

Порождающие состязательные сети II

Сергей Николенко

НИУ ВШЭ – Санкт-Петербург

12 декабря 2020 г.

Random facts:

- 12 декабря 627 года состоялась битва при Ниневии, в результате которой византийцы после 400 лет войн смогли-таки отстоять Малую Азию
- 12 декабря 1910 г. в Нью-Йорке наследница парфюмерной компании Дороти Арнольд отправилась за новым платьем и так и не вернулась; слухов и версий было много, семья искала её почти десять лет, но судьба её так и осталась неизвестной
- 12 декабря 1935 г. Генрих Гиммлер основал проект Lebensborn («Источник жизни») для подготовки молодых «расово чистых» матерей и воспитания «арийских» младенцев; а 12 декабря 1941 года на встрече Рейхсканцелярии Адольф Гитлер фактически объявил об окончательном решении еврейского вопроса
- 12 декабря 1993 г. всенародным голосованием была принята новая Конституция РФ
- 12 декабря 2000 г. Верховный суд США выпустил своё решение по делу Bush v. Gore, подтвердив, что Джордж Буш-младший стал президентом

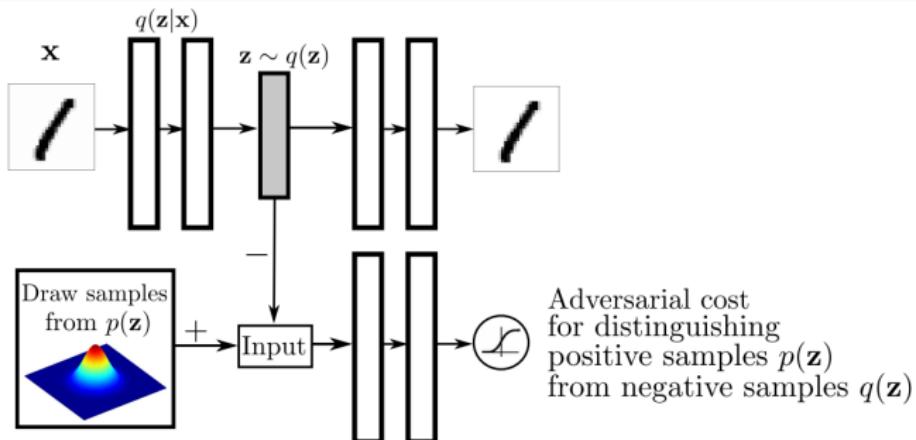
Что сейчас с GAN'ами делают

- 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.

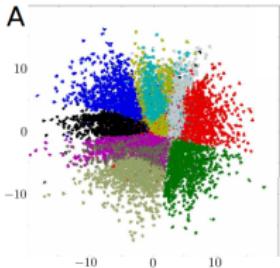


Архитектуры, основанные на GAN

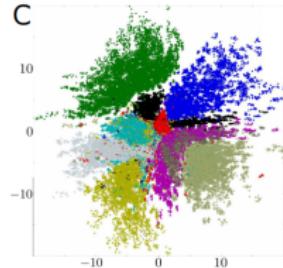
- Соперничающие автокодировщики (adversarial autoencoders; Makhzani et al., 2015): дискриминатор приводит распределение признаков (на скрытом слое) к заданному $p(z)$.



Adversarial Autoencoder



Variational Autoencoder

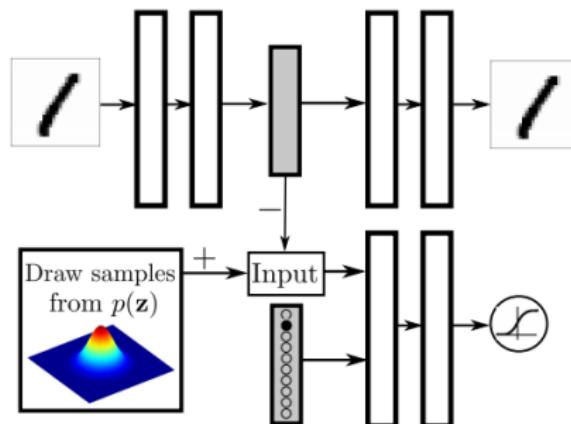


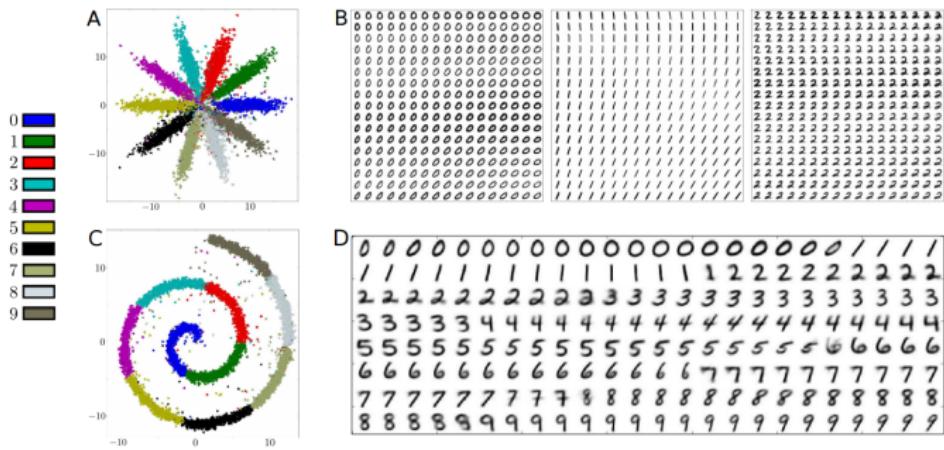
Manifold of Adversarial Autoencoder

A horizontal color scale with five pairs of colored squares. Each pair consists of a colored square on the left and a numerical label on the right. The colors are blue, green, red, cyan, and magenta, arranged from top to bottom. The numerical labels are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9, also arranged from top to bottom.

