

# Порождающие состязательные сети II

---

Сергей Николенко

Академия MADE – Mail.Ru

02 декабря 2020 г.

---

*Random facts:*

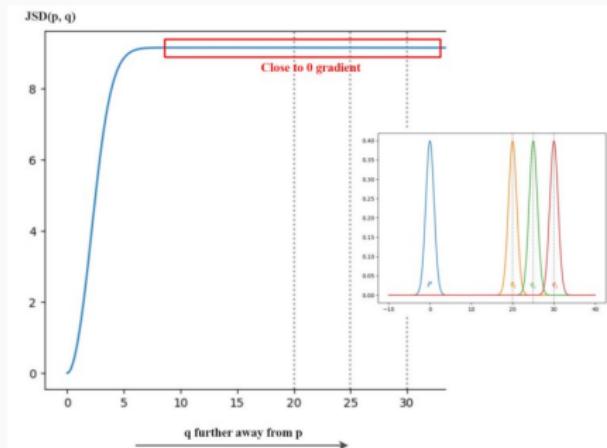
- 2 декабря — Международный день борьбы за отмену рабства, а в России — День банковского работника
- 2 декабря 1804 г. Наполеон I Бонапарт провозгласил себя императором Франции, 2 декабря 1805 г. он нанёс поражение русским и австрийским войскам при Аустерлице, а 2 декабря 1852 г. императором французов провозгласил себя его племянник Луи-Наполеон Бонапарт
- 2 декабря 1942 г. Энрико Ферми в рамках манхэттенского проекта получил первую в мире контролируемую цепную ядерную реакцию
- 2 декабря 1956 г. яхта *Granma* застряла на мели в мангровом болоте, но экспедиционеры успешно высадились на берег и добрались до гор Сьерра-Маэсты
- 2 декабря 1971 г. станция «Марс-3» впервые в мире совершила мягкую посадку на поверхность Марса

## Функции ошибки в GAN

---

# LSGAN

- (Mao et al., 2016): Least Squares GAN (LSGAN)
- Проблема с обычными GAN'ами: функция ошибки (дивергенция Йенсена-Шеннона) насыщается, когда распределение генератора далеко от правильного, и ничего не обучается.



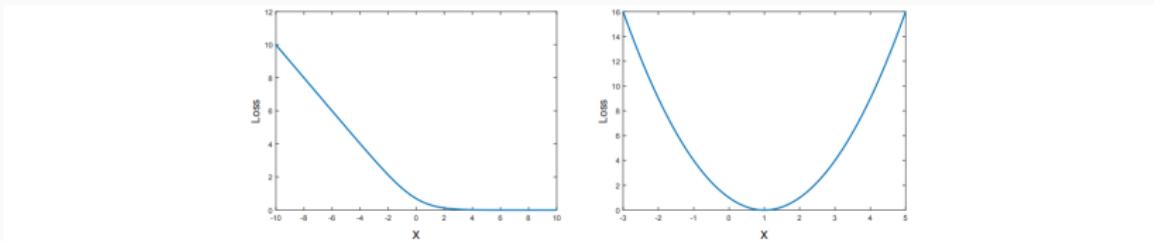
- Это потому, что дискриминатор в обычном GAN работает как классификатор с логистическим сигмоидом.

# LSGAN

- Давайте попробуем перейти от сигмоидальной к квадратичной функции ошибки:

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

т.е. дискриминатор учится отвечать  $a$  для фейков и  $b$  для настоящих данных, а генератор хочет убедить его отвечать  $c$  на фейках. Обычно берут просто  $a = 0$ ,  $b = c = 1$ .



# LSGAN

- Mao et al. показали, что LSGAN устойчивее к изменениям архитектуры, меньше страдает от mode collapse; в целом, сейчас квадратичная функция ошибки применяется часто.



(a) LSGANs.



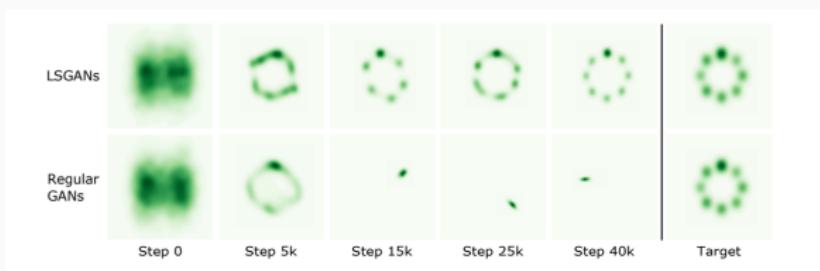
(b) Regular GANs.



(c) LSGANs.



(d) Regular GANs.



- (Chen et al., 2016): InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- Важное дополнение: было бы круто добавить disentanglement, чтобы разные признаки имели смысл
- В обычных GAN'ах используют просто вектор шума  $z$ . В InfoGAN мы его разбиваем на части:
  - $z$  – это действительно несжимаемый шум, источник энтропии;
  - $c = c_1c_2 \dots c_L$  – это латентный код.
- Можно в обычный GAN попробовать это добавить, чтобы генератор стал  $G(z, c)$ , но надо что-то добавить, чтобы  $c$  было важно и  $G$  не мог бы просто забыть на него.

# InfoGAN

- InfoGAN: будем требовать, чтобы была большая взаимная информация (mutual information)  $I(c; G(z, c))$ , т.е. в распределении  $G(z, c)$  должно оставаться много от  $c$ .
- Формально – вариационная оценка:

$$\begin{aligned} I(c; G(z, c)) &= H(c) - H(c|G(z, c)) \\ &= \mathbb{E}_{x \sim G(z, c)} [\mathbb{E}_{c' \sim P(c|x)} [\log P(c'|x)]] + H(c) \\ &= \mathbb{E}_{x \sim G(z, c)} [\underbrace{D_{\text{KL}}(P(\cdot|x) \parallel Q(\cdot|x))}_{\geq 0} + \mathbb{E}_{c' \sim P(c|x)} [\log Q(c'|x)]] + H(c) \\ &\geq \mathbb{E}_{x \sim G(z, c)} [\mathbb{E}_{c' \sim P(c|x)} [\log Q(c'|x)]] + H(c) \end{aligned}$$

- И эту нижнюю оценку мы через reparametrization trick параметризуем нейросетью  $Q$ ; она обычно почти целиком совпадает с  $D$ .

# InfoGAN

- Пример на MNIST с  $c = c_1c_2c_3$ , где  $c_1$  – дискретная метка с 10 категориями (без supervision! с равномерным априорным распределением), и  $c_2, c_3 \sim U[-1, 1]$ :

--	--

(a) Varying  $c_1$  on InfoGAN (Digit type)

(b) Varying  $c_1$  on regular GAN (No clear meaning)

--	--

(c) Varying  $c_2$  from  $-2$  to  $2$  on InfoGAN (Rotation)

(d) Varying  $c_3$  from  $-2$  to  $2$  on InfoGAN (Width)

- На самом деле по коду  $c_1$  классификатор даёт ошибку 5% на MNIST без всякого supervision.

# InfoGAN

- 3D-лица:



(a) Azimuth (pose)

(b) Elevation



(c) Lighting

(d) Wide or Narrow

InfoGAN

- 3D-стулья:



# Wassershtein GAN

- Wassershtein GAN (Arjovsky et al., 2017):
  - что такое обучить распределение вероятностей?
  - это значит минимизировать  $KL(p_{\text{data}} \parallel p_{\text{model}})$ ;
  - но для этого нужно, чтобы  $p_{\text{model}}$  существовала, а это неверно, если распределение сосредоточено на подмногообразиях малой размерности; вряд ли многообразие  $p_{\text{model}}$  будет существенно пересекаться с многообразием  $p_{\text{data}}$ , и  $KL$  будет не определено (бесконечно);
  - обычно мы добавляем шум (гауссовский, например), все модели в ML с шумом;
  - но для порождающих моделей шум мешает, делает порождённые примеры размытыми;
  - поэтому обычно шум используют для подсчёта ML, но убирают во время порождения, т.е. шум – это неправильно, но надо для ML.

