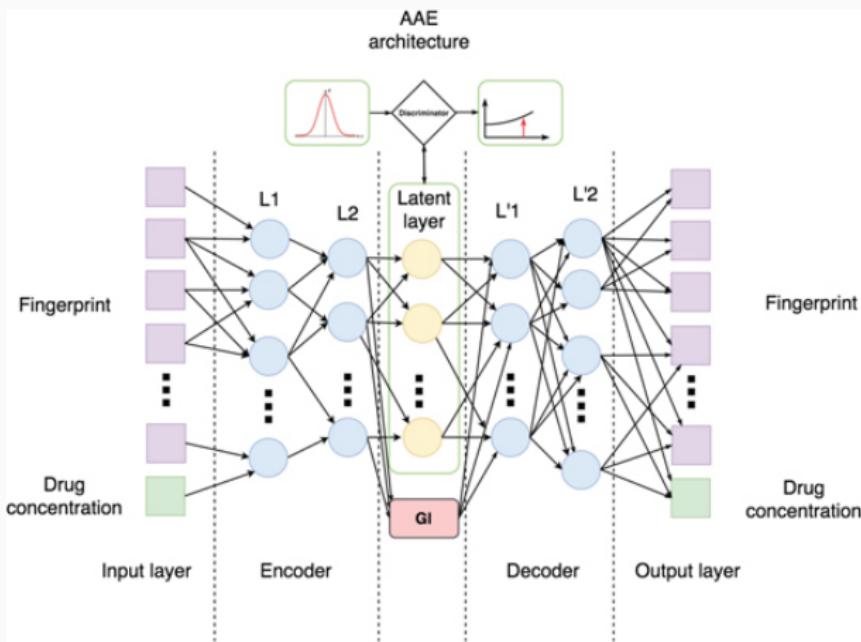
- 0 5
- 1 6
- 2 7
- 3 8
- 4 9

- Можно сделать распределения условными и получить semi-supervised learning:

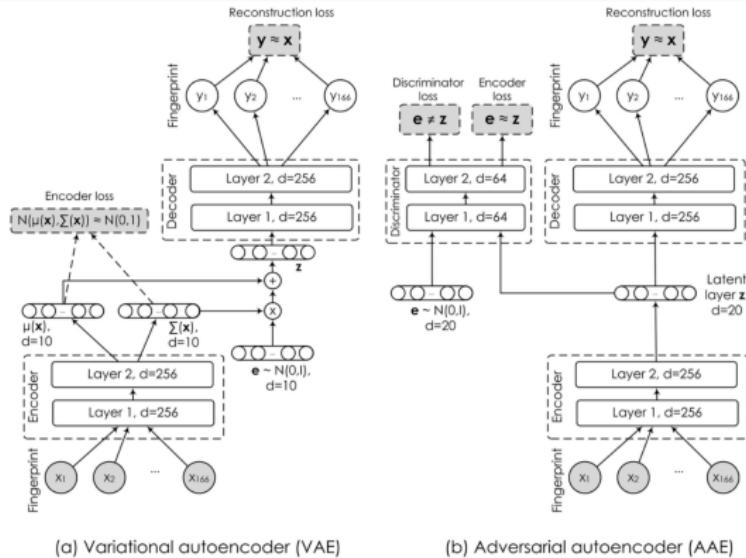




- (Kadurin et al., 2017) – ААЕ для порождения молекул в онкологии:



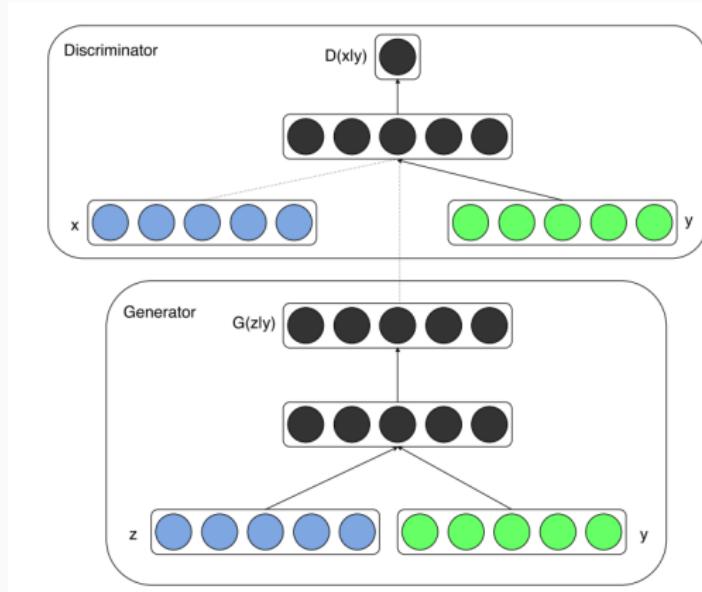
- (Kadurin et al., 2018) – druGAN для порождения молекул; сравнили с VAE:



- (Polikovsky et al., 2018): MOSES – платформа для сравнения порождающих моделей для молекул

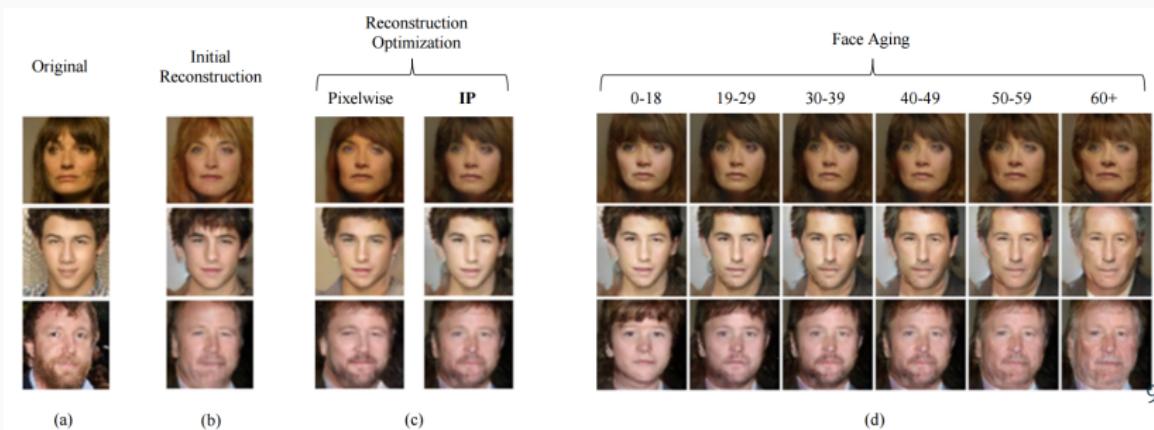
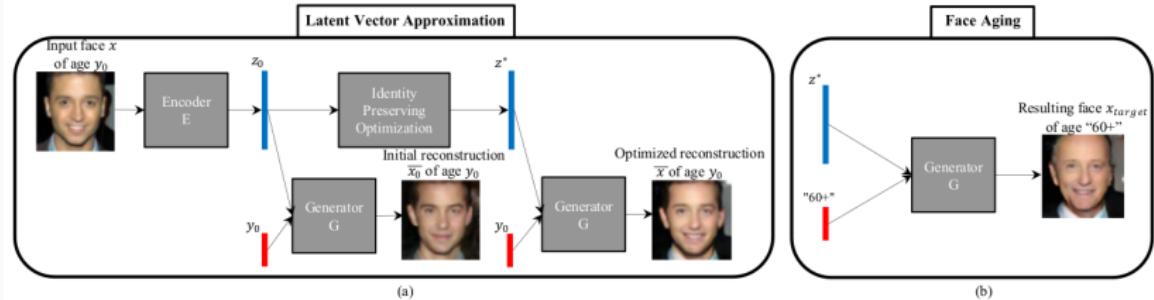
Условные GAN'ы

- Условные GAN'ы (conditional GANs; Mirza and Osindero, 2014):
 G получает случайный вектор и вектор входа, вход – это условие.



УСЛОВНЫЕ GAN'ы

- (Antipov et al., 2017) – состарим лицо условным GAN'ом:



Условные GAN'ы

- (Ledig et al., 2017) – ESRGAN для superresolution (но об этом можно долго разговаривать)



STACKED GANs

- Часто полезно сделать несколько GAN'ов подряд.
- (Huang et al., 2017): Stacked Generative Adversarial Networks
- G_i последовательно обучает более низкоуровневые представления, обращая глубокую сеть из E_i .
- Новые loss functions:
 - conditional loss – мы обучаем $\hat{\mathbf{h}}_i$ при условии \mathbf{h}_{i+1} , и чтобы генератор не игнорировал условие, надо, чтобы восстанавливались \mathbf{h}_{i+1} через соответствующий encoder:

$$\mathcal{L}^{\text{cond}} = \mathbb{E}_{\mathbf{h}_{i+1} \sim p_{\text{data}}, \mathbf{z}_i \sim p_{\mathbf{z}_i}} [d(E_i(G_i(\mathbf{h}_{i+1}, \mathbf{z}_i)), \mathbf{h}_{i+1})];$$

- entropy loss – но и надо, чтобы G_i не игнорировал \mathbf{z} , надо добавить diversity; для этого максимизируем энтропию:

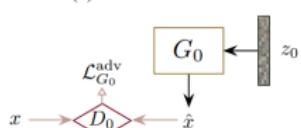
$$\mathcal{L}^{\text{ent}} = \mathbb{E}_{\mathbf{z}_i \sim p_{\mathbf{z}_i}} \left[\mathbb{E}_{\hat{\mathbf{h}}_i \sim G_i(\hat{\mathbf{h}}_i | \mathbf{z}_i)} \left[-\log Q_i(\mathbf{z}_i | \hat{\mathbf{h}}_i) \right] \right],$$

где Q_i – параметризованное нейросетью приближение для апостериорного распределения $p_i(\mathbf{z}_i | \hat{\mathbf{h}}_i)$ (вариационная оценка).