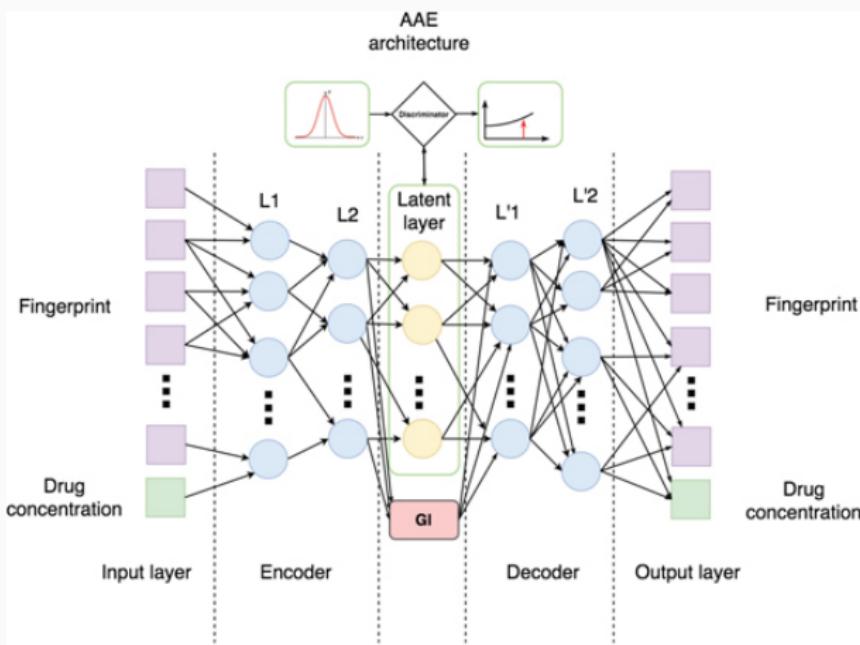
0	5
1	6
2	7
3	8
4	9

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

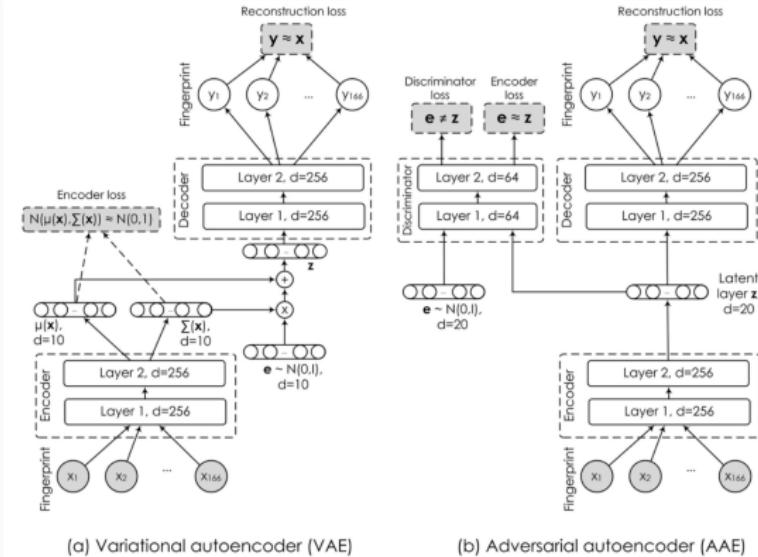




- (Kadurin et al., 2017) – ААЕ для порождения молекул в онкологии:



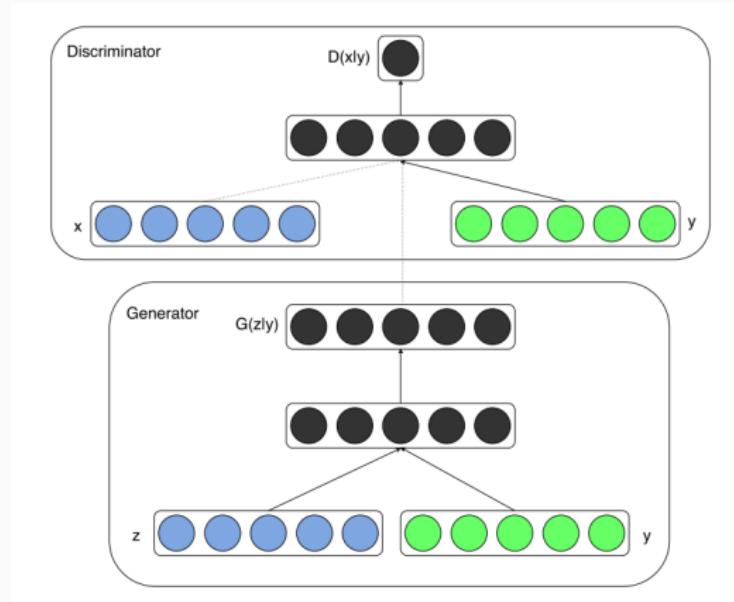
- (Kadurin et al., 2018) – druGAN для порождения молекул; сравнили с VAE:



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

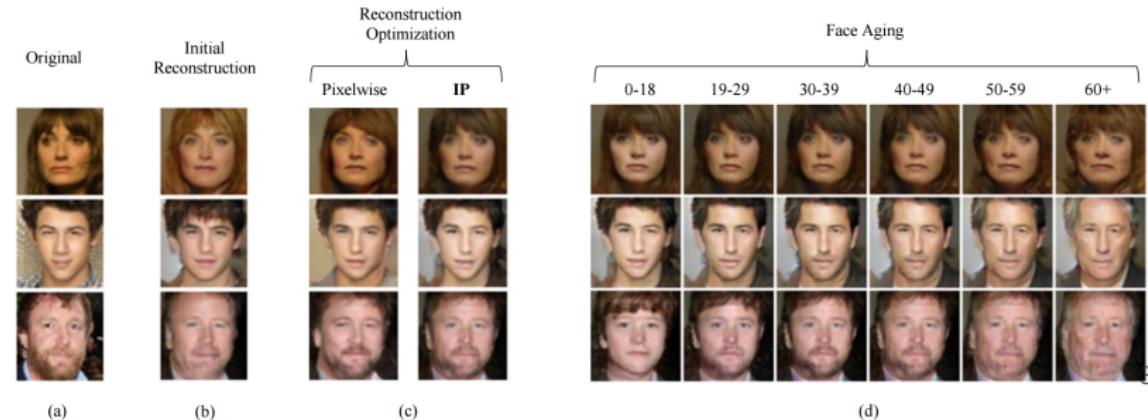
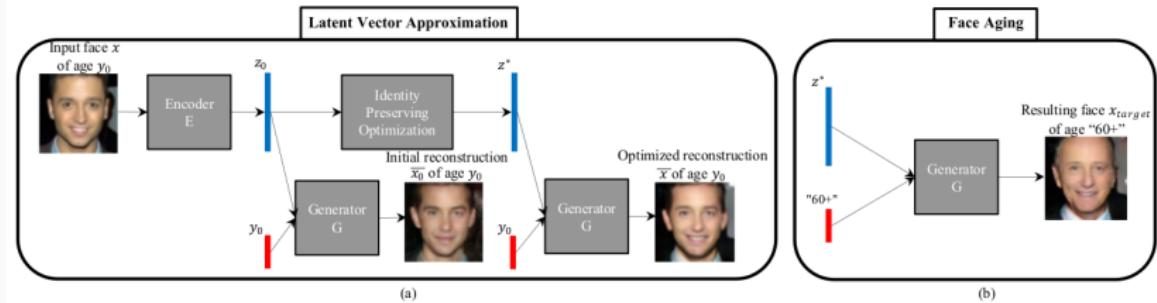
Условные GAN'ы

- 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:

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

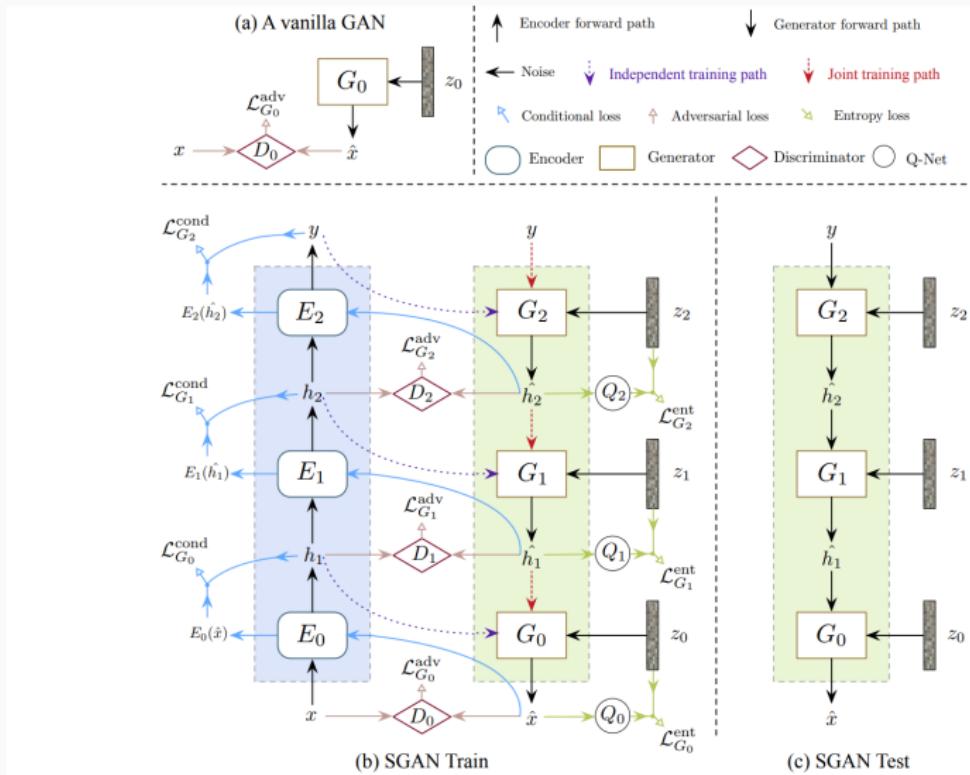
- entropy loss – но и надо, чтобы G_i не игнорировал z , надо добавить diversity; для этого максимизируем энтропию:

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

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

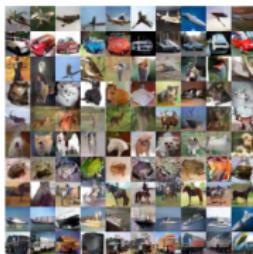
Stacked GANs

- SGAN:



Stacked GANs

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



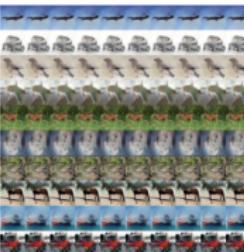
(a) SGAN samples (conditioned on labels)



(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 ± 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

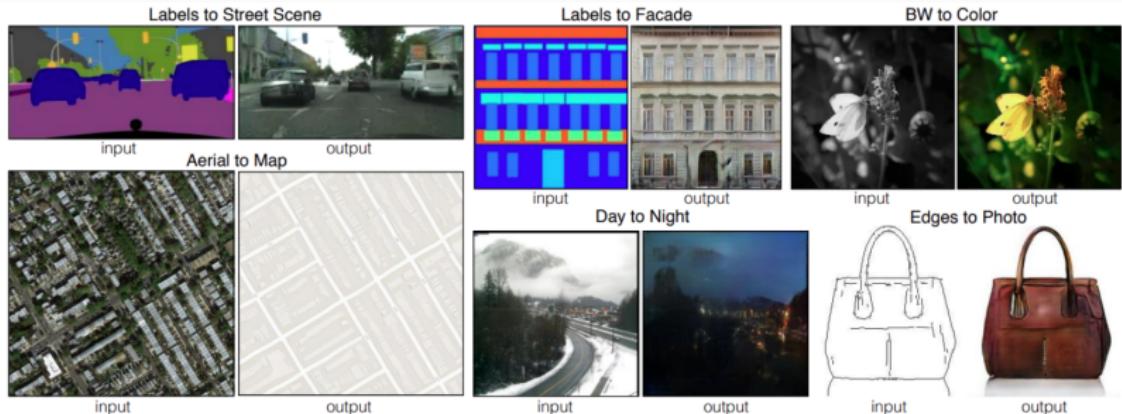
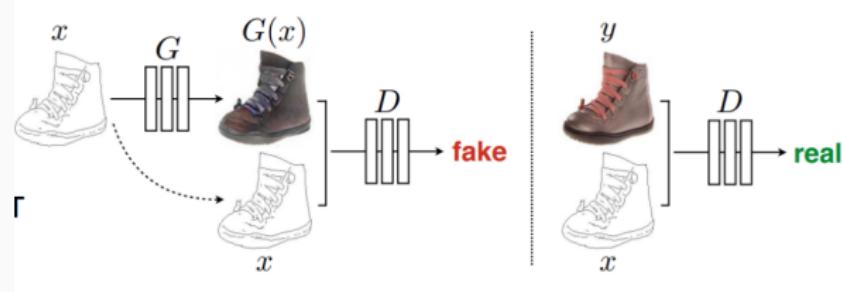
[†] Trained with labels.

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

Case study: перенос стиля

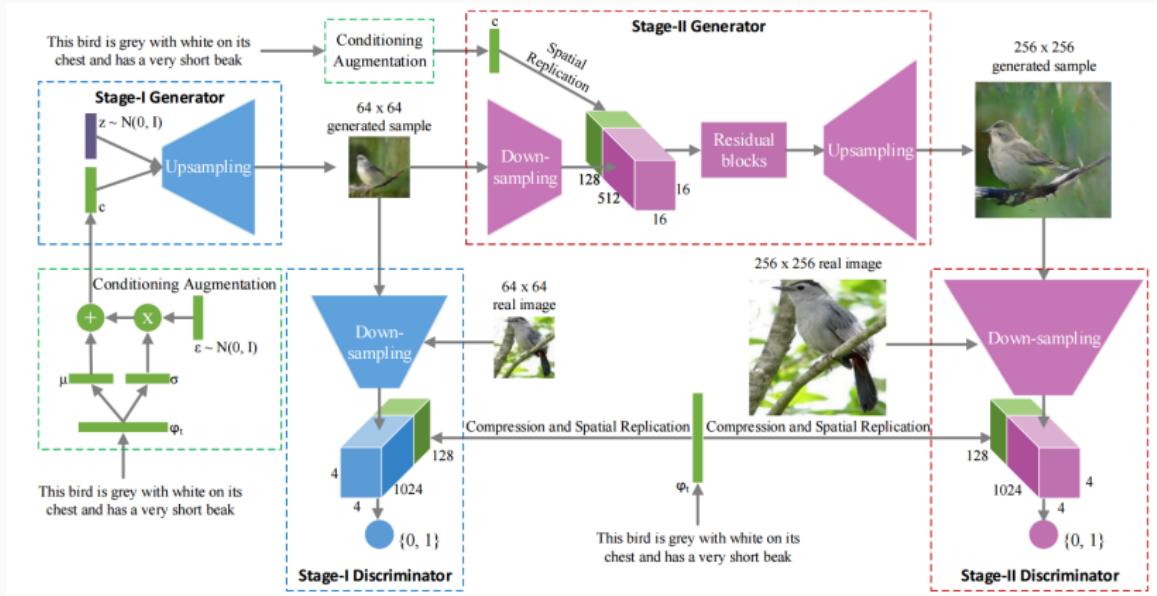
Перенос стиля с GAN'ами

- 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:



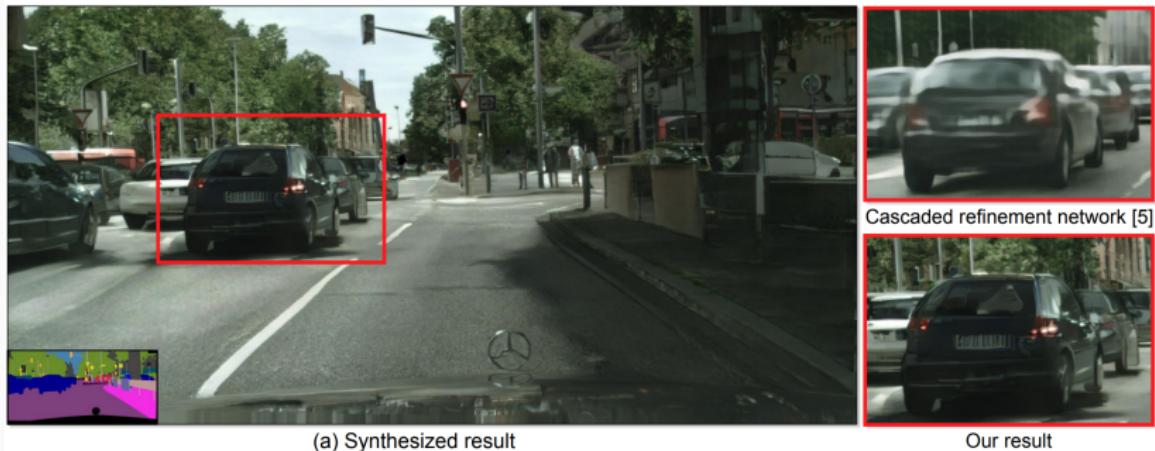
Перенос стиля с GAN'ами

- Вот результаты реального порождения (best of 16):