# Wassershtein GAN

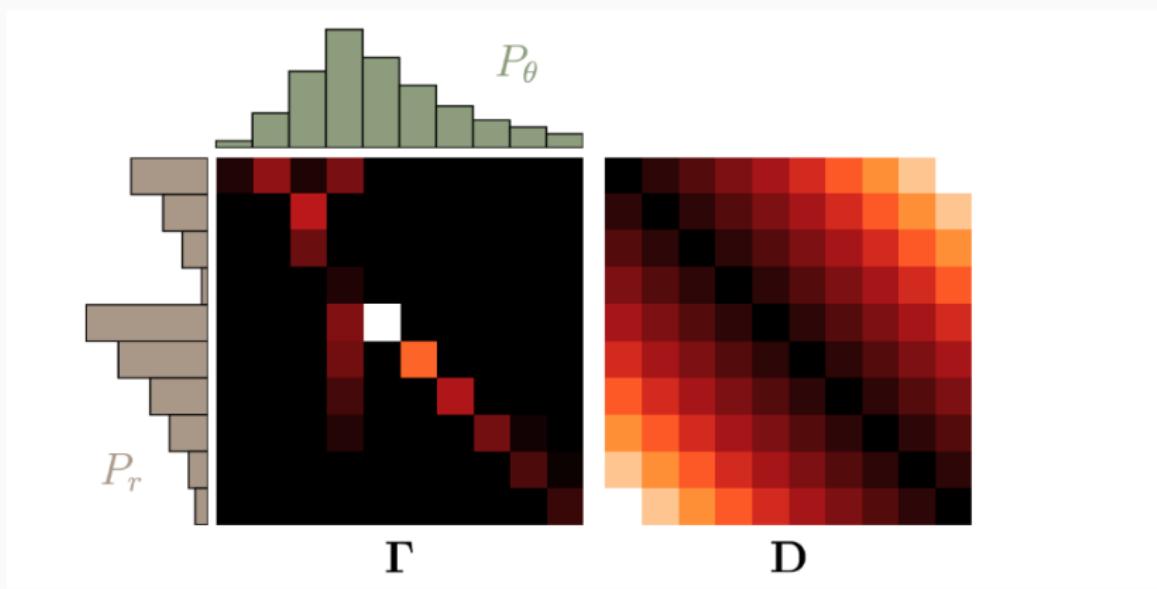
- Wassershtein GAN (Arjovsky et al., 2017): давайте посмотрим на другие метрики близости между  $p_{\text{data}}$  и  $p_{\text{model}}$ .
- Earth Mover distance, или Wassershtein-1:

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

где  $\Pi(p_{\text{data}}, p_{\text{model}})$  – множество совместных распределений  $\gamma(x, y)$ , маргиналы которых –  $p_{\text{data}}$  и  $p_{\text{model}}$ .

# Wassershtein GAN

- Т.е.  $\gamma(x, y)$  показывает, сколько надо «массы» перераспределить от  $x$  к  $y$ , чтобы превратить  $p_{\text{data}}$  в  $p_{\text{model}}$ .



# Wassershtein GAN

- Пример: рассмотрим  $Z \sim U[0, 1]$ ,  $\mathbb{P}_0$  – распределение  $(0, Z)$  на  $\mathbb{R}^2$ , и  $g_\theta(z) = (\theta, z)$ , где  $\theta$  – параметр. Тогда

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|,$

- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0, \end{cases}$

- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0, \end{cases}$

- and  $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0. \end{cases}$

- Т.е. только EM-расстояние действительно куда-то сходится при  $\theta \rightarrow 0$ , а это ведь очень похоже на обучение распределения, сосредоточенного на подмногообразии.

# Wassershtein GAN

- Можно доказать, что  $W(p_{\text{data}}, p_{\text{model}})$  – непрерывная функция от параметров  $\theta$  распределения  $p_{\text{model}}$  при достаточно слабых условиях.
- А двойственность Монжа-Канторовича (Kantorovich-Rubinstein duality) говорит, что инфимум

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

эквивалентен

$$W(p_{\text{data}}, p_{\text{model}}) = \sup_{\|f\|_L \leq 1} (\mathbb{E}_{x \sim p_{\text{data}}} [f(x)] - \mathbb{E}_{x \sim p_{\text{model}}} [f(x)]),$$

где супремум берётся по всем функциям с липшицевой константой  $\leq 1$ .

- А мы хотим обучить порождающую модель  $p_{\text{model}} = g_{\theta}(z)$ .
- Давайте всё параметризуем нейросетями.

# Wassershtein GAN

- Сеть  $f_w$  для функции  $f$  и сеть для  $g_\theta$ . Тогда:
  - для данного  $g_\theta$  обучим веса  $f_w$ , максимизируя

$$\mathbb{E}_{x \sim p_{\text{data}}} [f(x)] - \mathbb{E}_{x \sim p_{\text{model}}} [f(x)];$$

- для данного  $f_w$  подсчитаем

$$\begin{aligned}\nabla_\theta W(p_{\text{data}}, p_{\text{model}}) &= \nabla_\theta (\mathbb{E}_{x \sim p_{\text{data}}} [f(x)] - \mathbb{E}_{x \sim p_{\text{model}}} [f(x)]) = \\ &= -\mathbb{E}_{z \sim z} [\nabla_\theta f_w(g_\theta(z))].\end{aligned}$$

- А чтобы  $f_w$  оставалось липшицевым, можно просто weight clipping сделать для градиентов.

# Wassershtein GAN

- Итого алгоритм обучения:

---

**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:** :  $\alpha$ , the learning rate.  $c$ , the clipping parameter.  $m$ , the batch size.  
 $n_{\text{critic}}$ , the number of iterations of the critic per generator iteration.

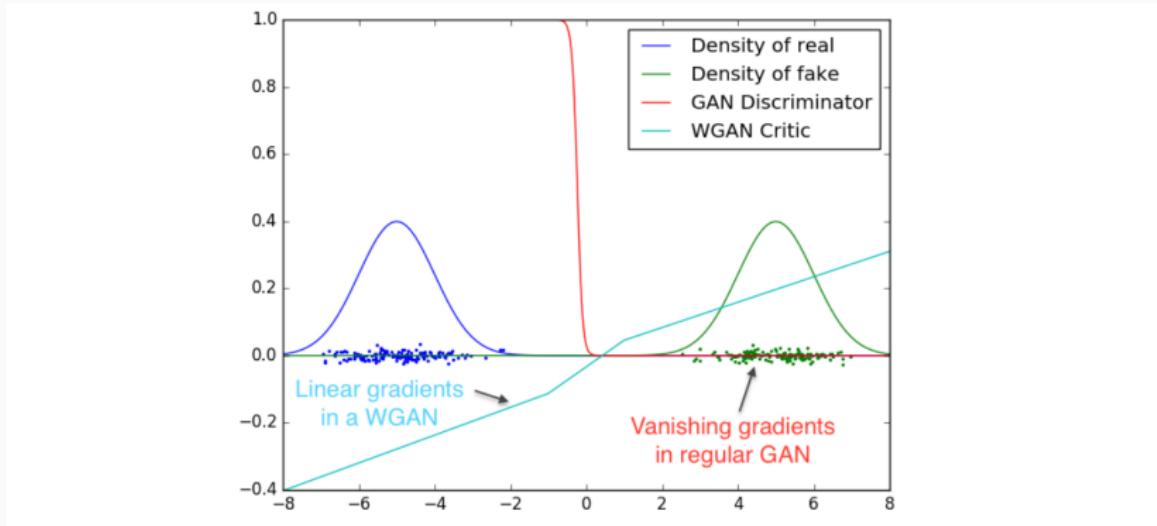
**Require:** :  $w_0$ , initial critic parameters.  $\theta_0$ , initial generator's parameters.

```
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:      $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$ 
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:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$ 
12: end while
```

---

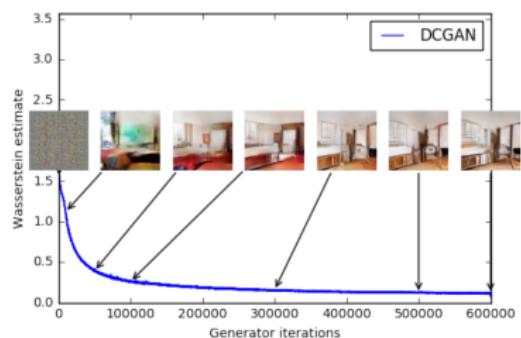
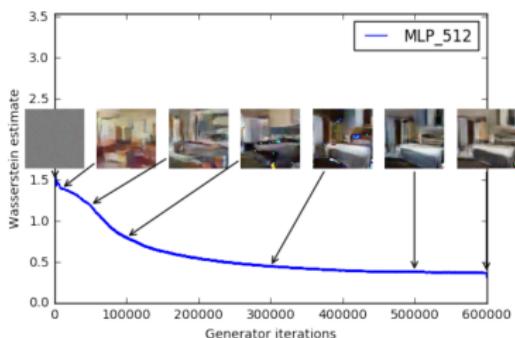
# Wassershtein GAN

- Пример, на котором видно, что в WGAN можно спокойно обучать  $f_w$  до сходимости, а в обычном GAN дискриминатор, может, и не стоит так обучать:



# Wassershtein GAN

- Качество тесно коррелирует с функцией ошибки; слева тут просто полносвязная сеть в качестве генератора, справа – DCGAN:



# Wassershtein GAN

- WGAN устойчивее к изменению архитектуры; вот обычные WGAN и DCGAN (снизу):