STACKED GANs

- SGAN:

(a) A vanilla GAN



Encoder forward path

\leftarrow Noise

Independent training path

Joint training path

Conditional loss

Adversarial loss

Entropy loss

Generator forward path

\downarrow

Joint training path

Entropy loss

Q-Net

$\mathcal{L}_{G_2}^{\text{cond}}$

E_2

y

\hat{h}_2

$\mathcal{L}_{G_2}^{\text{adv}}$

D_2

\hat{x}

$\mathcal{L}_{G_1}^{\text{cond}}$

E_1

y

\hat{h}_1

$\mathcal{L}_{G_1}^{\text{adv}}$

D_1

\hat{x}

$\mathcal{L}_{G_0}^{\text{cond}}$

E_0

y

\hat{h}_0

$\mathcal{L}_{G_0}^{\text{adv}}$

D_0

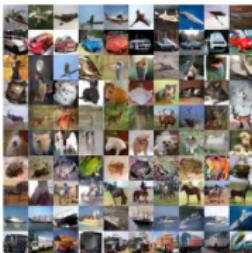
\hat{x}

(b) SGAN Train

(c) SGAN Test

STACKED GANs

- Получается лучше, чем у предшественников (хотя ещё лучше progressive growing):



(a) SGAN samples (conditioned on labels)



(b) Real images (nearest neighbor)



(c) SGAN samples (conditioned on generated fc3 features)



(d) SGAN samples (conditioned on generated fc3 features, trained without entropy loss)

Method	Score
Infusion training [1]	4.62 ± 0.06
ALI [10] (as reported in [63])	5.34 ± 0.05
GMAN [11] (best variant)	6.00 ± 0.19
EGAN-Ent-VI [4]	7.07 ± 0.10
LR-GAN [65]	7.17 ± 0.07
Denoising feature matching [63]	7.72 ± 0.13
DCGAN [†] (with labels, as reported in [61])	6.58
SteinGAN [†] [61]	6.35
Improved GAN [†] [53] (best variant)	8.09 ± 0.07
AC-GAN [†] [43]	8.25 ± 0.07
DCGAN (\mathcal{L}^{adv})	6.16 ± 0.07
DCGAN ($\mathcal{L}^{adv} + \mathcal{L}^{ent}$)	5.40 ± 0.16
DCGAN ($\mathcal{L}^{adv} + \mathcal{L}^{cond}$) [†]	5.40 ± 0.08
DCGAN ($\mathcal{L}^{adv} + \mathcal{L}^{cond} + \mathcal{L}^{ent}$) [†]	7.16 ± 0.10
SGAN-no-joint [†]	8.37 ± 0.08
SGAN [†]	8.59 ± 0.12
Real data	11.24 ± 0.12

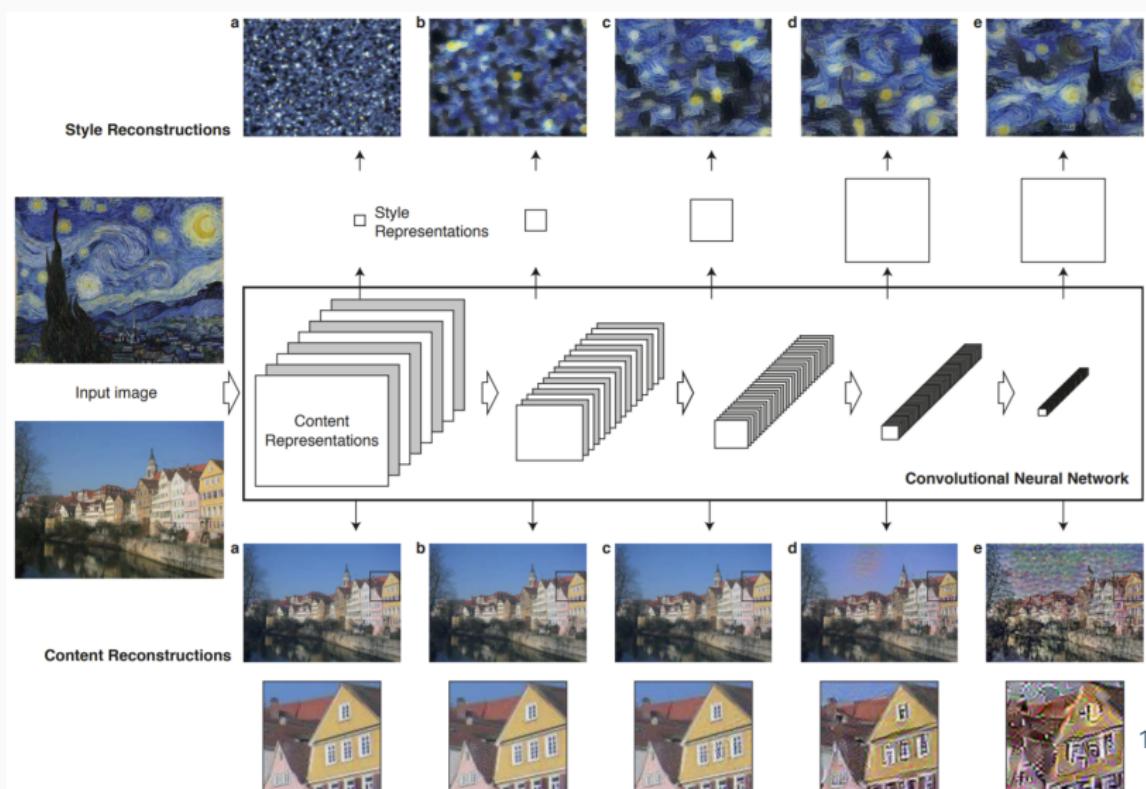
[†] Trained with labels.

Table 1: Inception Score on CIFAR-10. SGAN and SGAN-no-joint outperform previous state-of-the-art approaches.