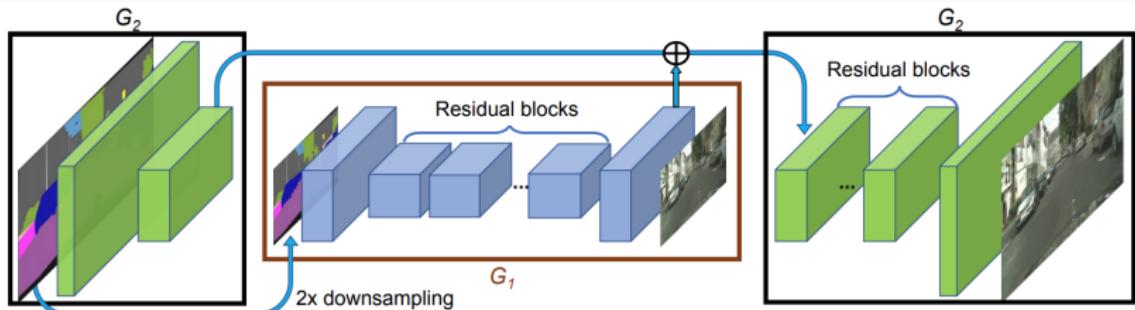
Перенос стиля с GAN'ами

- 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:

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

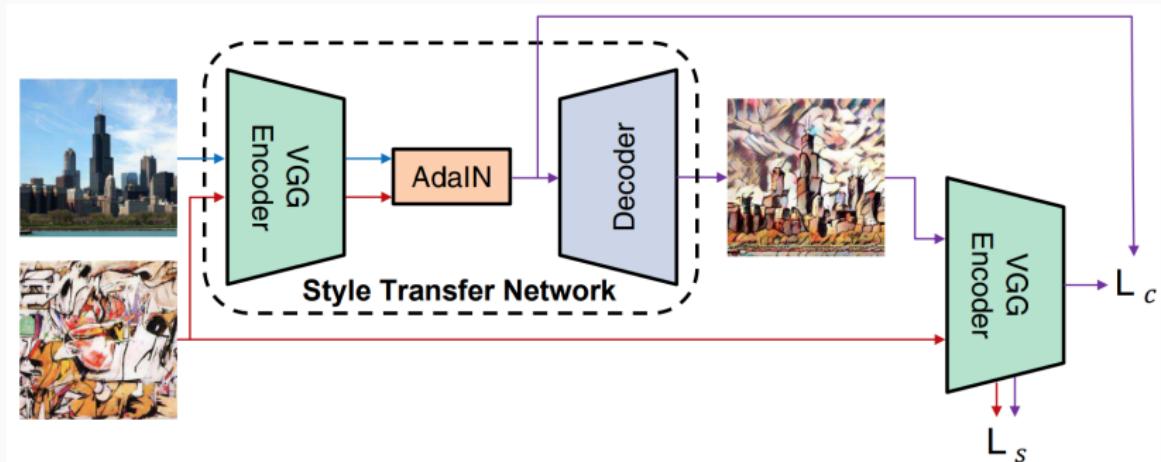
- AdaIN – это то же самое, но мы просто берём параметры от другой картинки, которая задаёт стиль:

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

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

Перенос стиля с GAN'ами

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



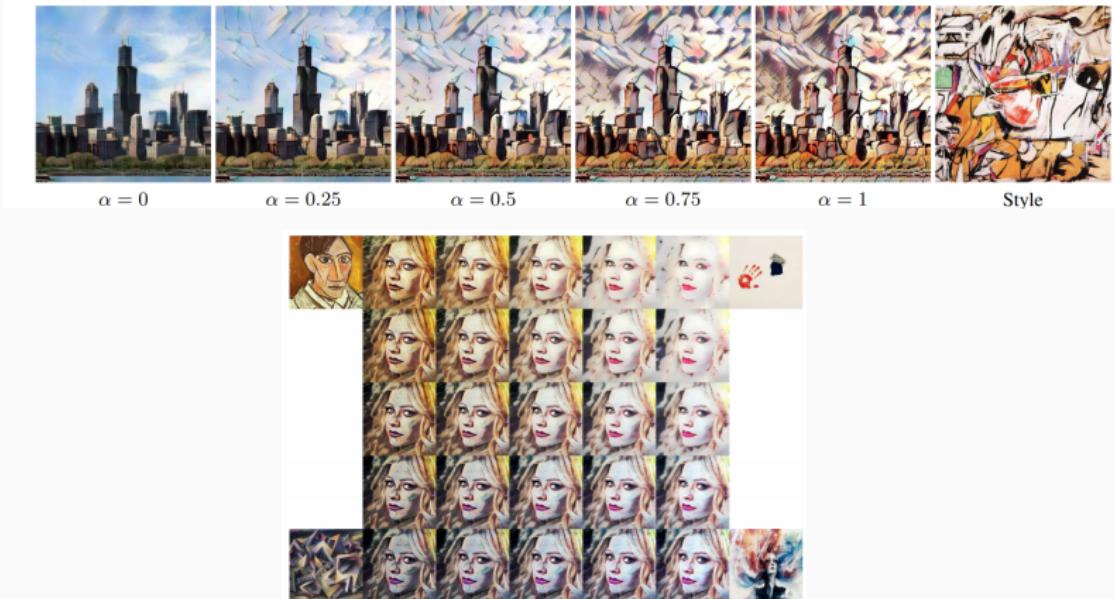
Перенос стиля с GAN'ами

- И получается круто:



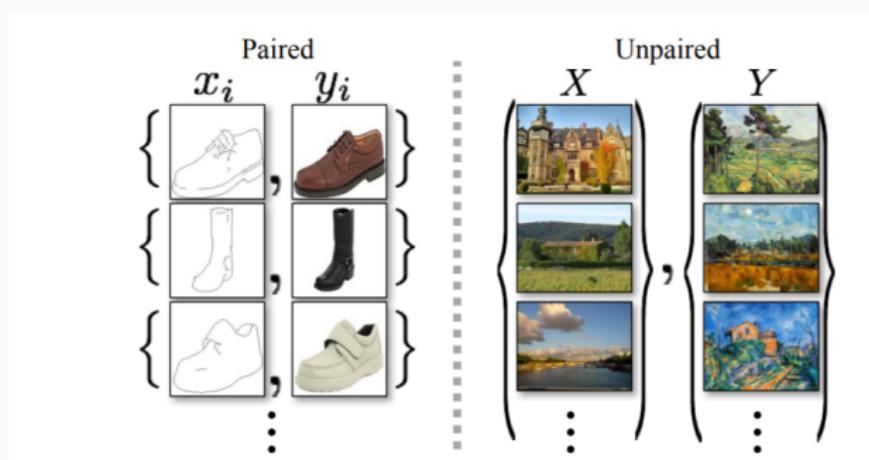
Перенос стиля с GAN'ами

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



CycleGAN

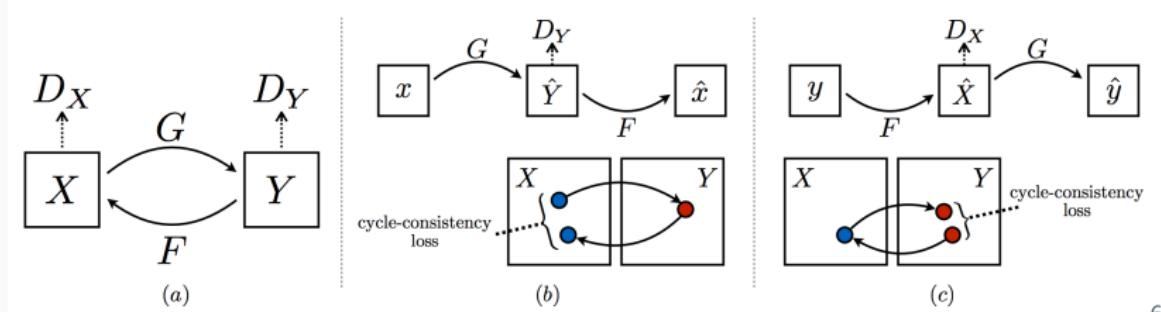
- CycleGAN (Zhu et al., 2017)
 - что делать, если данные unpaired?
 - например, для style transfer:



CycleGAN

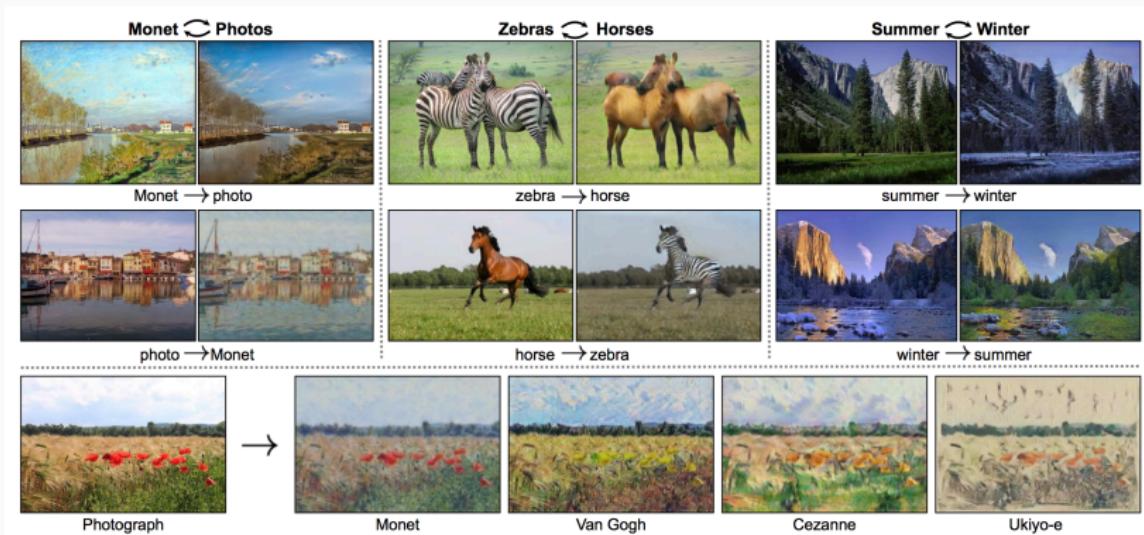
- CycleGAN (Zhu et al., 2017)
- Основная идея в том, что после двойного цикла style transfer должно опять получиться то же самое, $G(F(x)) = x$
- Общая функция потерь складывается из двух GAN'ов для G и F и cycle loss:

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



CycleGAN

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



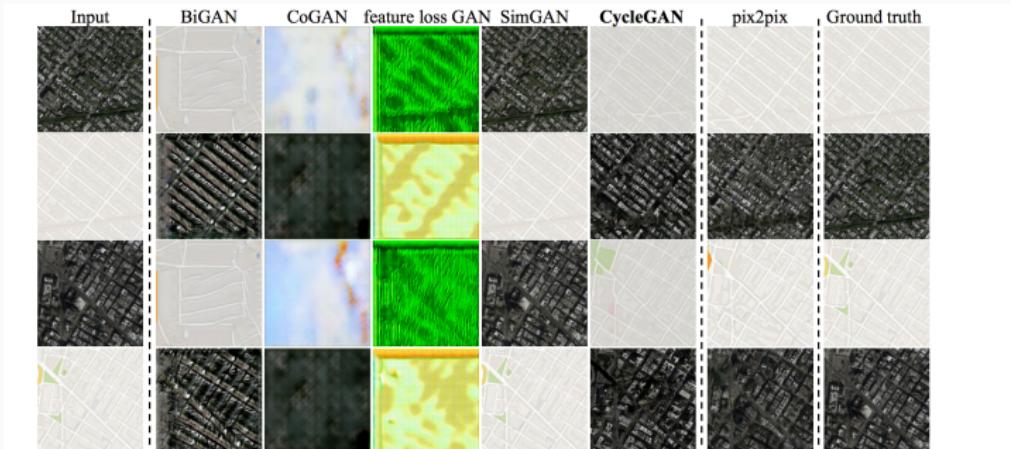
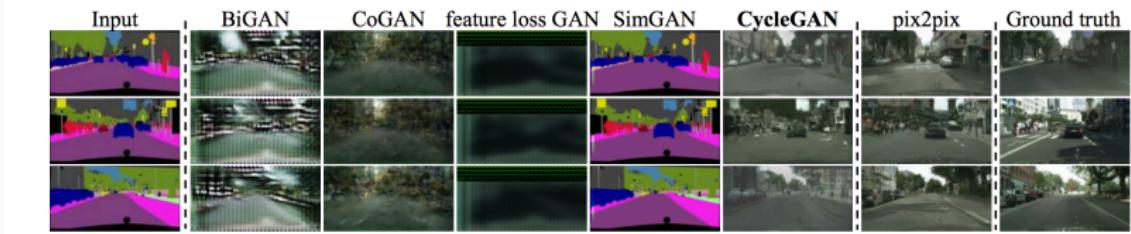
CycleGAN

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



CycleGAN

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



Case study: Superresolution

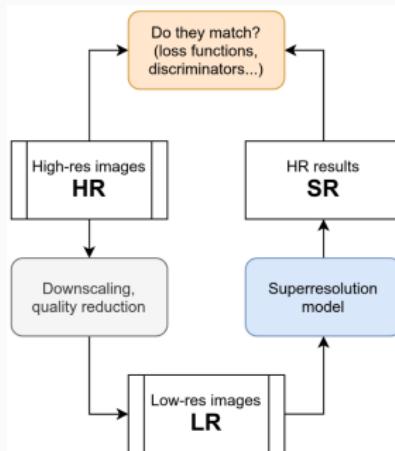
Superresolution

- Смысл задачи – как перейти от картинок слева к картинкам справа?
- Т.е. надо научиться увеличивать разрешение фотографий



Superresolution

- Почти все решения работают так:
 - берём датасет картинок высокого разрешения
 - делаем из них картинки низкого разрешения
 - обучаем модель восстанавливать high-res; т.е. paired dataset как бы и не нужен...