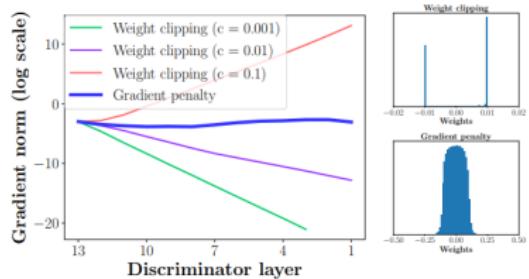
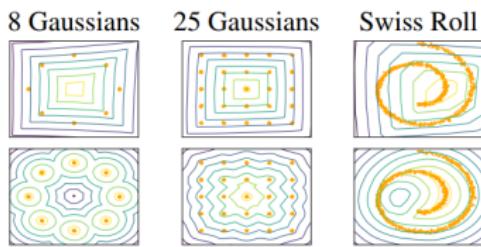
# Wassershtein GAN

- А вот что будет, если убрать из генератора batchnorm и сделать фильтров поменьше (одинаковое число на каждом уровне):



# Wassershtein GAN

- (Gulrajani et al., 2017): Improved Training of Wasserstein GANs
- Weight clipping в критике на самом деле ведёт к проблемам, не обучаются более высокие моменты распределений и градиенты взрываются/затухают
- Лучше делать мягкий регуляризатор на градиент, типа  $\lambda \mathbb{E}_{\hat{x} \sim P_{\hat{x}}} \left[ (\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2 \right]$



## EBGAN и BEGAN

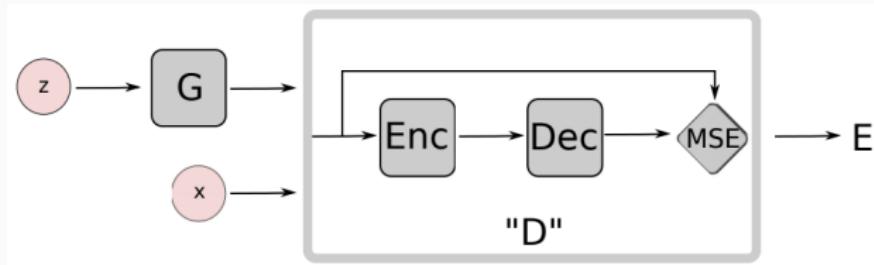
- (Zhao et al., 2016): Energy-Based Generative Adversarial Networks (EBGAN)
- Давайте рассмотрим дискриминатор как функцию энергии, которая присваивает низкие значения энергии областям вблизи распределения данных и высокие всем остальным.
- Генератор должен порождать контрастные сэмплы с минимальными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

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

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

## EBGAN и BEGAN

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



- 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:

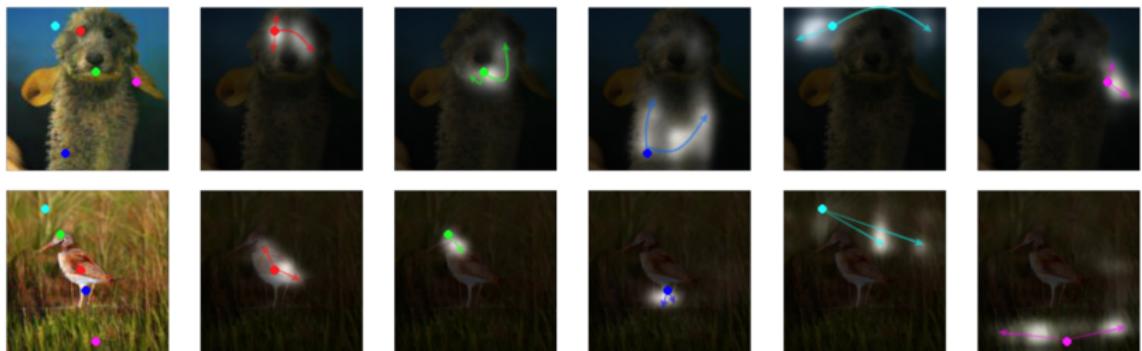


(a) EBGAN (64x64)

(b) Our results (128x128)