CASE STUDY: ПЕРЕНОС СТИЛЯ

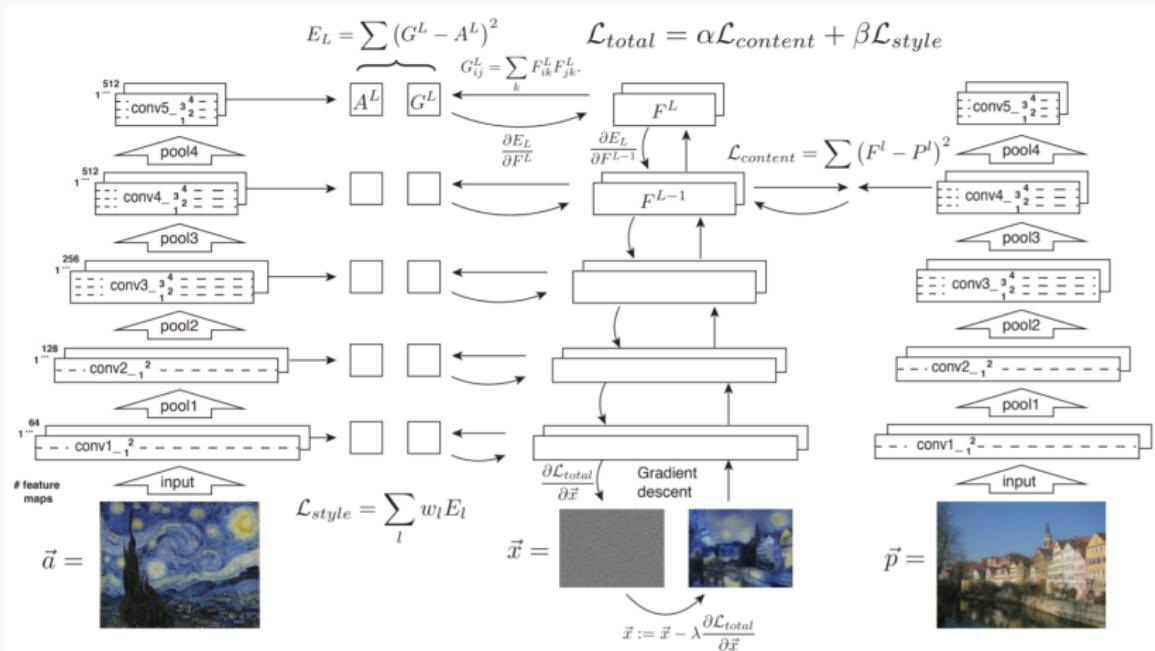
ПЕРЕНОС СТИЛЯ до GAN'ов

- (Gatys et al., 2015): перенос стиля свёрточными сетями



ПЕРЕНОС СТИЛЯ до GAN'ов

- (Gatys et al., 2015): перенос стиля свёрточными сетями



ПЕРЕНОС СТИЛЯ до GAN'ов

- (Gatys et al., 2015): перенос стиля свёрточными сетями

A



B



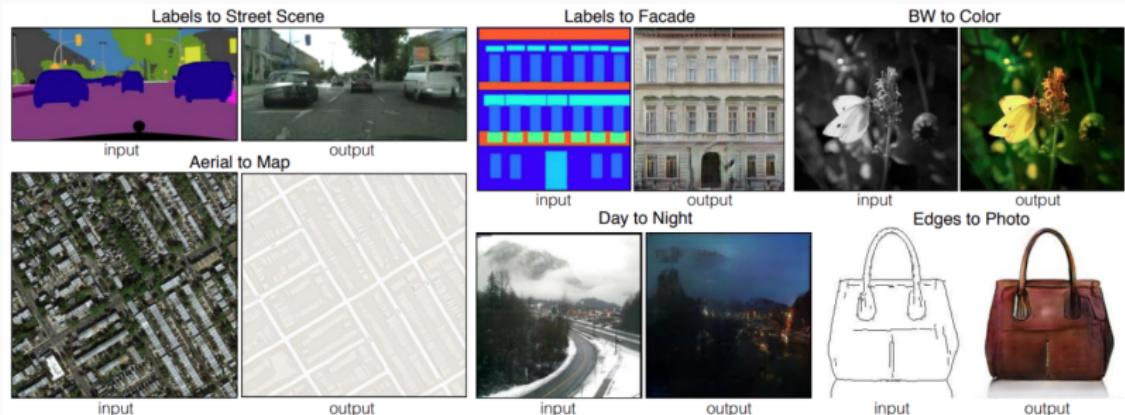
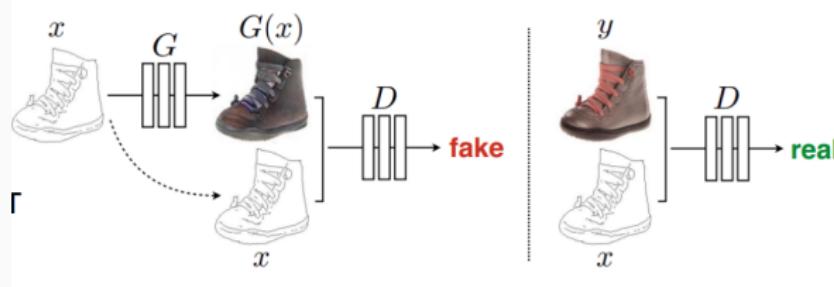
ПЕРЕНОС СТИЛЯ до GAN'ов

- (Gatys et al., 2015): перенос стиля свёрточными сетями



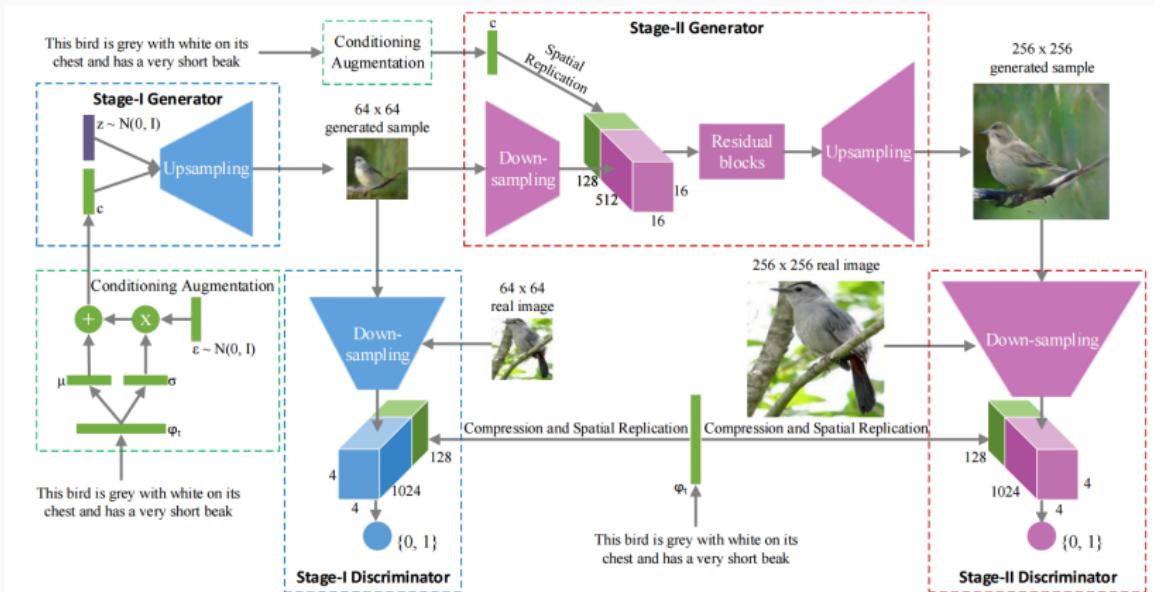
ПЕРЕНОС СТИЛЯ с GAN'ами

- pix2pix (Isola et al., 2017): отлично работает с paired dataset



ПЕРЕНОС СТИЛЯ с GAN'ами

- Стиль не обязан быть изображением!
- StackGAN (Zhang et al., 2016) – по тексту породим картинку:



ПЕРЕНОС СТИЛЯ с GAN'ами

- Вот результаты реального порождения (best of 16):



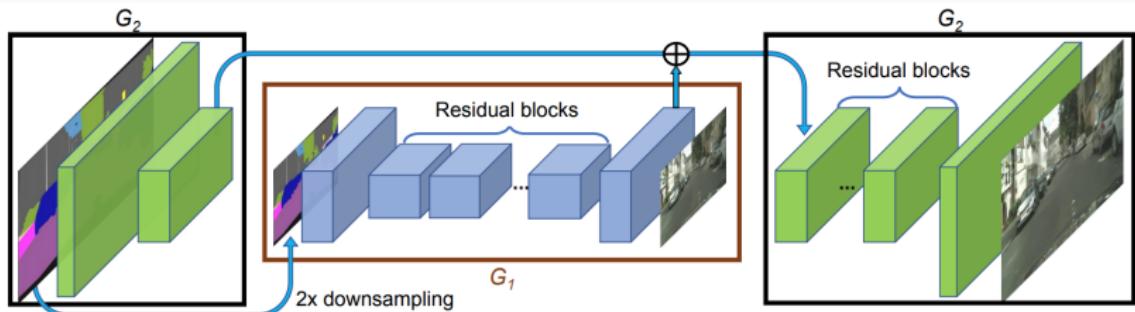
ПЕРЕНОС СТИЛЯ с GAN'ами

- pix2pixHD (Wang et al., 2017) – то же самое в высоком разрешении



ПЕРЕНОС СТИЛЯ с GAN'ами

- Генератор теперь из двух частей, вторая улучшает результат первой



ПЕРЕНОС СТИЛЯ с GAN'ами

- Ещё кое-какие трюки, в общем, лучше получается:



ПЕРЕНОС СТИЛЯ с GAN'ами

- А параллельно с этим Huang и Belongie придумали AdaIN (adaptive instance normalization):
 - batch normalization использует статистики по мини-батчу:

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

- instance normalization IN(x) – это то же самое, но статистики по каждому каналу и каждой картинке по отдельности
- conditional instance normalization – это когда γ и β обучаются для каждого стиля:

$$\text{CIN}(x; s) = \gamma_s \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta_s$$

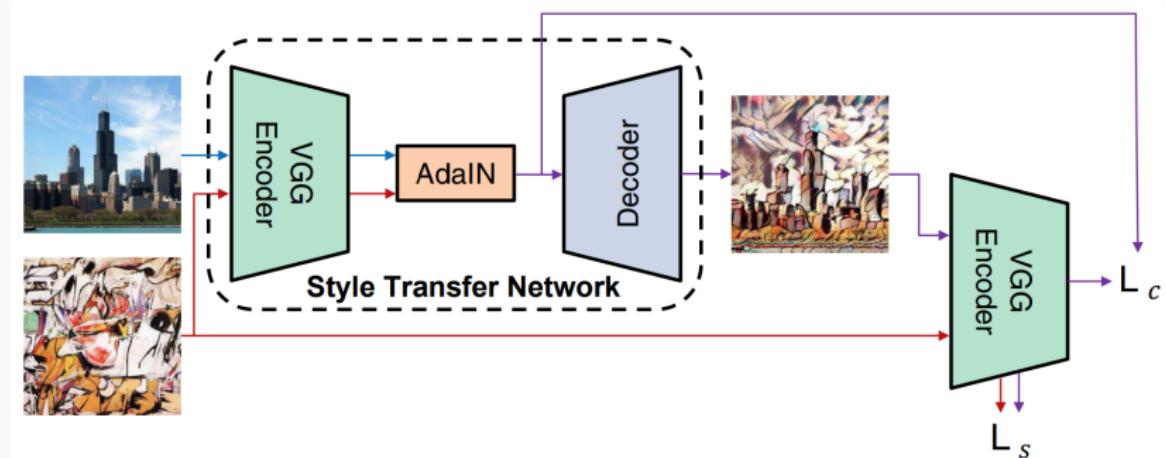
- AdaIN – это то же самое, но мы просто берём параметры от другой картинки, которая задаёт стиль:

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

- Чем-то похоже на самый ранний artistic style transfer (Gatys

ПЕРЕНОС СТИЛЯ с GAN'ами

- Теперь сама сеть супер-простая – AdaIN действует в feature space какого-нибудь автокодировщика:



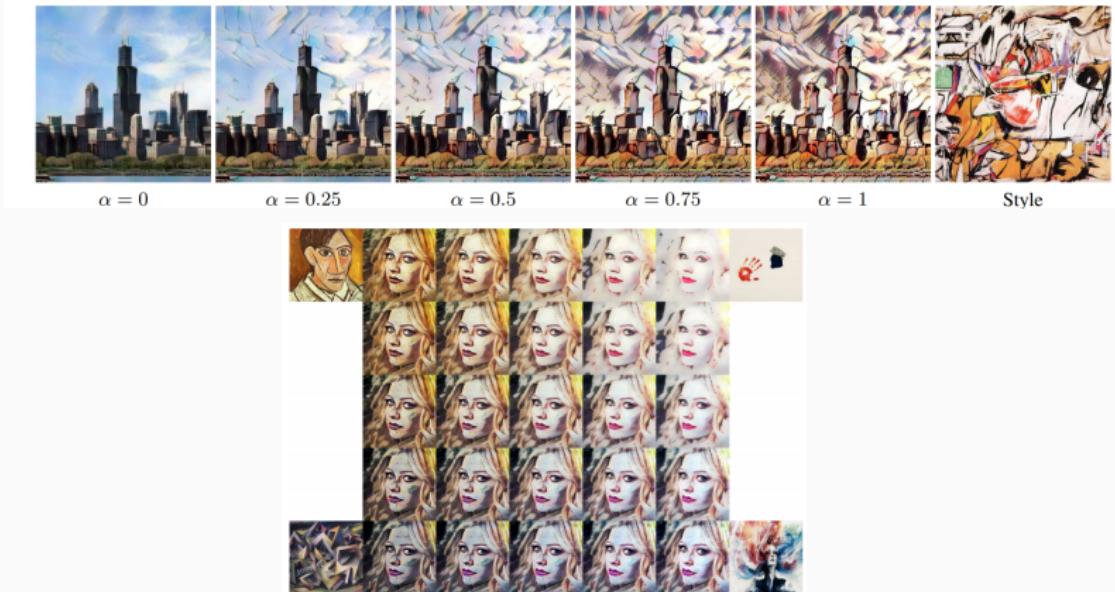
ПЕРЕНОС СТИЛЯ с GAN'ами

- И получается круто:



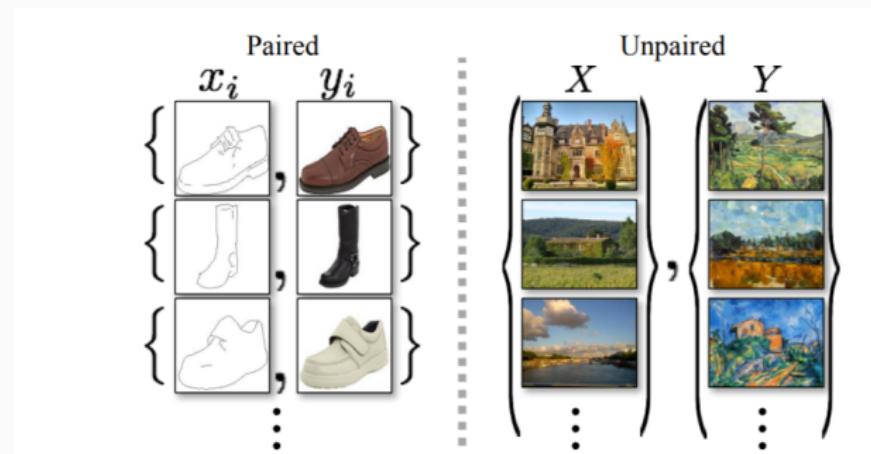
ПЕРЕНОС СТИЛЯ с GAN'ами

- А ещё можно интерполировать и задавать «силу» стиля:



CYCLEGAN

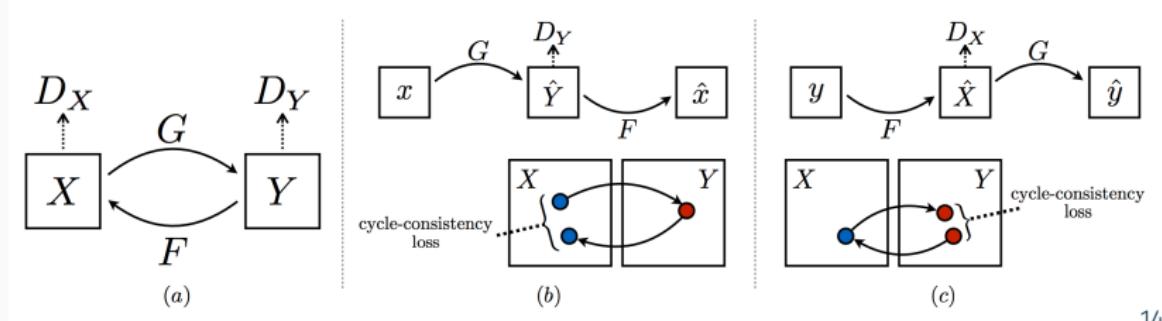
- CycleGAN (Zhu et al., 2017)
 - что делать, если данные unpaired?
 - например, для style transfer:



CYCLEGAN

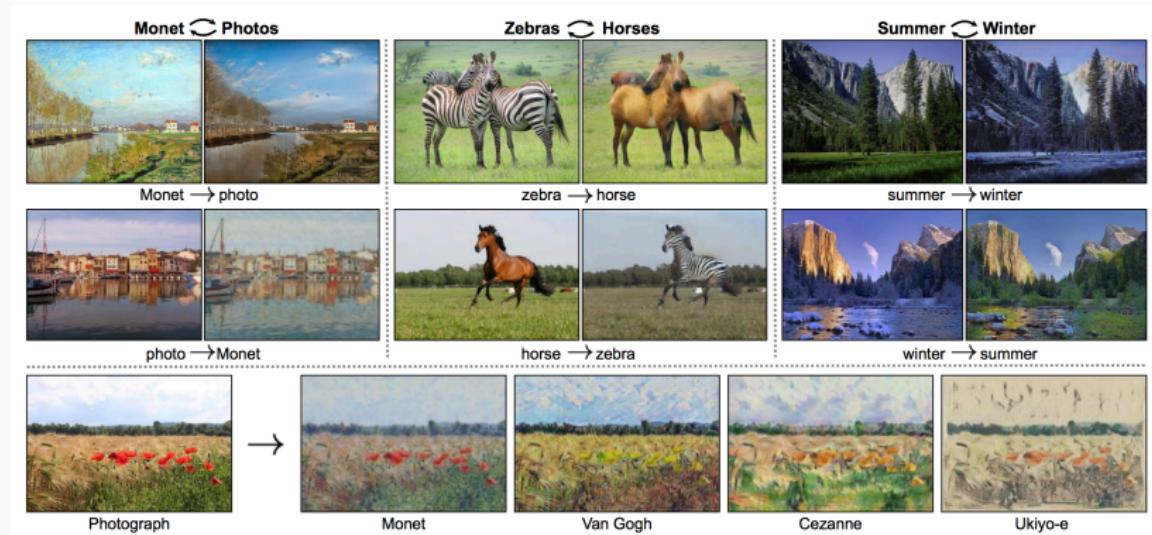
- CycleGAN (Zhu et al., 2017)
- Основная идея в том, что после двойного цикла style transfer должно опять получиться то же самое, $G(F(\mathbf{x})) = \mathbf{x}$
- Общая функция потерь складывается из двух GAN'ов для G и F и cycle loss:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = & \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ & + \lambda \mathcal{L}_{\text{cyc}}(G, F)\end{aligned}$$



CYCLEGAN

- CycleGAN (Zhu et al., 2017)
- И получается очень хороший style transfer без всяких выровненных данных



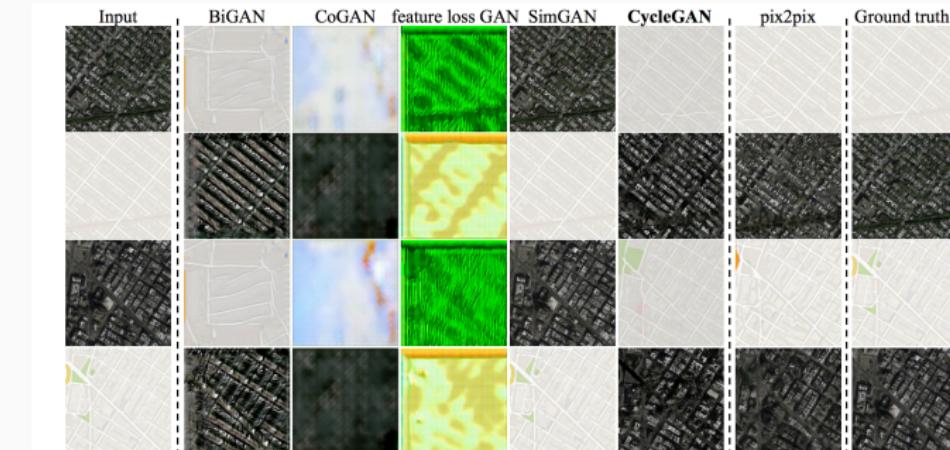
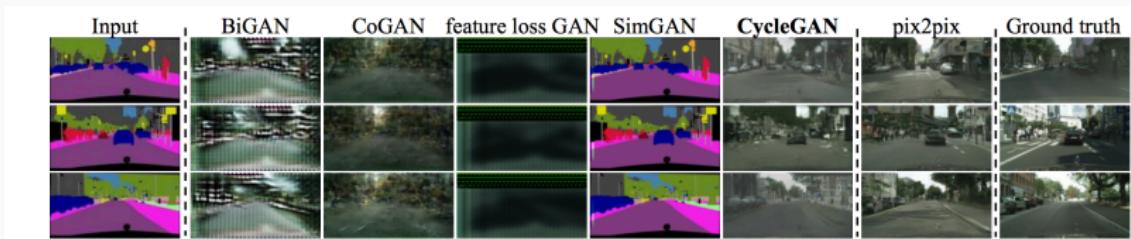
CYCLEGAN

- CycleGAN (Zhu et al., 2017)
- Можно посмотреть на реконструкции тоже



CYCLEGAN

- CycleGAN (Zhu et al., 2017)
- А можно сравнить с предшественниками

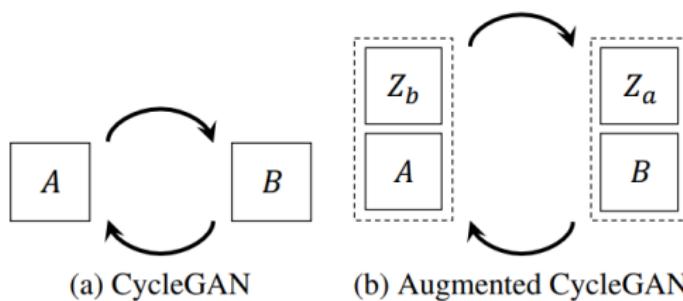


AUGMENTED CYCLEGAN

- Augmented CycleGAN (Almahairi et al., 2018):
 - у CycleGAN есть ограничение – только one-to-one отображения обучаются, у модели один вход соответствует одному выходу
 - а на самом деле всё обычно сложнее, отображения many-to-many в жизни;
 - даже спутниковый снимок из карты можно по-разному сделать, а если, например, лицо из признаков...
- Как вообще обучить many-to-many?