Superresolution

- Прежде чем начнём что-то делать: как мы будем измерять результаты?
- Стандартные метрики:
 - SSIM, structural similarity index: для двух окон x и y

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)},$$

где $c_1 = (0.01 \cdot L)^2$, $c_2 = (0.03 \cdot L)^2$, L – максимальное значение пикселя (динамический диапазон)

- это на самом деле комбинация трёх метрик: luminance, contrast, structure
- PSNR, peak signal-to-noise ratio – это просто функция от MSE:

$$\text{MSE}(x, y) = \frac{1}{mn} \sum_{i,j} (x_{i,j} - y_{i,j})^2,$$

$$\text{PSNR}(x, y) = 10 \log_{10} \left(\frac{\max_x}{\text{MSE}} \right),$$

где \max_x – максимальное значение пикселя

Superresolution

- SSIM и PSNR – основные метрики в литературе по SR, да и вообще в оценках качества, например, компрессии изображений
- Но в какой-то момент они перестают уже различать модели, и они не точно соответствуют человеческому взгляду
- Поэтому есть большая область разработки и сравнения разных метрик качества; (Athar, Wang, 2019):

Part I: All Databases			
FR Method	PLCC	FR Method	SRCC
RAS5	0.8985	RAS4	0.8907
RAS6	0.8979	RAS5	0.8903
RAS4	0.8977	RAS1	0.8783
RAS_MMF4	0.8967	RAS2	0.8777
RAS7	0.8935	RAS_MMF4	0.8771
RAS_MMF3	0.8912	RAS6	0.8761
RAS3	0.8911	RAS_MMF3	0.8724
RAS1	0.8908	RAS7	0.8710
RAS_MMF2	0.8888	RAS_MMF2	0.8690
RAS_B1	0.8881	RAS3	0.8665
RAS2	0.8879	RAS_B1	0.8653
RAS_B2	0.8850	CISI	0.8634
CISI	0.8831	VSI	0.8631
IWSSIM	0.8787	FSIMc	0.8628
RAS_MMF1	0.8786	RAS_B2	0.8607
ESIMs	0.8705	IWSSIM	0.8550

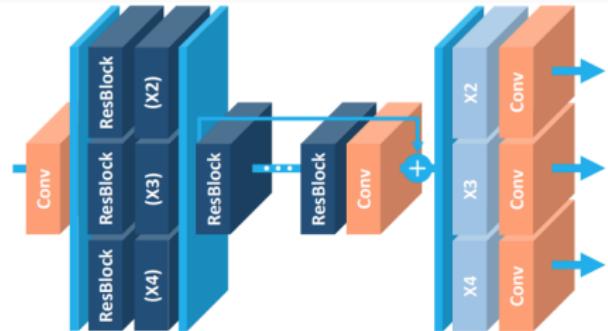
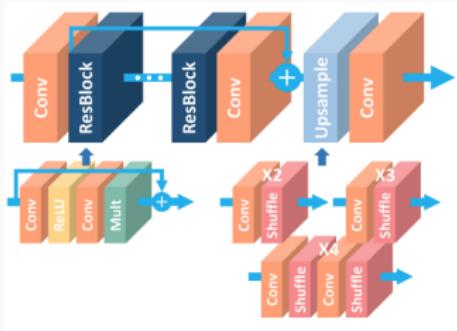
- Например:
 - VSI, Visual Saliency-Induced Index (Zhang et al., 2014): сравниваем, взвешивая пиксели по saliency maps, которые пытаются найти патчи, важные для человеческого внимания;
 - SFF, Sparse Feature Fidelity (Chang et al., 2013): сравниваем, переводя сначала в разреженные представления, которые должны, по идее, соответствовать тому, что делает зрительная кора...
- В общем, большая наука, далёкая от завершения. Давайте вернёмся к собственно методам superresolution.

Superresolution

- Есть три основных подхода, на которые можно классифицировать методы. Оказывается, что от метода, которым мы пользуемся для уменьшения картинок, очень многое зависит:
 - *non-blind SR* – зафиксируем некоторое «идеальное» ядро для уменьшения (например, бикубическое); тогда всё будет работать очень хорошо, когда ядро действительно совпадает, но будет быстро ухудшаться, когда ядро будет другим, а оно ведь будет другим (Shocher et al., 2018; Lim et al., 2017; Zhang et al., 2018)
 - *blind SR* – методы, которые стараются не делать предположений о ядре
 - *kernel estimation* – давайте заставим наш blind SR «прозреть», обучив подходящее ядро для каждой отдельной картинки.

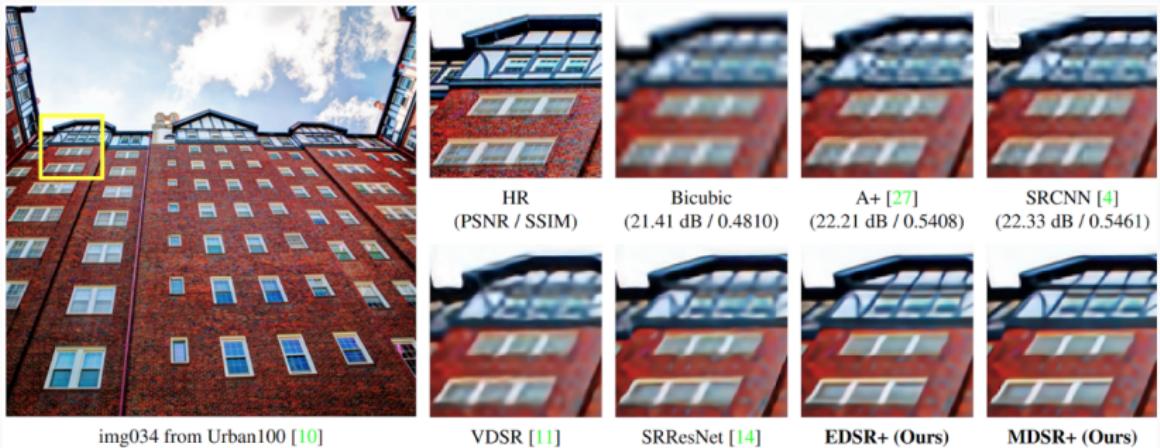
Superresolution

- Пример supervised подхода: EDSR (Enhanced Deep Super-Resolution network), уже классика (Lim et al., 2017)
- Обучаем просто свёрточную сеть восстанавливать картинки; есть single-scale и multi-scale варианты



Superresolution

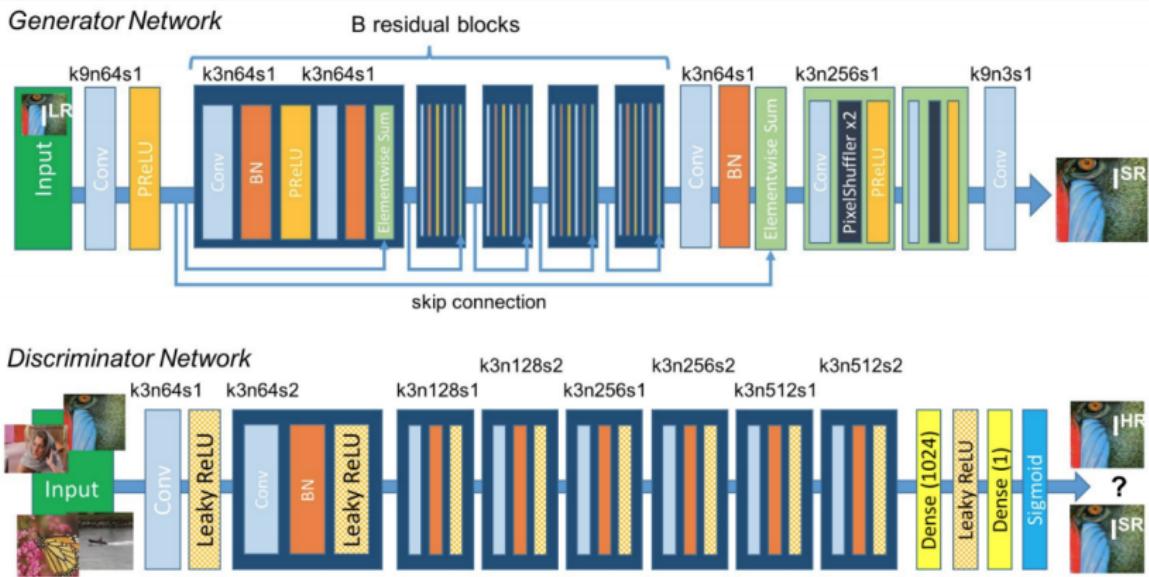
- EDSR долго определял state of the art и вообще не так уж плохо работал:



- А дальше начались как раз GAN'ы...