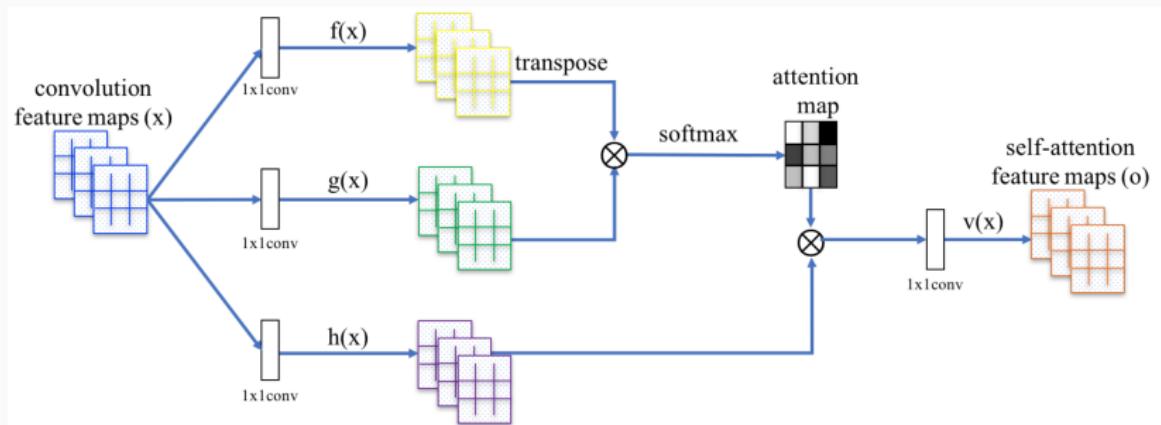
# SAGAN

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



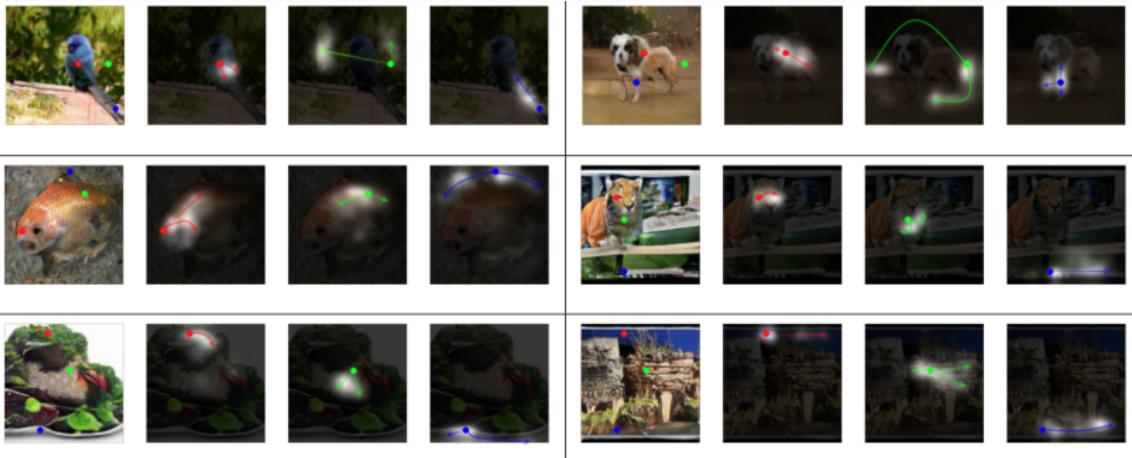
# SAGAN

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

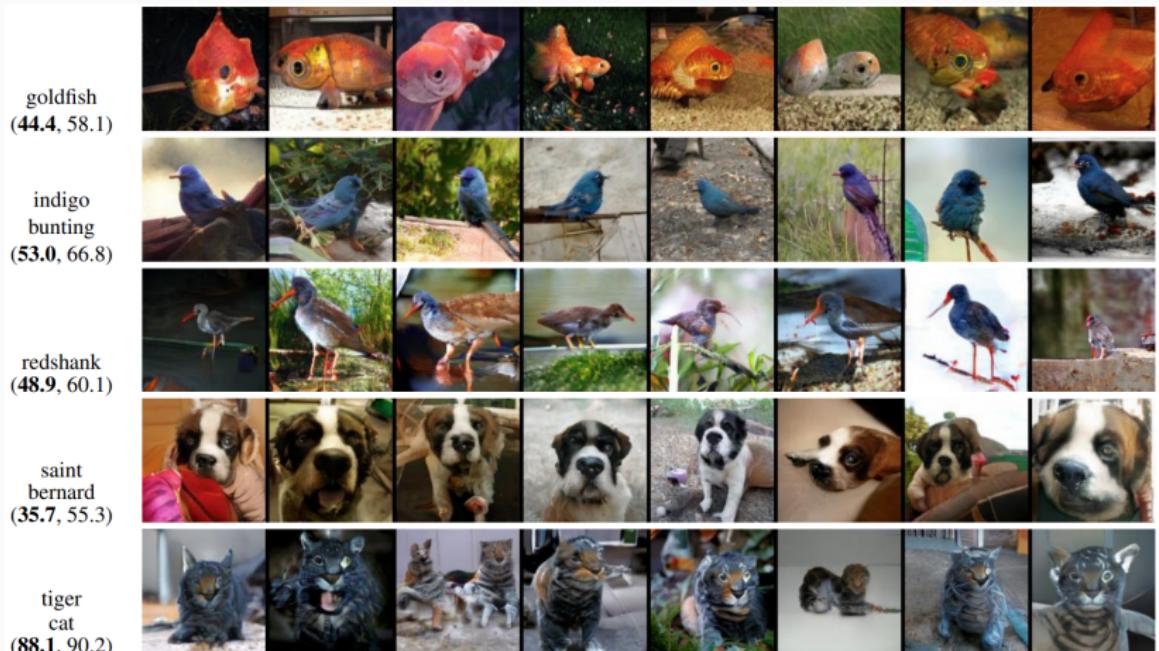


# SAGAN

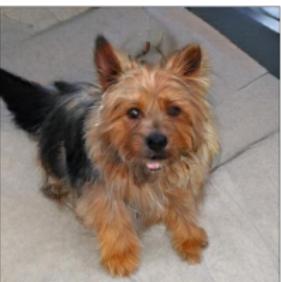
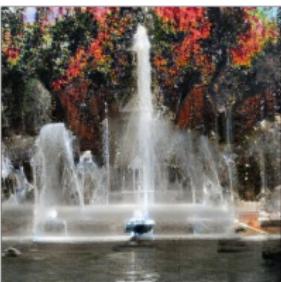
- И получается, что действительно генератор (здесь примеры с последнего слоя) может смотреть достаточно далеко, когда порождает картинку:



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

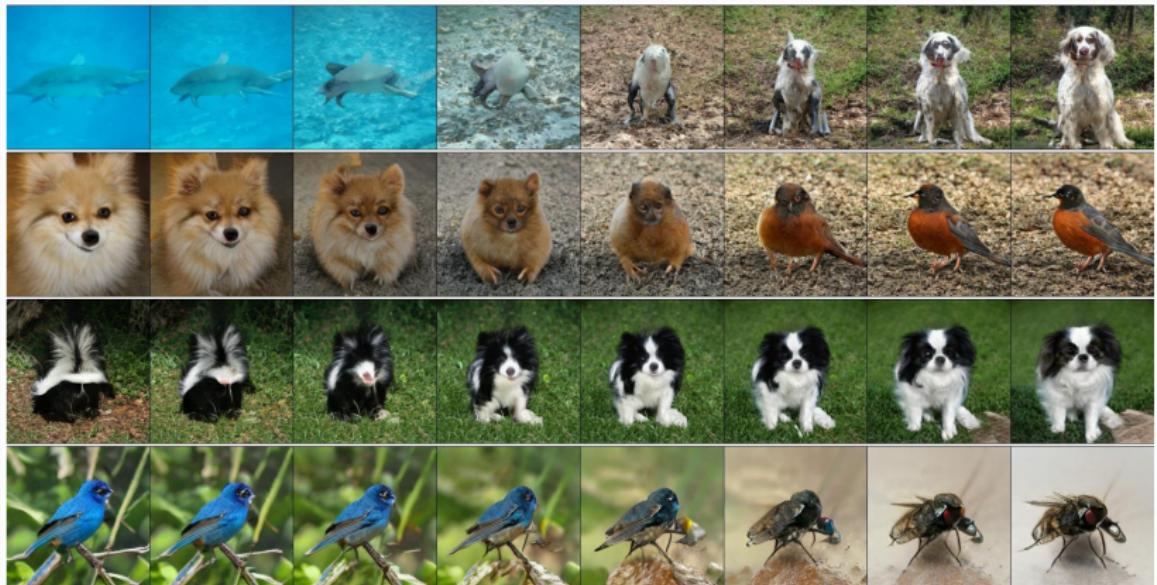


- Следующий этап – BigGAN (Brock et al., 2019), улучшил Inception score в три раза...



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

- Интерполяции:



- Ой, а что такое «лучшее»?..

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

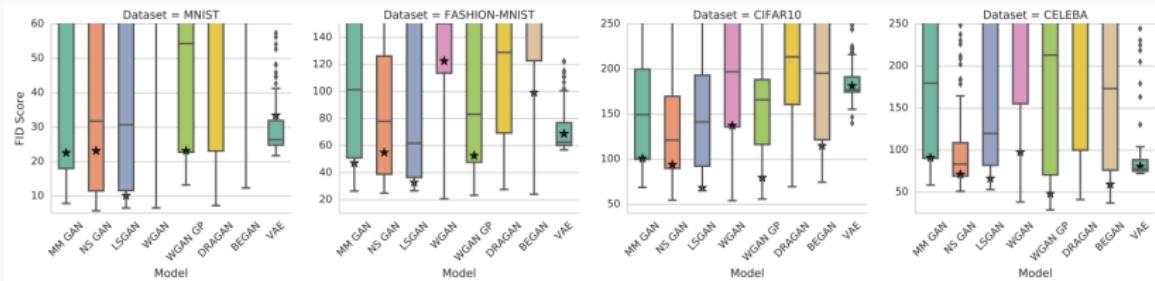
# Are GANs Created Equal?

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

# Are GANs Created Equal?

- Все GAN'ы очень чувствительны к гиперпараметрам:

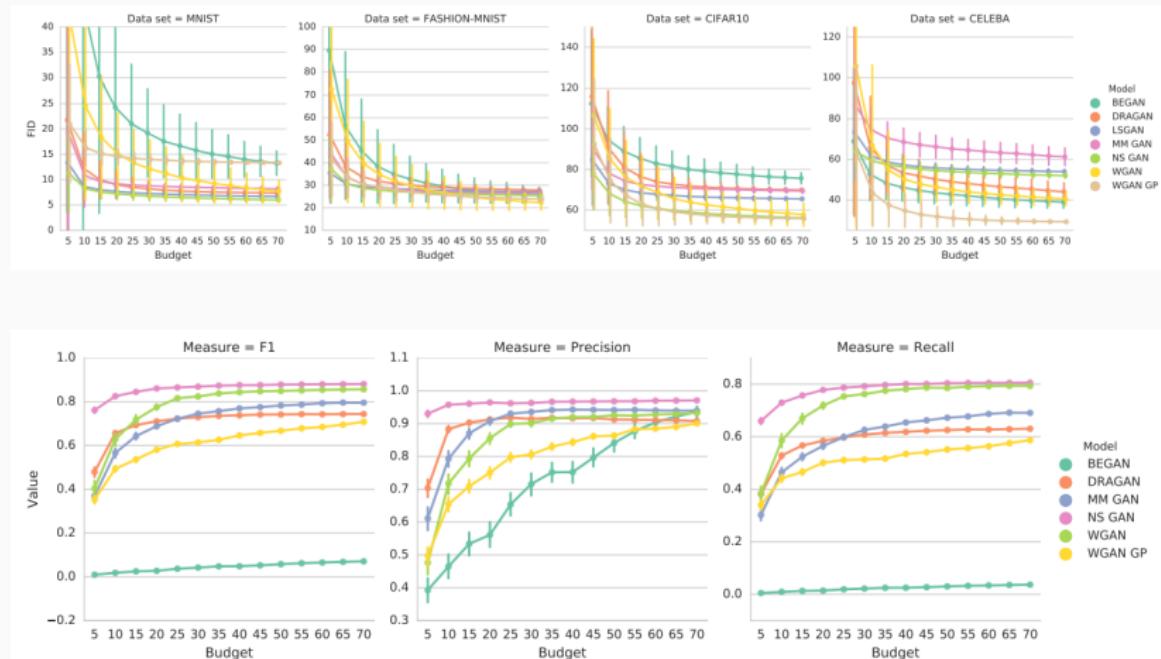


- Кстати, метрика здесь – Frechet Inception Distance (FID):  
считаем средние и матрицы ковариаций кодов настоящих и  
фейковых данных и считаем между ними расстояние Фреше

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

# Are GANs Created Equal?

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



# Что сейчас с GAN'ами делают

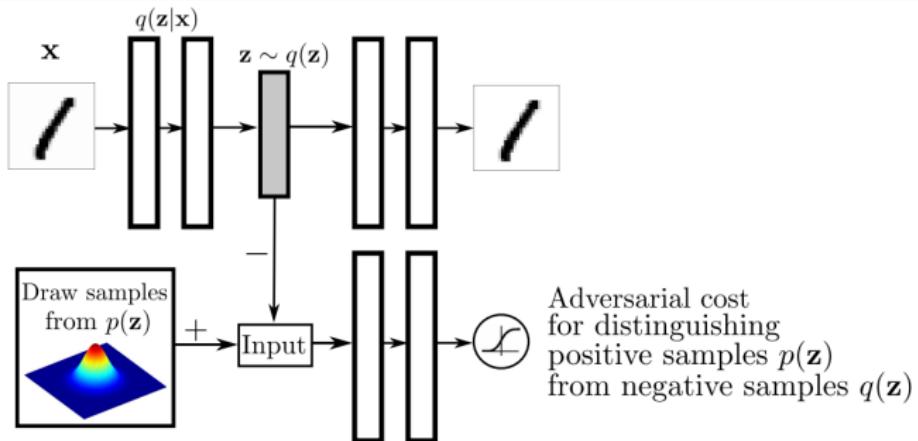
- Как видите, GAN'ы – очень интересная наука, много математики, настоящие теоремы надо доказывать...
- Но, конечно, тут же выяснилось, что часто достаточно просто хорошо придумать loss functions из общих соображений. Чем все и занимаются.