AUGMENTED CYCLEGAN

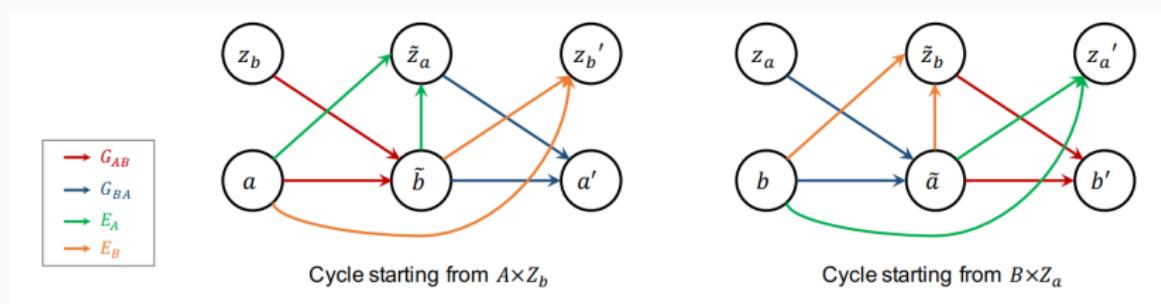
- Надо добавить латентный код, который можно менять:



- Тогда во всех функциях потерь добавится просто ожидание по z

AUGMENTED CYCLEGAN

- Итого мы обучаем $G_{AB} : A \times Z_b \mapsto B$ и $G_{BA} : B \times Z_a \mapsto A$
- Ещё нужны encoders $E_A : A \times B \mapsto Z_a$, $E_B : A \times B \mapsto Z_b$:



AUGMENTED CYCLEGAN

- Получаются разнообразные сэмплы:



(a) AugCGAN



(b) StochCGAN

AUGMENTED CYCLEGAN

- А вот циклы:



(c) AugCGAN



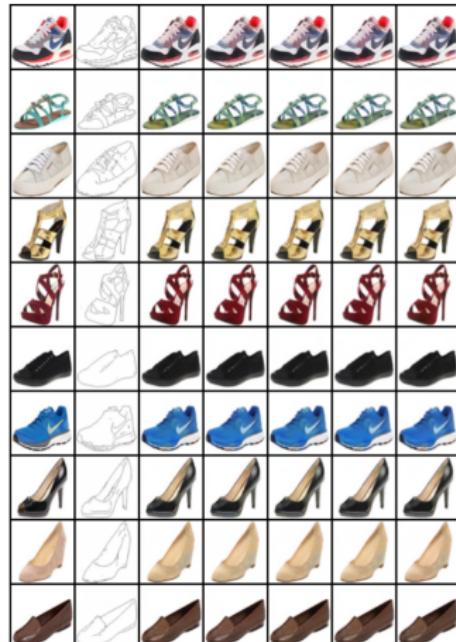
(d) StochCGAN

AUGMENTED CYCLEGAN

- У AugCGAN получается в цикле разнообразие:



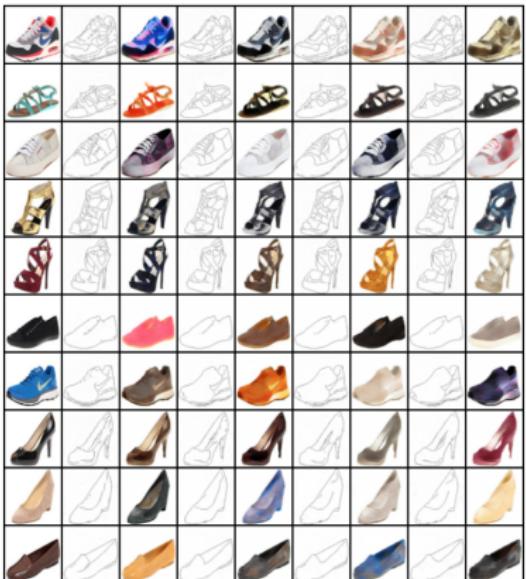
(a) AugCGAN



(b) StochCGAN

AUGMENTED CYCLEGAN

- У AugCGAN получается в цикле разнообразие:



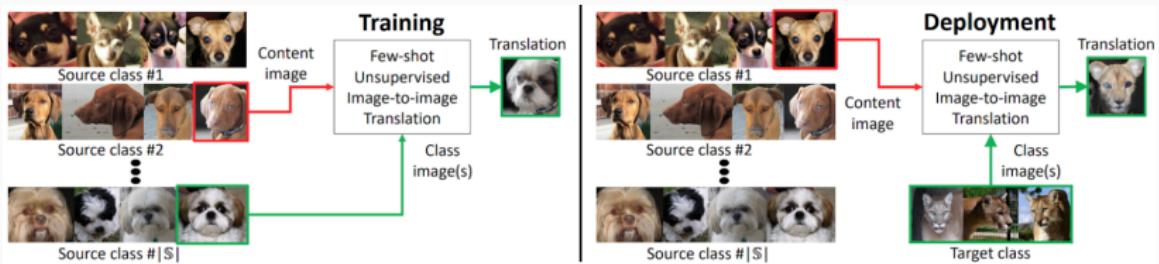
(a) AugCGAN



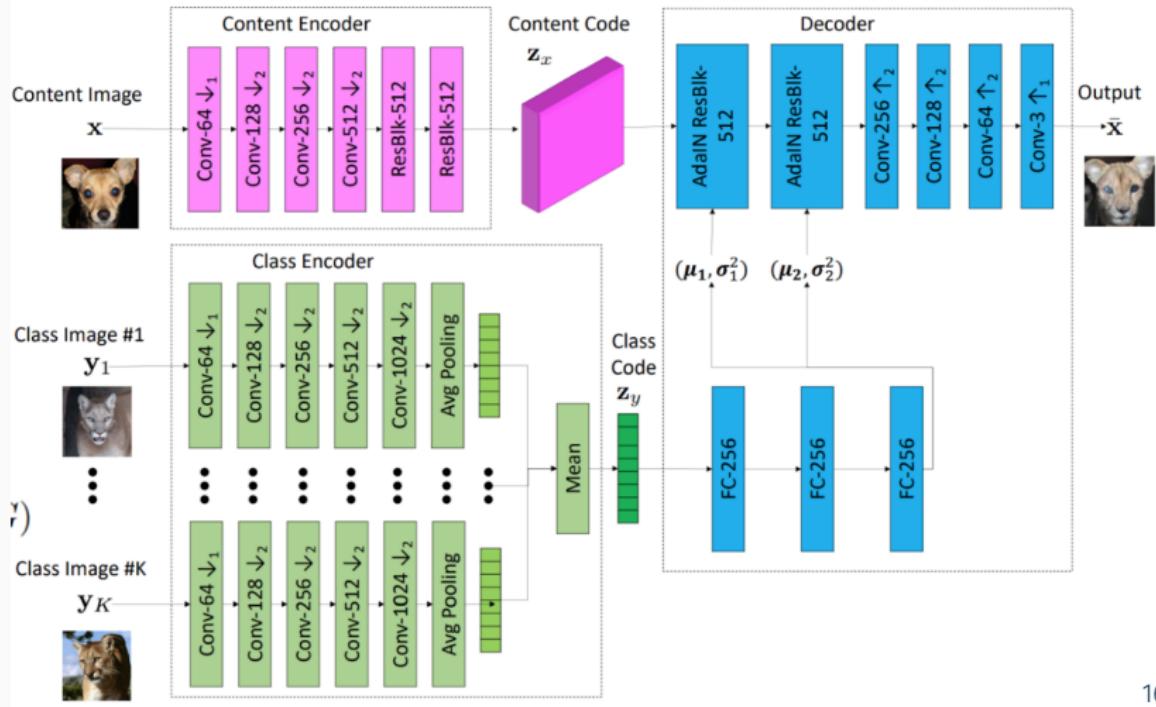
(b) StochCGAN

FUNIT

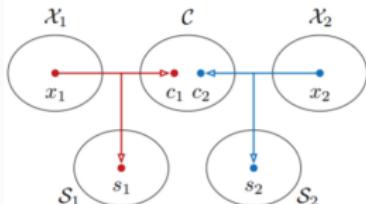
- Но всем этим моделям (кроме AdaIN) нужны большие датасеты в каждом стиле
- FUNIT (Liu et al., 2019) может делать трансфер между многими стилями, определяемыми несколькими картинками