Superresolution

- SRGAN (Ledig et al., 2016): базовая схема GAN-based SR:
 - генератор делает SR
 - дискриминатор пытается отличить результаты генератора от настоящих HR картинок



Superresolution

- Дискриминатор оптимизирует обычную кросс-энтропию, как всегда:

$$\max_D \mathbb{E}_{I_{\text{HR}}} [\log D(I_{\text{HR}})] + \mathbb{E}_{I_{\text{LR}}} [\log(1 - D(G(I_{\text{LR}})))]$$

- А интересная часть – как сделать функцию ошибки для генератора:

$$\mathcal{L}^{\text{SR}} = \mathcal{L}_*^{\text{SR}} + \lambda \cdot \mathcal{L}_{\text{adv}}^{\text{SR}},$$

где $\mathcal{L}_{\text{adv}}^{\text{SR}}$ – adversarial loss, как обычно:

$$\mathcal{L}_{\text{adv}}^{\text{SR}} = \sum_{n=1}^N -\log D(G(I_{\text{LR}}))$$

Superresolution

- А вот $\mathcal{L}_*^{\text{SR}}$ – content loss – уже интереснее; SRGAN рассматривают MSE как вариант, но пишут, что лучше работает VGG loss:

$$\mathcal{L}_{\text{VGG}}^{\text{SR}} = \frac{1}{WH} \sum_{i=1}^w \sum_{j=1}^h (\phi(I_{\text{HR}})_{ij} - \phi(G(I_{\text{LR}}))_{ij})^2,$$

где ϕ – признаки, выделенные где-то в VGG-сети

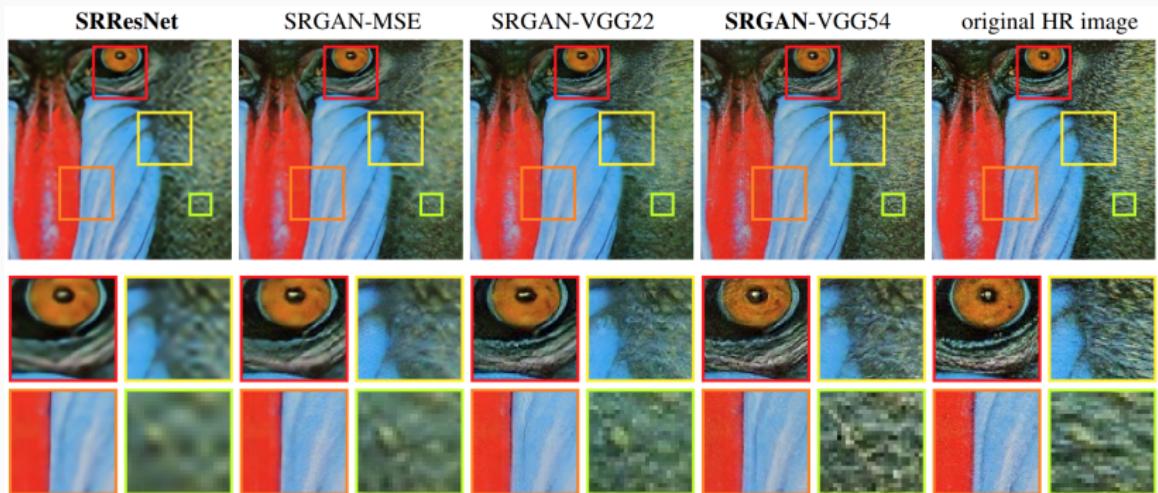
- Разница прямо есть:

Set5	SRResNet-		SRGAN-		
	MSE	VGG22	MSE	VGG22	VGG54
PSNR	32.05	30.51	30.64	29.84	29.40
SSIM	0.9019	0.8803	0.8701	0.8468	0.8472
MOS	3.37	3.46	3.77	3.78	3.58

Set14			
PSNR	28.49	27.19	26.92
SSIM	0.8184	0.7807	0.7611
MOS	2.98	3.15*	3.43

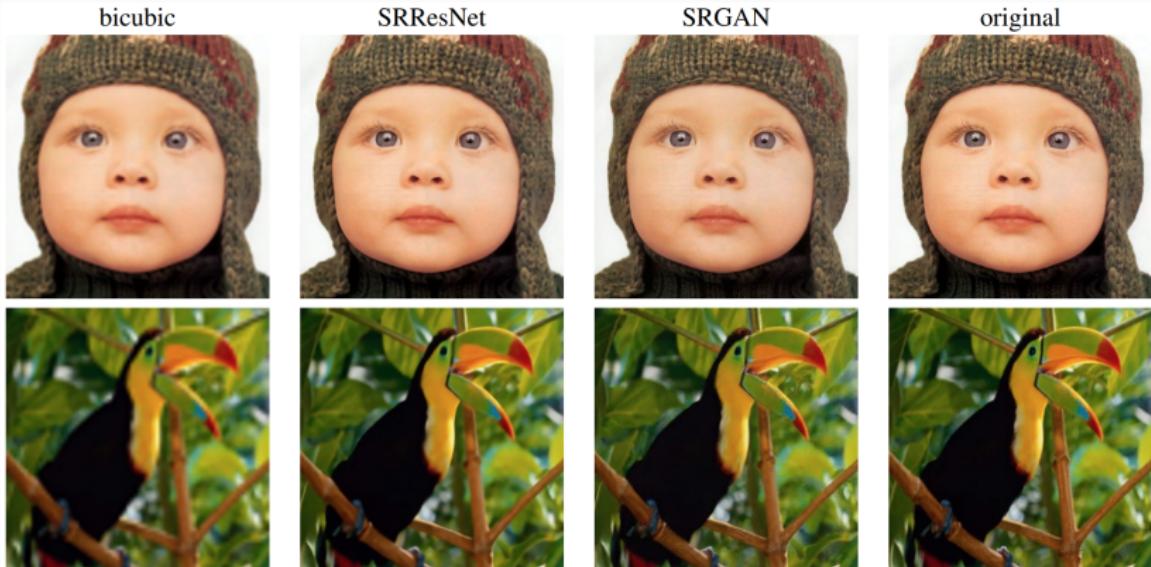
Superresolution

- И на практике тоже видна эта разница:



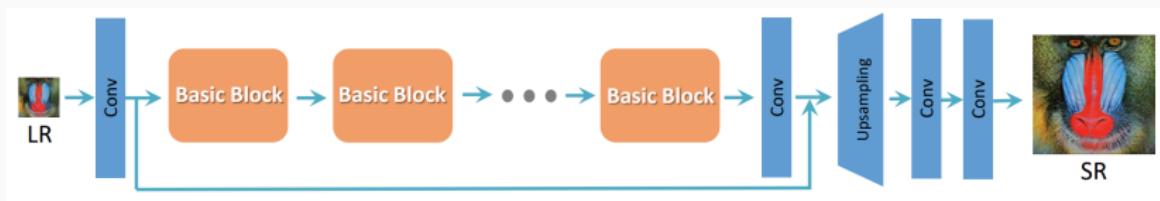
Superresolution

- Кажется, что уже лучше трудно и придумать, но на самом деле всё только начинается...