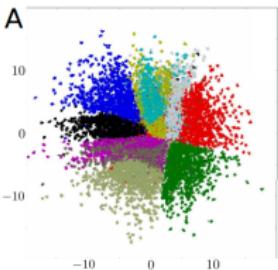
## Архитектуры, основанные на GAN

---

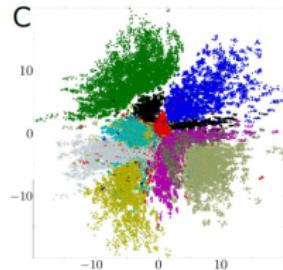
- Соперничающие автокодировщики (adversarial autoencoders; Makhzani et al., 2015): дискриминатор приводит распределение признаков (на скрытом слое) к заданному  $p(z)$ .



## Adversarial Autoencoder



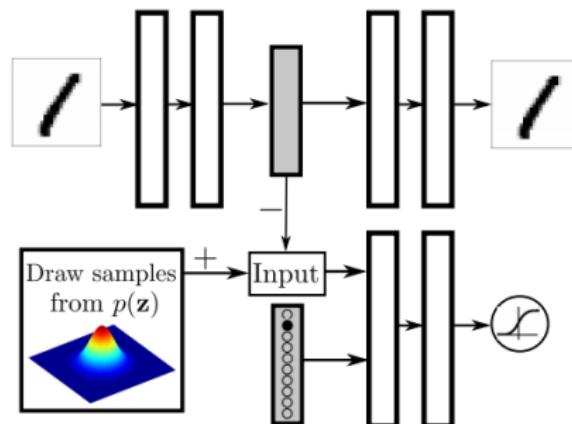
## Variational Autoencoder

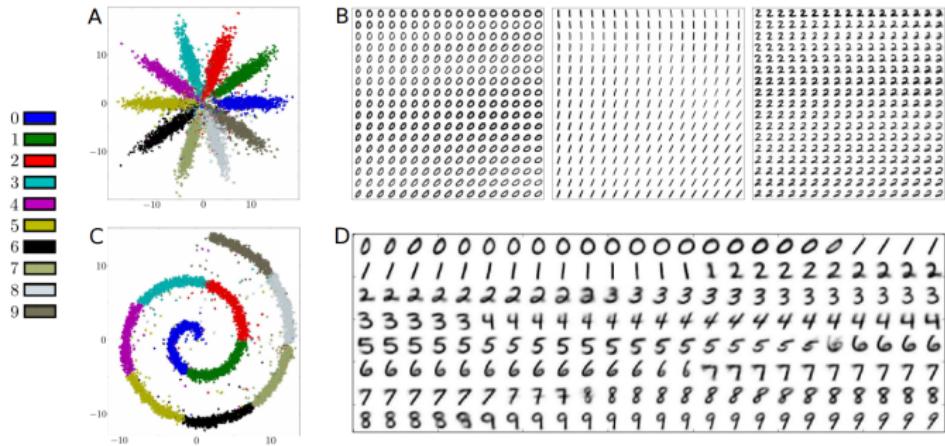


## Manifold of Adversarial Autoencoder

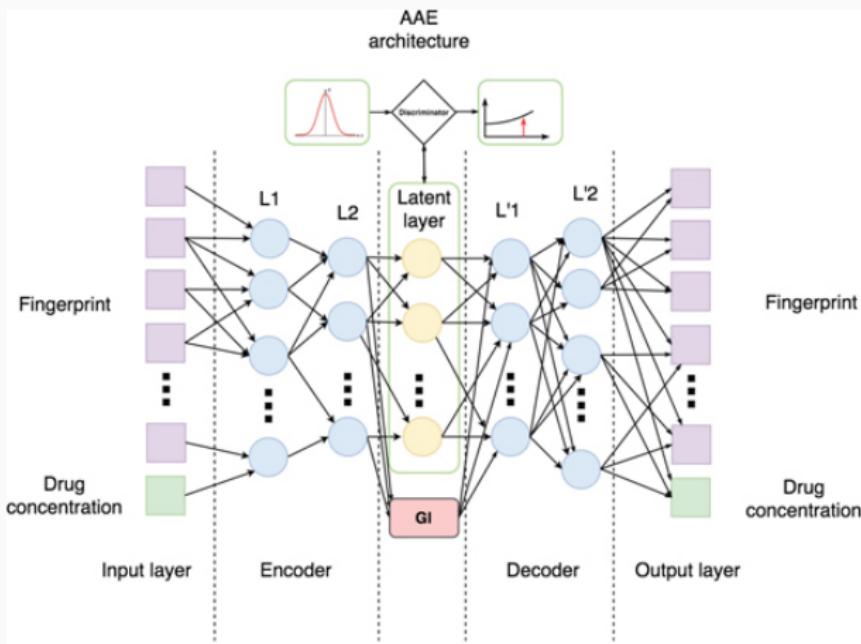
	0		5
	1		6
	2		7
	3		8
	4		9

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

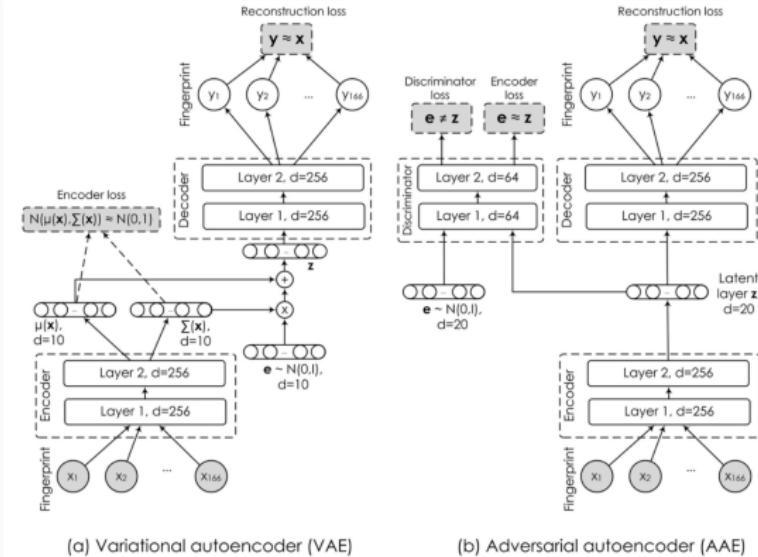




- (Kadurin et al., 2017) – ААЕ для порождения молекул в онкологии:



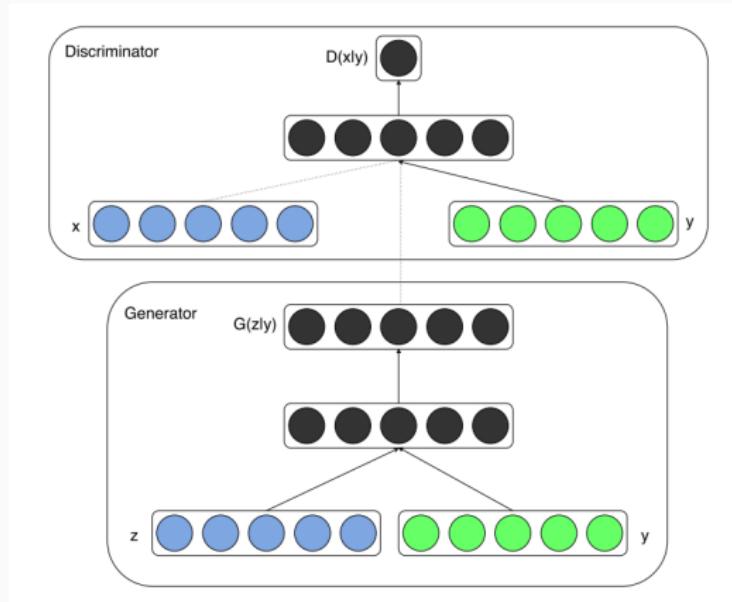
- (Kadurin et al., 2018) – druGAN для порождения молекул; сравнили с VAE:



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

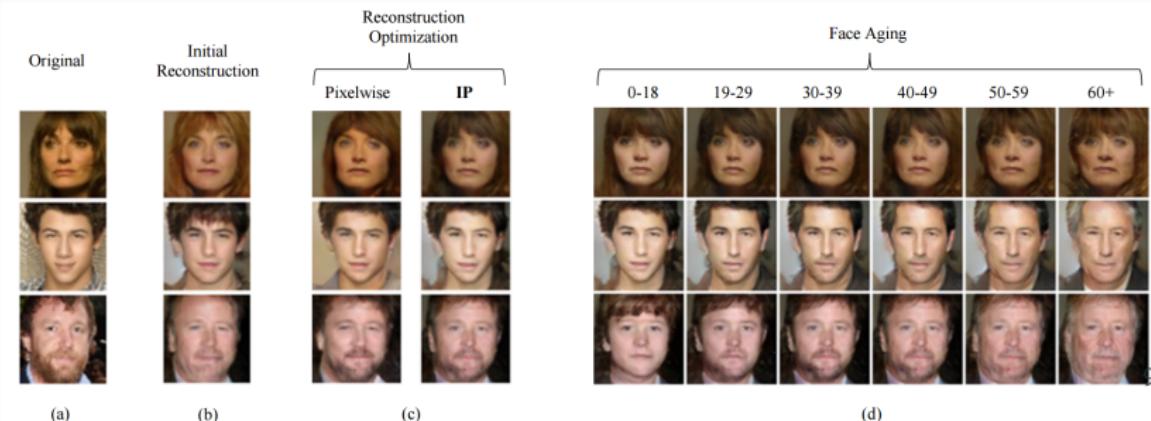
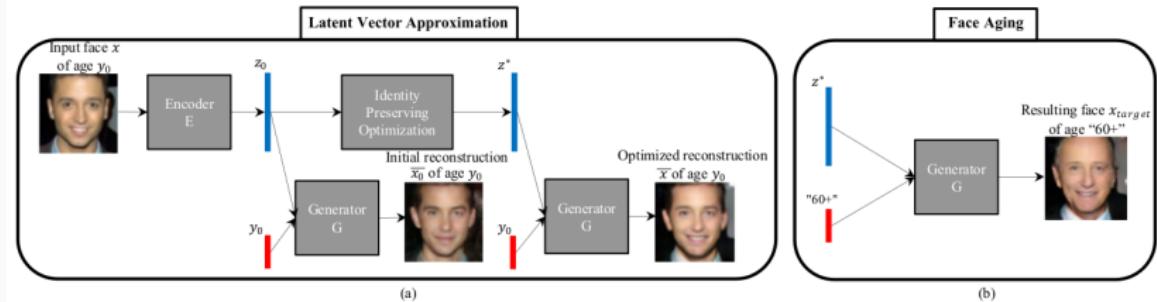
# Условные GAN'ы

- 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  $\hat{h}_i$  при условии  $h_{i+1}$ , и чтобы генератор не игнорировал условие, надо, чтобы восстанавливались  $h_{i+1}$  через соответствующий encoder:

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

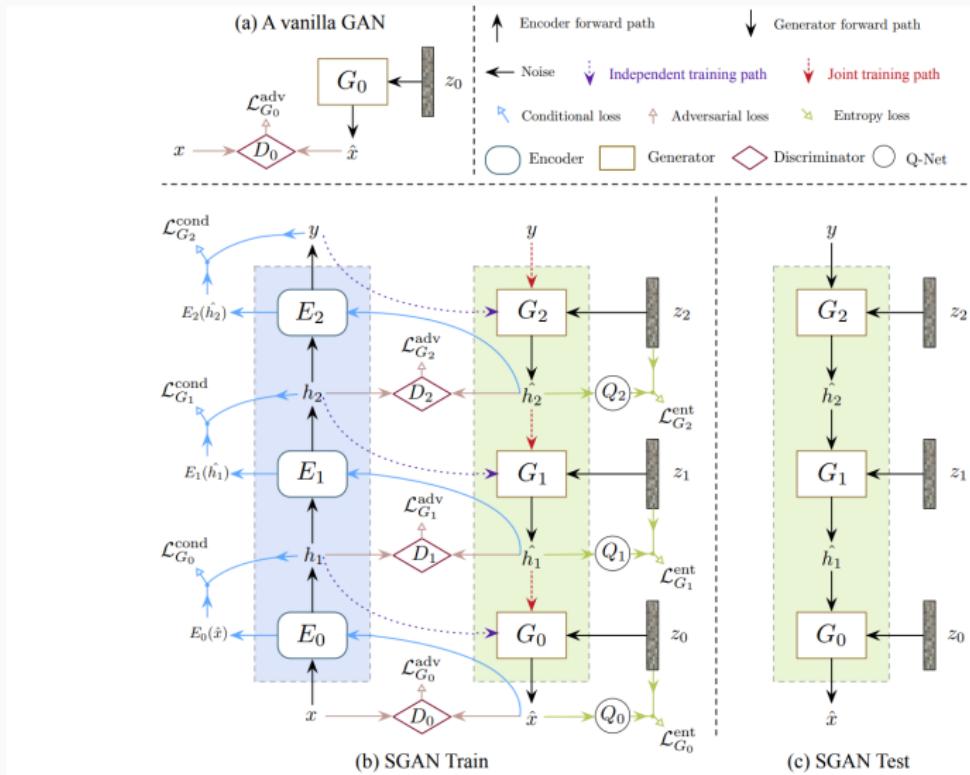
- entropy loss – но и надо, чтобы  $G_i$  не игнорировал  $z$ , надо добавить diversity; для этого максимизируем энтропию:

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

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

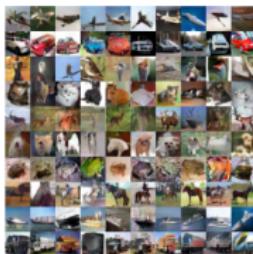
# Stacked GANs

- SGAN:



# Stacked GANs

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



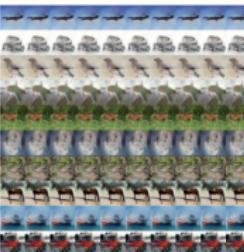
(a) SGAN samples (conditioned on labels)



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

<sup>†</sup> Trained with labels.

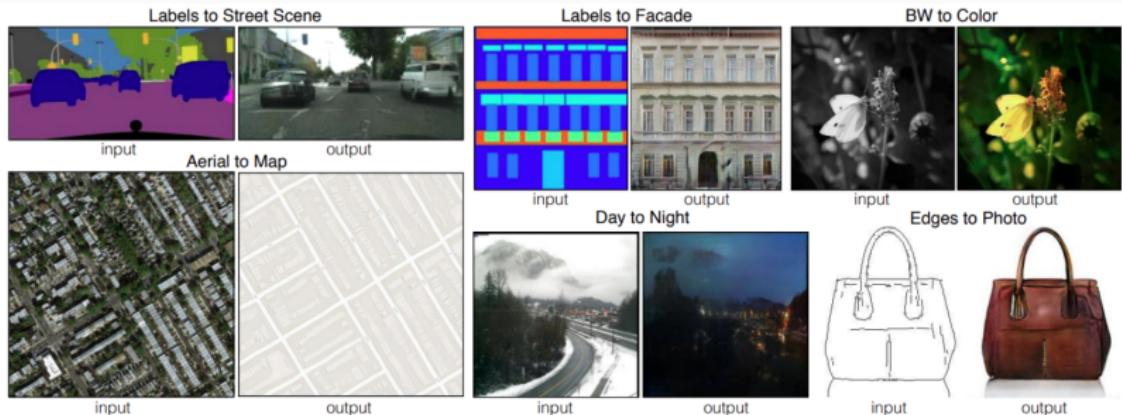
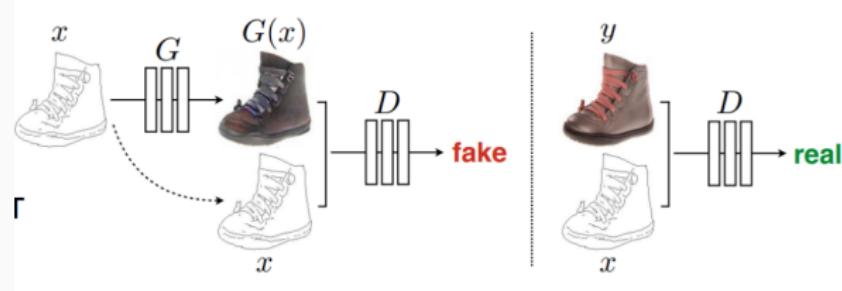
Table 1: **Inception Score on CIFAR-10.** SGAN and SGAN-no-joint outperform previous state-of-the-art approaches.