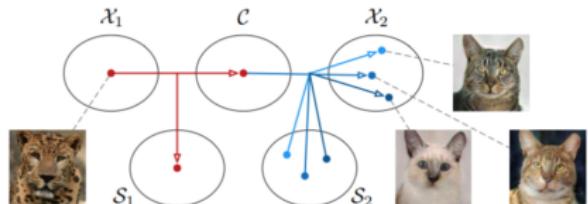
- Идея: условный генератор и multitask adversarial дискриминатор



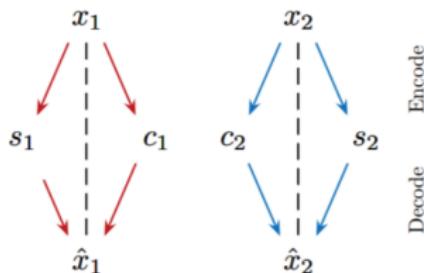
- Следующий шаг – MUNIT (Huang et al., 2018)



(a) Auto-encoding



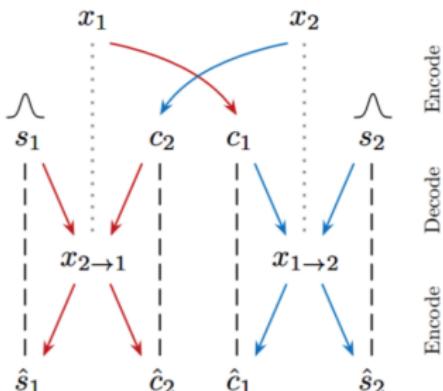
(b) Translation



Encode Decode

\mathcal{L}_1	domain 1	\mathcal{C}	content	x	images
loss	auto encoders	\mathcal{C}	features	x	
GAN	domain 2	\mathcal{S}	style		
loss	auto encoders	\mathcal{S}	features	\mathcal{N}	Gaussian prior

(a) Within-domain reconstruction



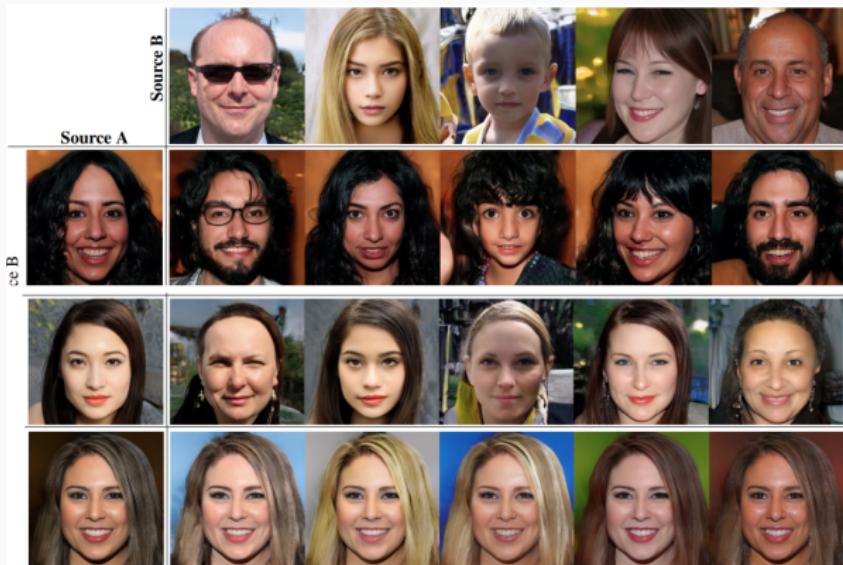
Encode Decode Encode Decode

(b) Cross-domain translation

STYLEGAN

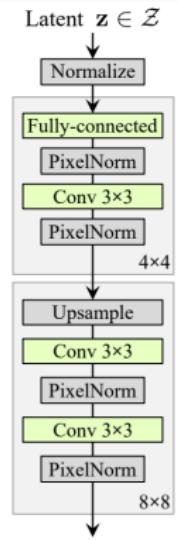
- StyleGAN от NVIDIA (Karras et al., 2019) порождает лица с условиями, взятыми из других лиц:

<https://www.youtube.com/watch?v=kSLJria0umA>

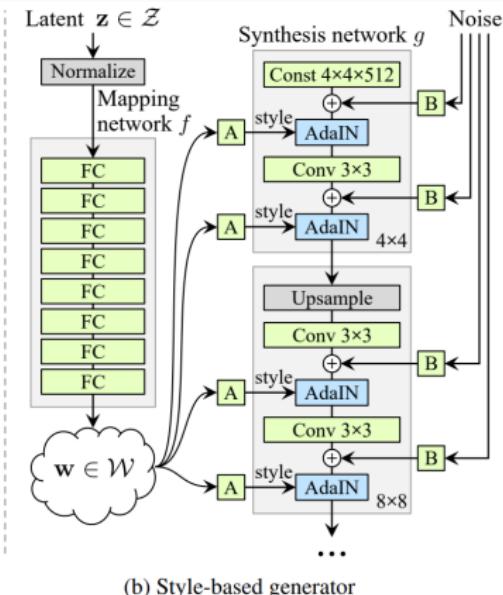


STYLEGAN

- Смысл в том, чтобы подавать стиль на вход на разных уровнях генератора, и использовать его в AdaIN:



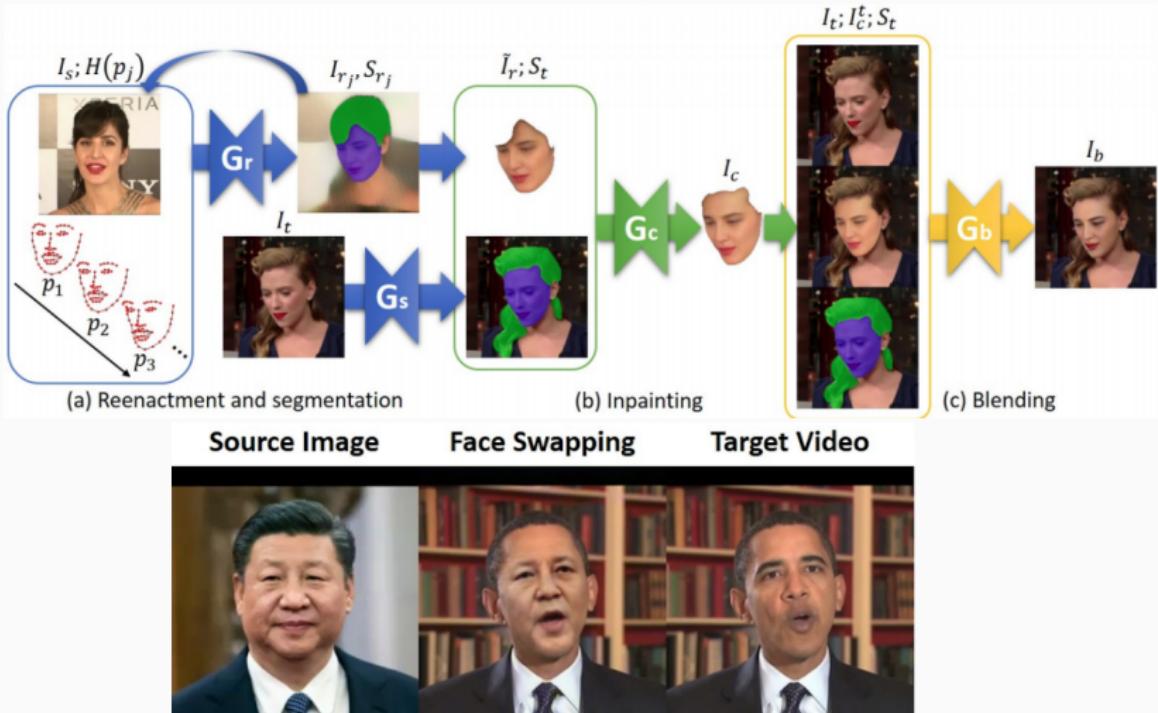
(a) Traditional



(b) Style-based generator

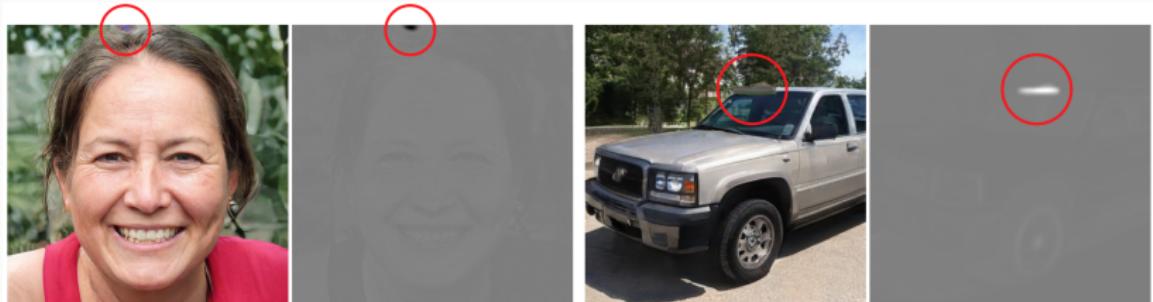
STYLEGAN

- Deepfakes тоже так работают: FSGAN (Nirkin et al., 2019)

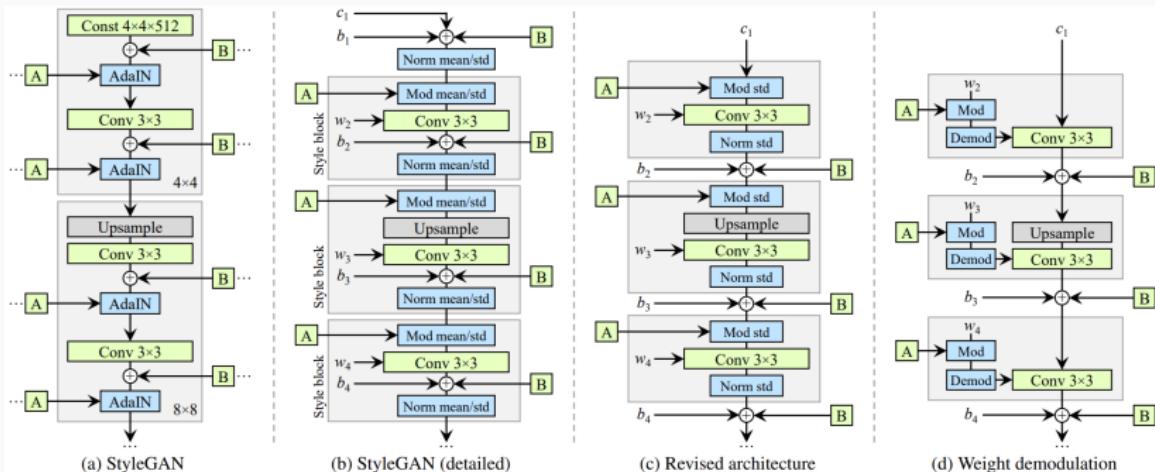


STYLEGAN

- StyleGAN2 (Karras et al., 2019) – NVIDIA смотрела на то, как дальше улучшать качество картинок, получающихся из StyleGAN
- Неочевидные проблемы – например, вот этот артефакт происходит от instance normalization:



- Поэтому они переделали архитектуры сетей в StyleGAN2:

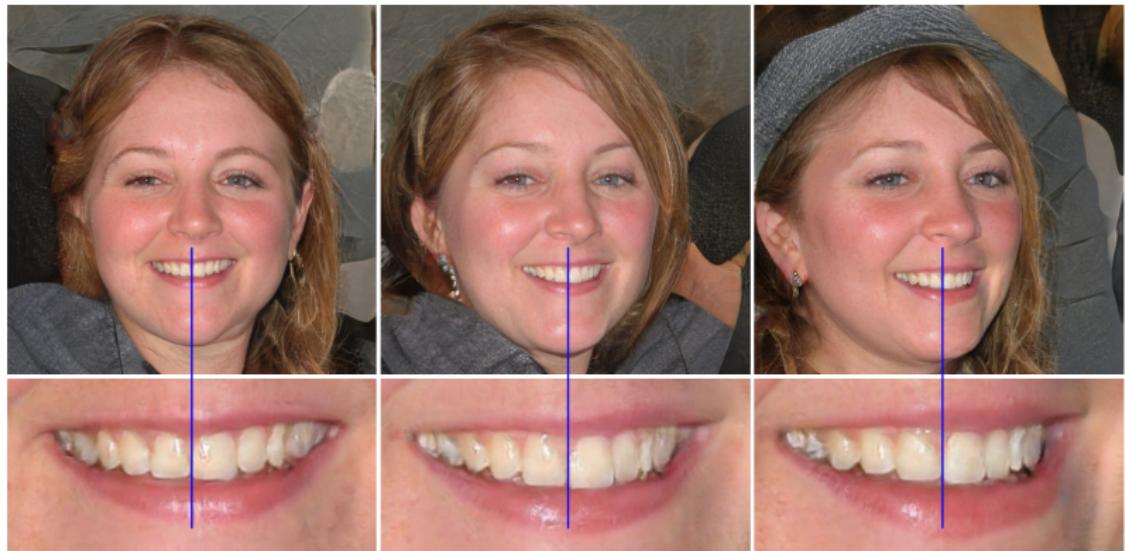


STYLEGAN

- Заменяют нормализацию на демодуляцию, т.е. чуть другое преобразование:

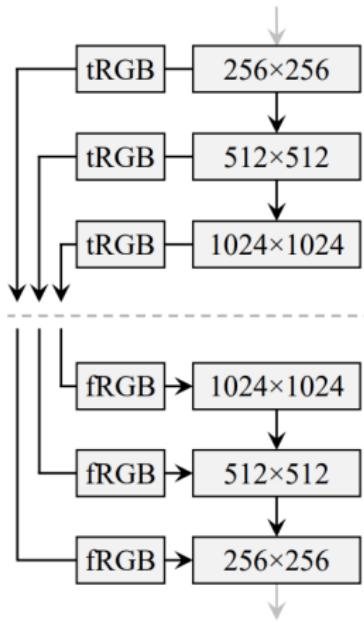


- Есть артефакты и от progressive growing:

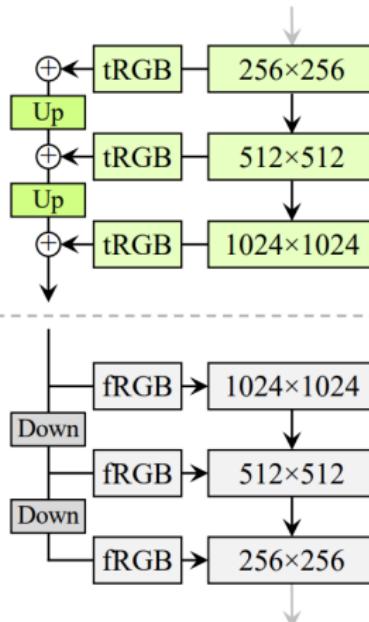


STYLEGAN

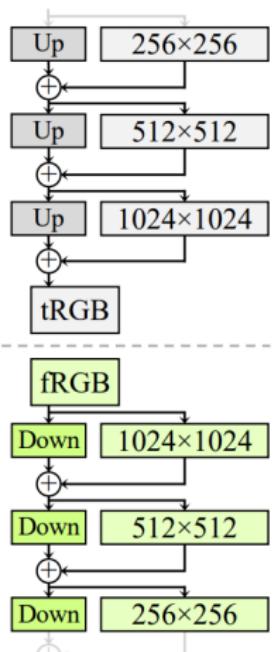
- Поэтому вместо этого сравнивают архитектуры, которые добиваются того же эффекта внутри сети:



(a) MSG-GAN

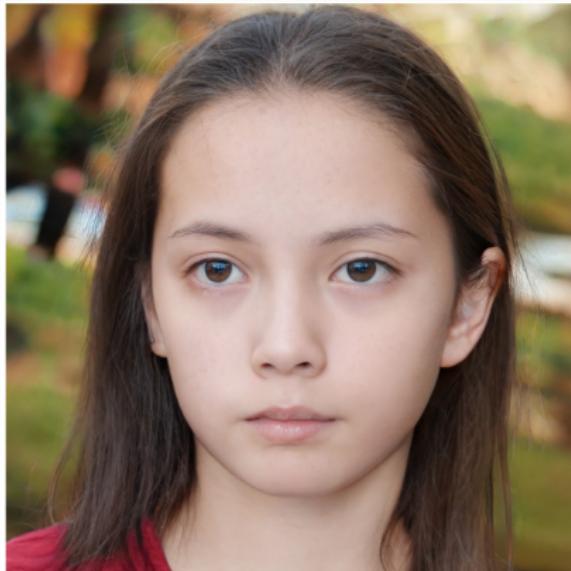


(b) Input/output skips



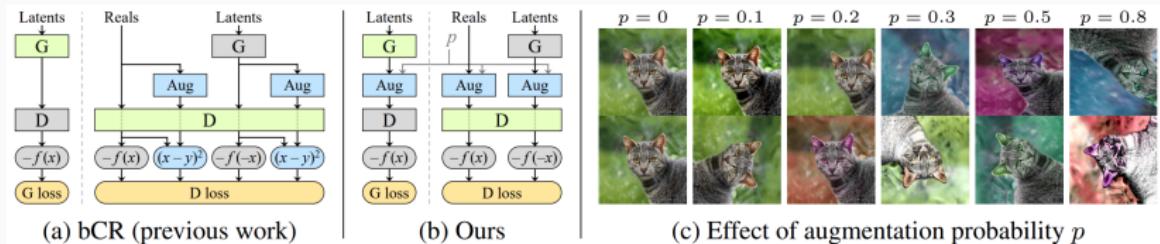
(c) Residual nets¹⁷

- Ну и, конечно, ещё лучше получается:



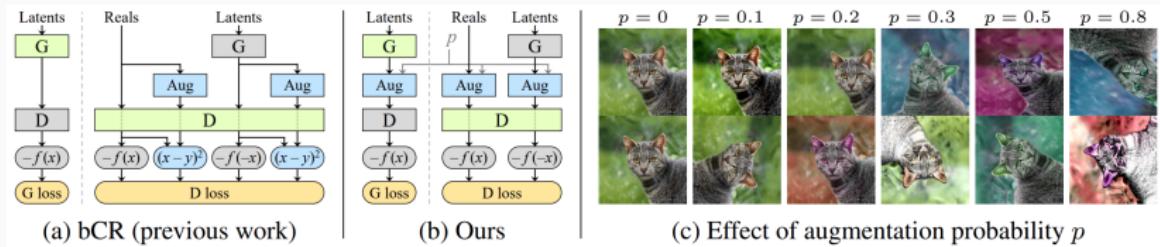
STYLEGAN

- А последняя от них работа, (Karras et al., Oct. 2020), обращается к тому, что делать, когда данных мало
- Тут очень важны аугментации
- Раньше аугментации применялись к тому, что видит D , плюс регуляризатор на то, чтобы они не «протекали», т.е. чтобы дискриминатор не обращал внимания на аугментации