## Case study: перенос стиля

---

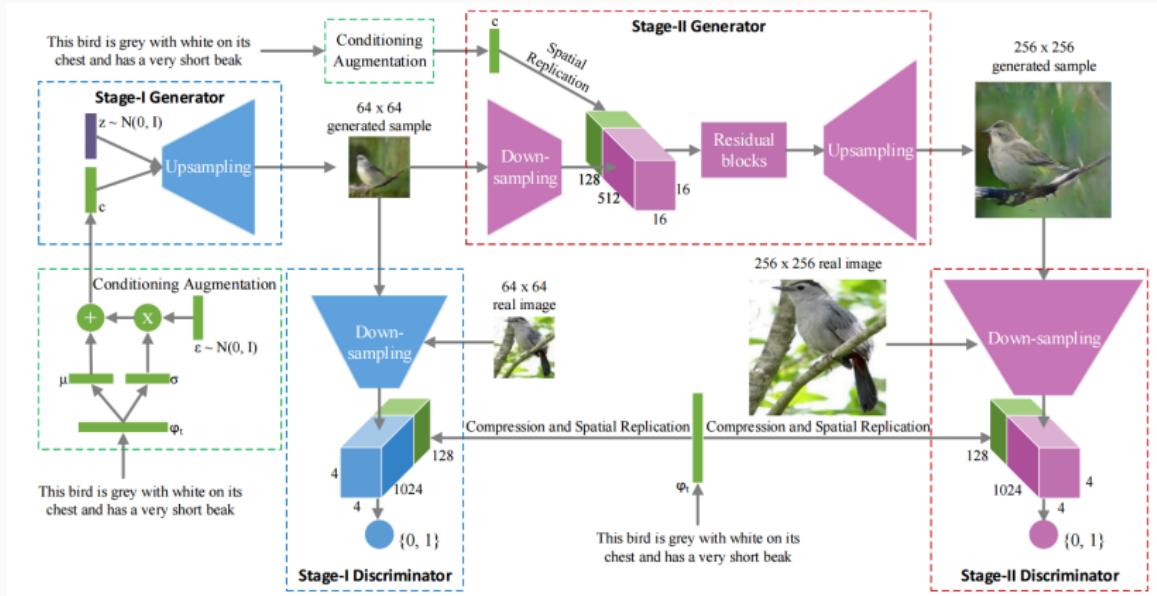
# Перенос стиля с GAN'ами

- 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:



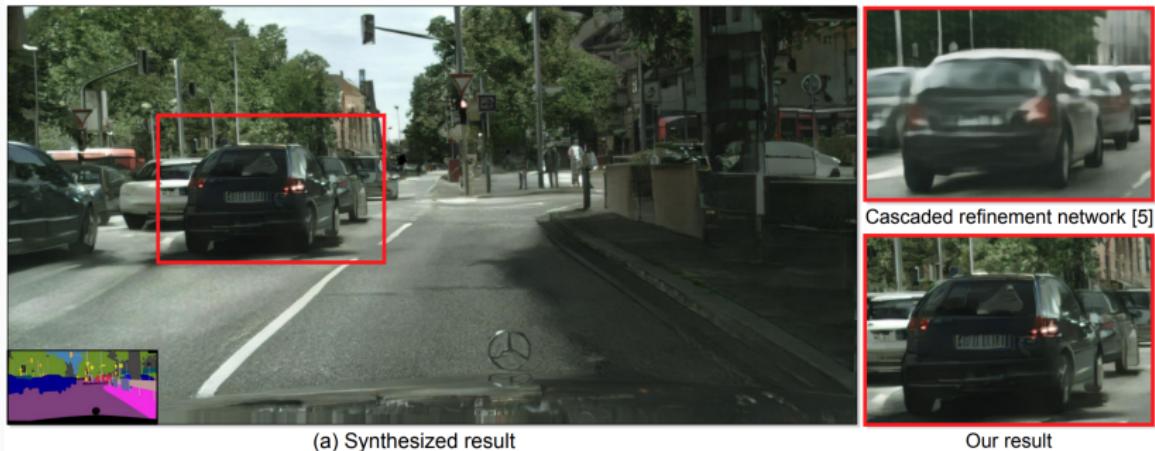
# Перенос стиля с GAN'ами

- Вот результаты реального порождения (best of 16):



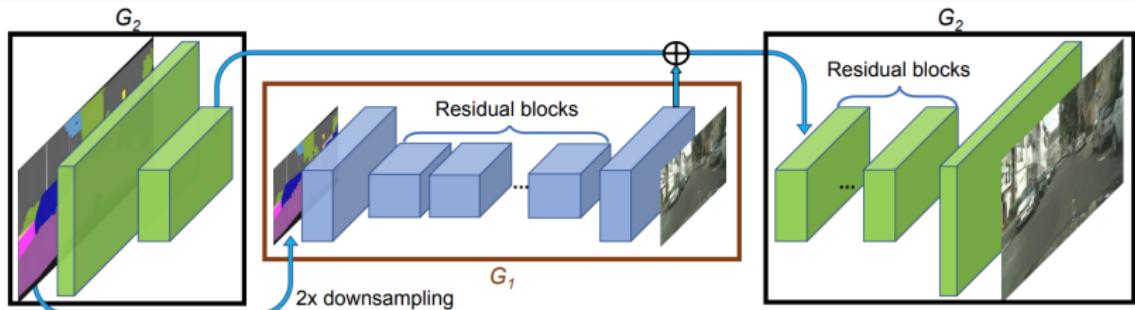
# Перенос стиля с GAN'ами

- 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  $\gamma$  и  $\beta$  обучаются для каждого стиля:

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

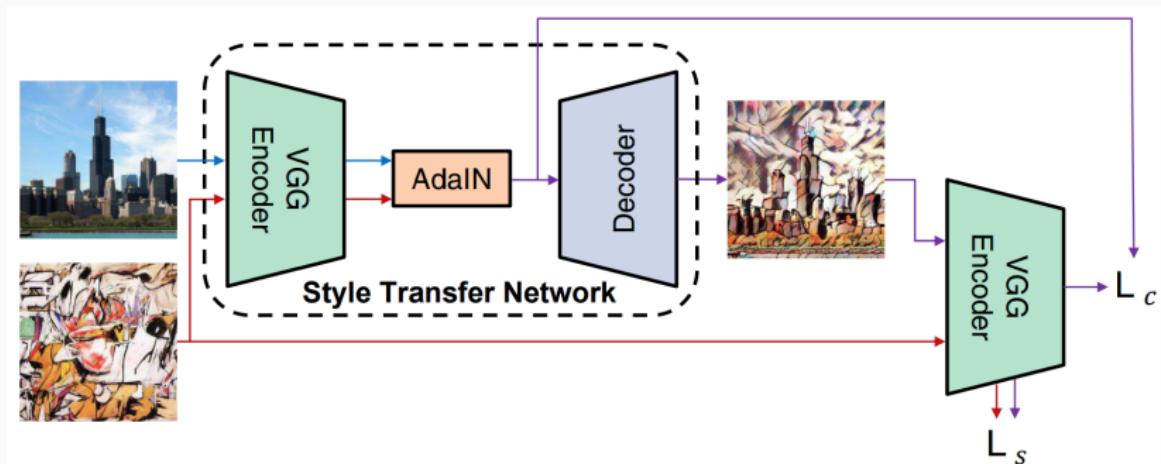
- AdaIN – это то же самое, но мы просто берём параметры от другой картинки, которая задаёт стиль:

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

- Чем-то похоже на самый ранний artistic style transfer (Catvs.st)

# Перенос стиля с GAN'ами

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



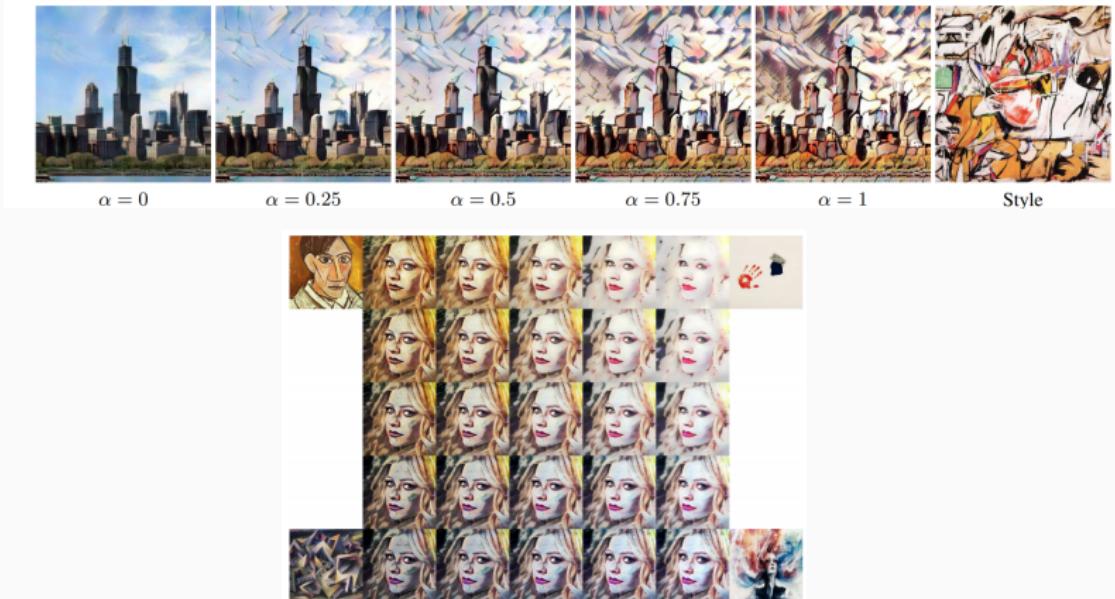
# Перенос стиля с GAN'ами

- И получается круто:



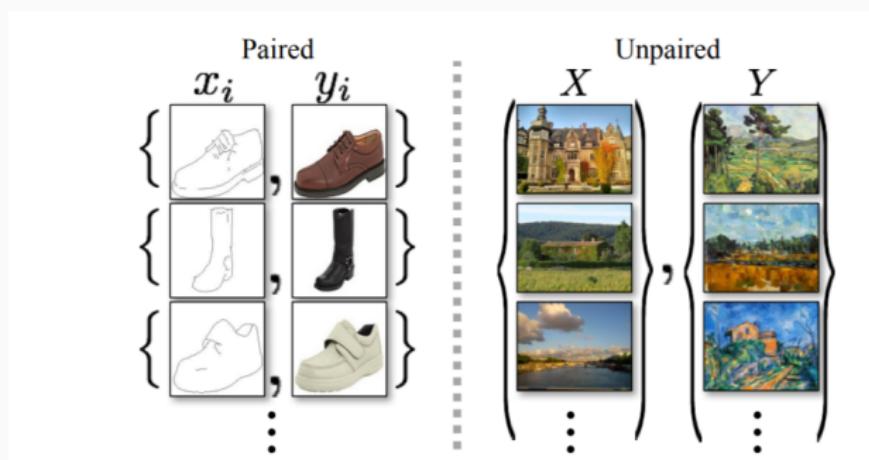
# Перенос стиля с GAN'ами

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



# CycleGAN

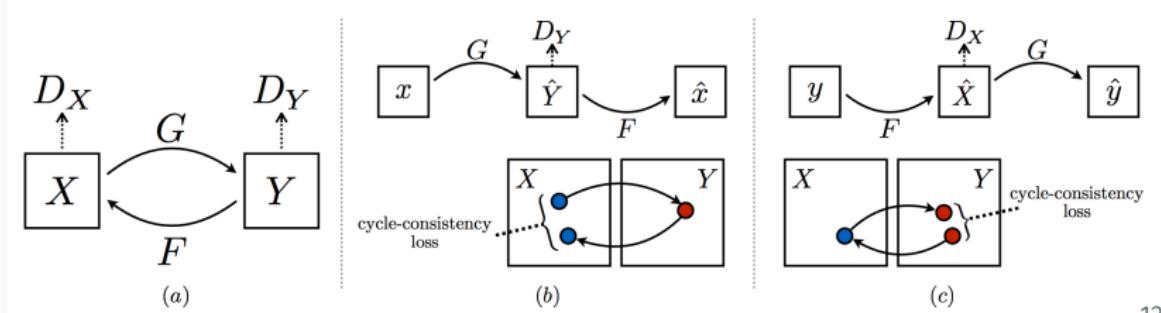
- CycleGAN (Zhu et al., 2017)
  - что делать, если данные unpaired?
  - например, для style transfer:



# CycleGAN

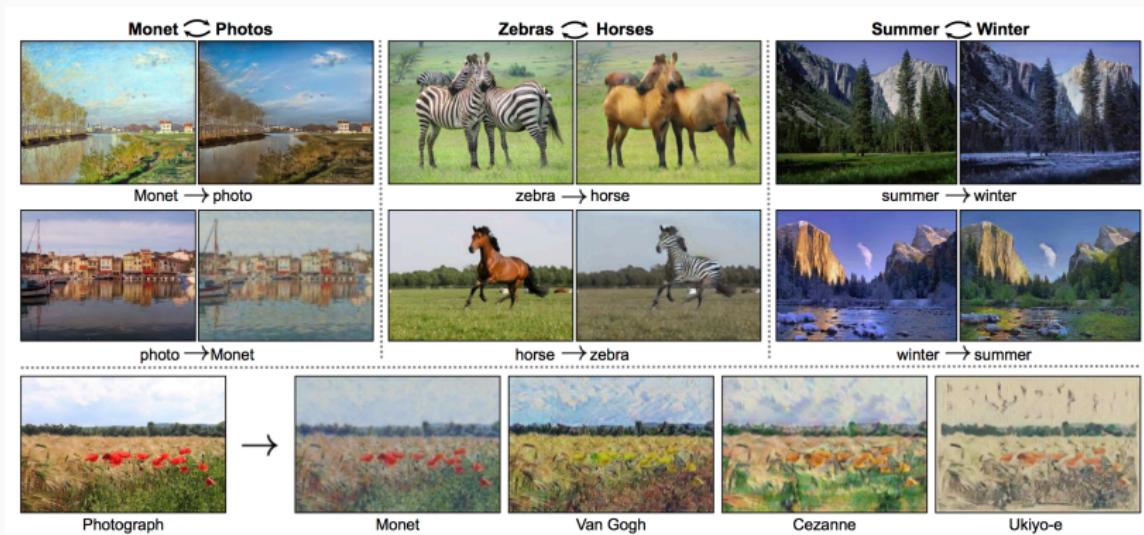
- CycleGAN (Zhu et al., 2017)
- Основная идея в том, что после двойного цикла style transfer должно опять получиться то же самое,  $G(F(x)) = x$
- Общая функция потерь складывается из двух GAN'ов для  $G$  и  $F$  и cycle loss:

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



# CycleGAN

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



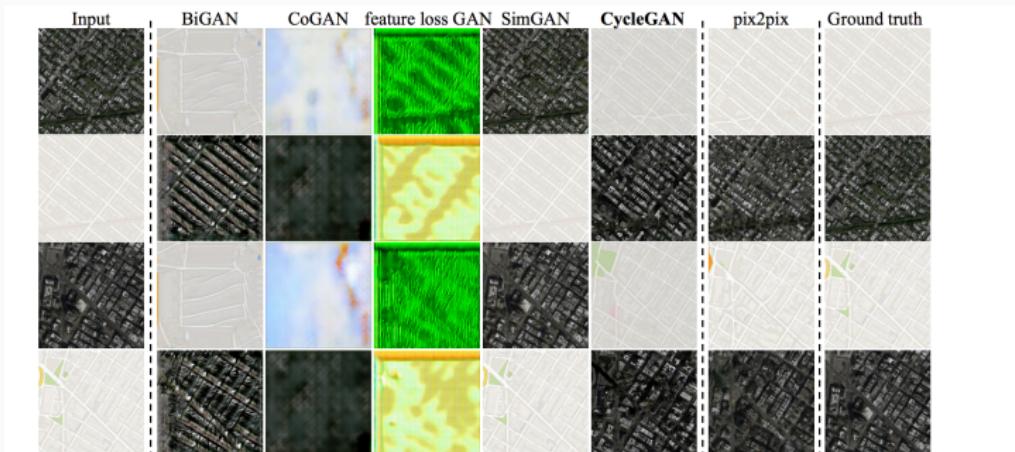
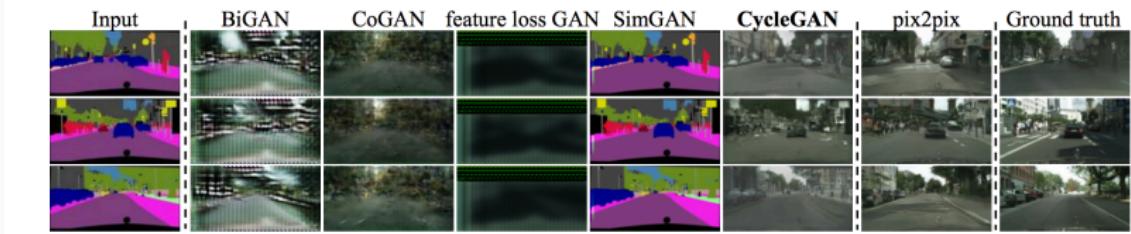
# CycleGAN

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



# CycleGAN

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



Спасибо!

Спасибо за внимание!