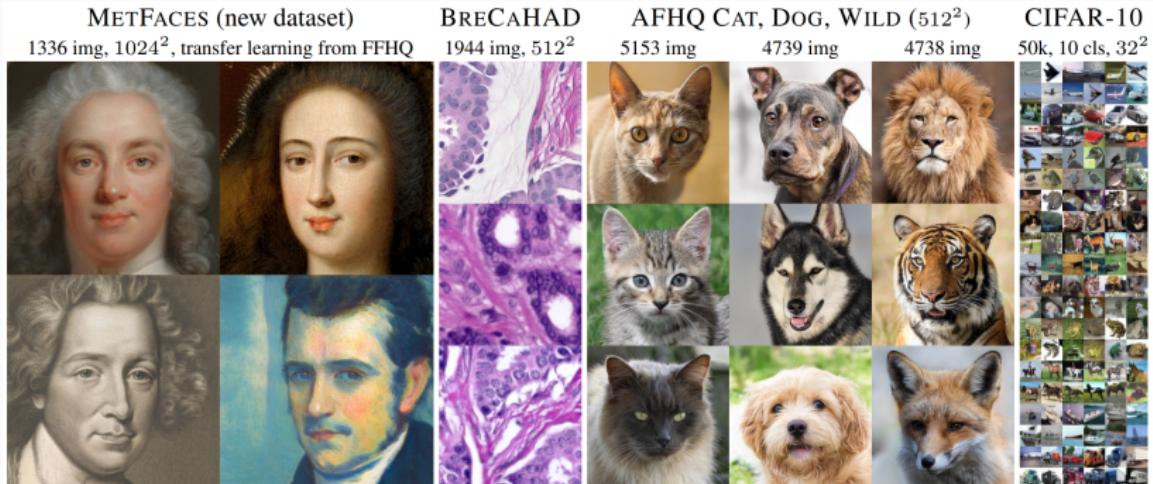
STYLEGAN

- Но тогда аугментации могут «протекать» всё равно, ведь теперь для D не важно, была аугментация или нет
- Karras et al. применяют аугментации просто стохастически
- Вторая новизна – адаптивная аугментация, т.е. p настраивается динамически в зависимости от того, насколько сильный оверфиттинг мы видим



STYLEGAN

- Получается отлично, особенно при transfer learning, т.е. если начать с конфигурации, обученной на другом датасете:



STYLEGAN

- Вот так делал StyleGAN2:



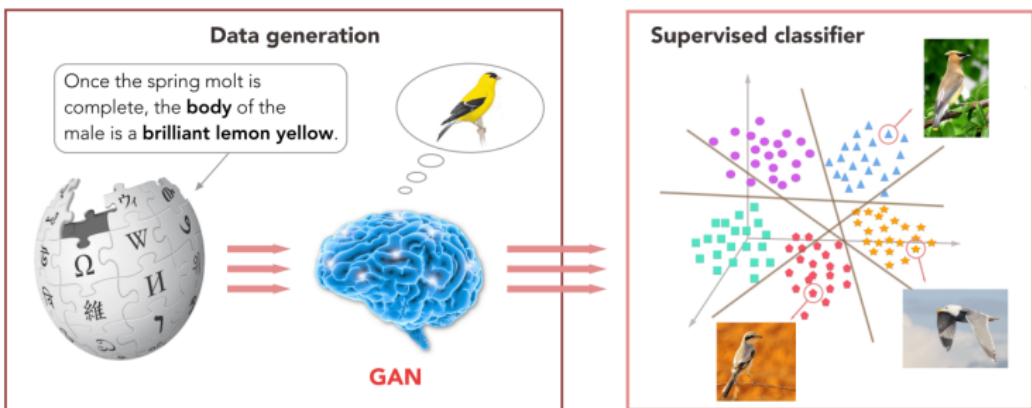
STYLEGAN

- А вот так новый StyleGAN Ada:



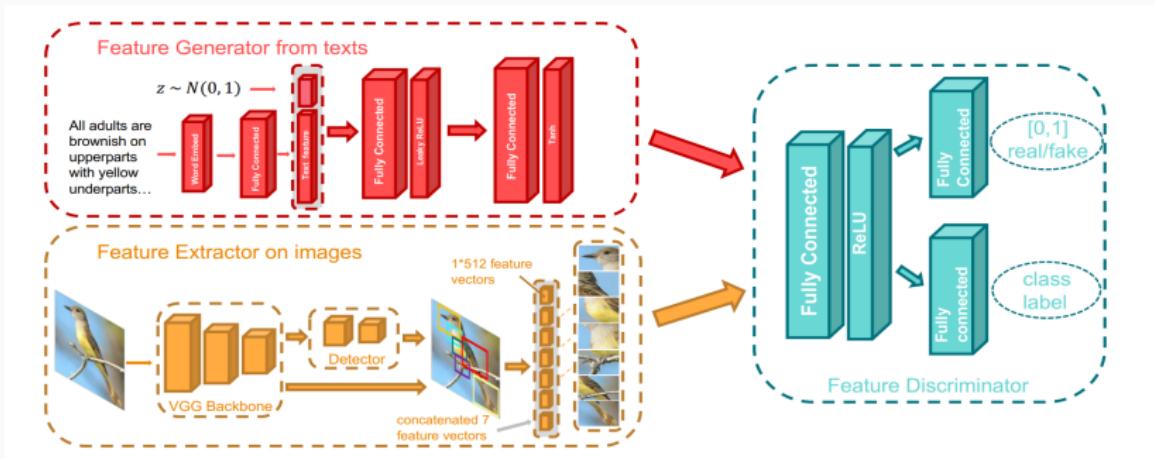
ZERO-SHOT LEARNING

- (Zhu et al., CVPR 2018): A Generative Adversarial Approach for Zero-Shot Learning from Noisy Texts
- Можно делать вообще безумные вещи – обучать классификатор картинок на текстах из википедии:



ZERO-SHOT LEARNING

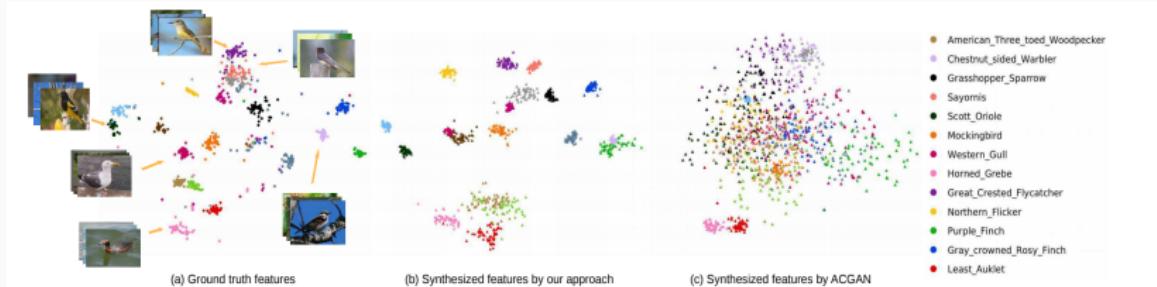
- Логичная архитектура:



- Тут важно, что для классификатора не надо порождать картинки, можно просто признаки порождать.

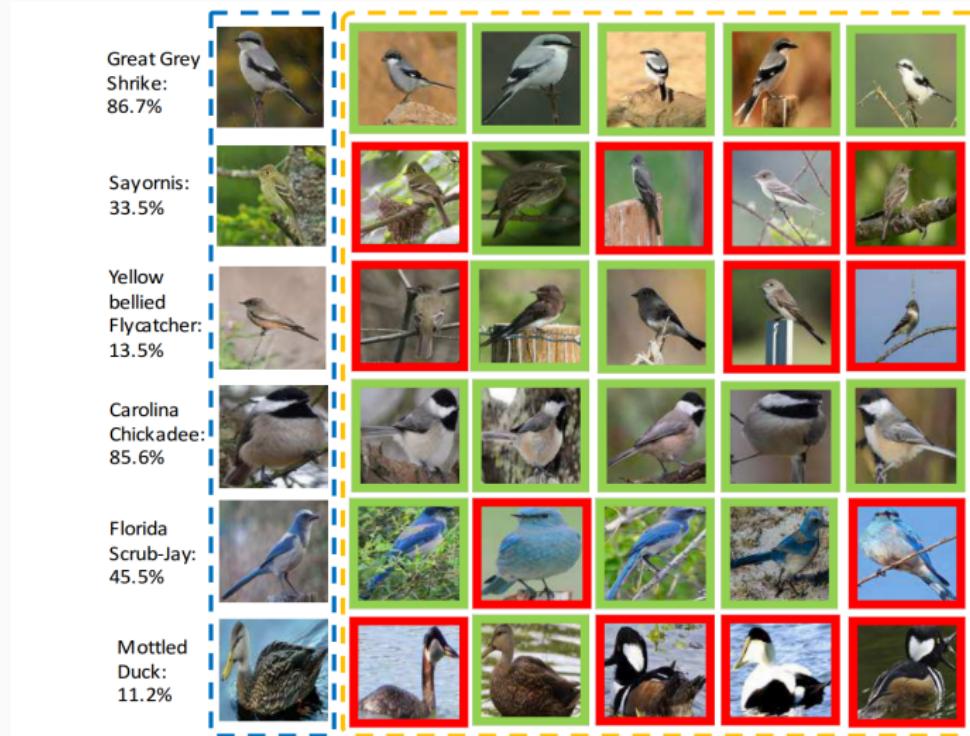
ZERO-SHOT LEARNING

- И ведь работает:



ZERO-SHOT LEARNING

- Можно делать zero-shot retrieval: искать картинки птичек, которых модель никогда не видела



Спасибо!

Спасибо за внимание!