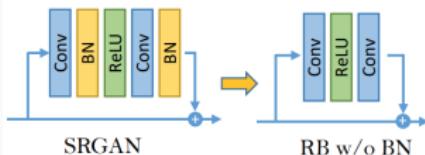


Superresolution

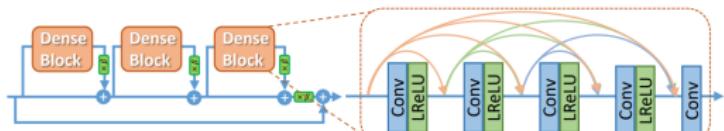
- ESRGAN (Wang et al., 2018) улучшает базовую идею SRGAN; тот же принцип, но архитектуры другие, и генератор в основном работает в feature space LR-картинок:



Residual Block (RB)



Residual in Residual Dense Block (RRDB)



Superresolution

- Relativistic discriminator:

$$D(x_r) = \sigma(C(\text{Real})) \rightarrow 1 \quad \text{Real?}$$

$$D(x_f) = \sigma(C(\text{Fake})) \rightarrow 0 \quad \text{Fake?}$$



a) Standard GAN

$$D_{Ra}(x_r, x_f) = \sigma(C(\text{Real}) - \mathbb{E}[C(\text{Fake})]) \rightarrow 1$$

$$D_{Ra}(x_f, x_r) = \sigma(C(\text{Fake}) - \mathbb{E}[C(\text{Real})]) \rightarrow 0$$

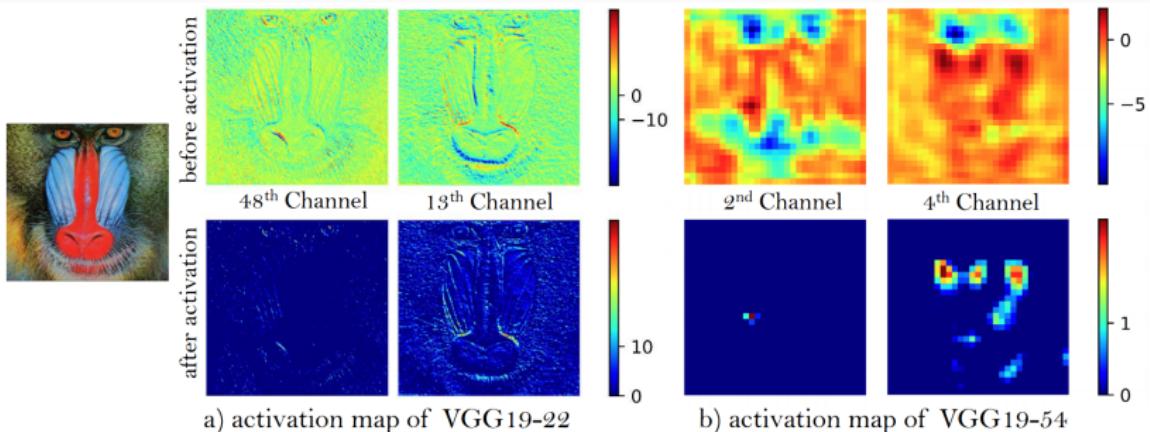
b) Relativistic GAN

More realistic
than fake data?

Less realistic
than real data?

Superresolution

- Perceptual loss тоже немножко изменился – теперь берём признаки не после функции активации, а до, там меньше разреженность:



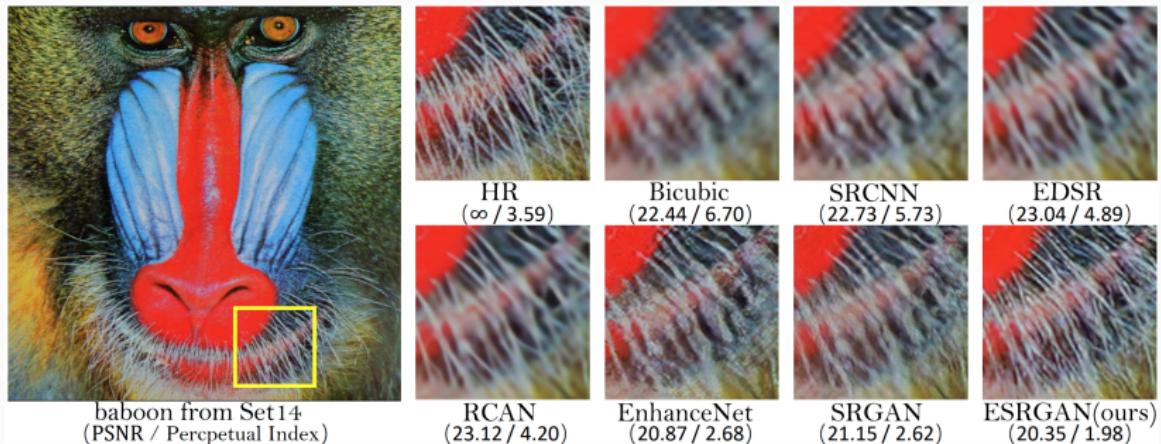
- Но в принципе то же самое:

$$\mathcal{L}^{\text{SR}} = \mathcal{L}_*^{\text{SR}} + \lambda \cdot \mathcal{L}_{\text{adv}}^{\text{SR}} + \eta L_1.$$

- Ещё интересно, что обучают G_{PSNR} на основе PSNR, потом делают fine-tuning в виде GAN до G_{GAN} , а потом интерполируют веса между этими двумя сетями.

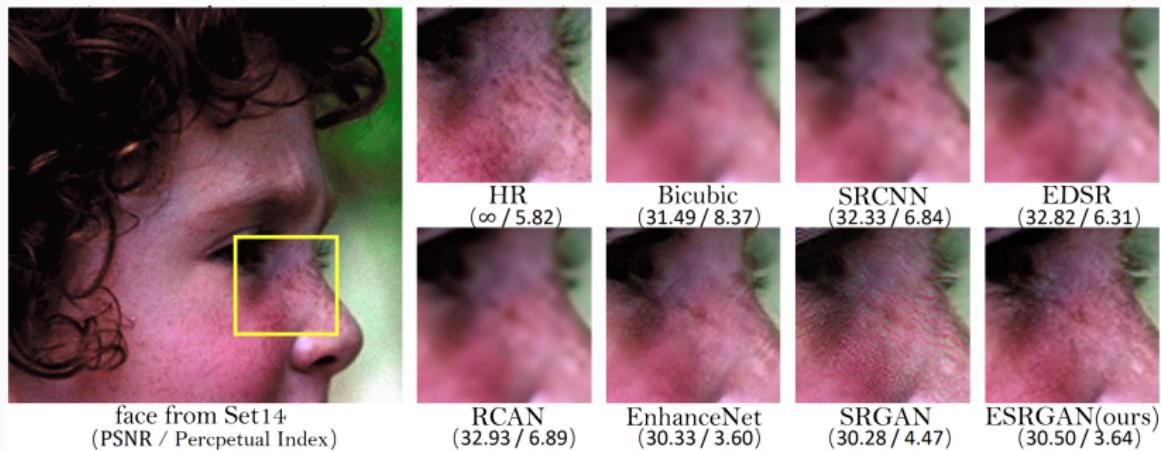
Superresolution

- И действительно разницу можно увидеть:



Superresolution

- И действительно разницу можно увидеть:



Superresolution

- А вот разница между G_{PSNR} и G_{GAN} :



Superresolution

- ESRGAN довольно актуален до сих пор. Но есть и новые веяния
- FastAI: Decrappify, Deoldify, and Superresolution



'Migrant Mother' by Dorothea Lange (1936) colorized by DeOldify (right) and baseline algorithm (center)

Superresolution

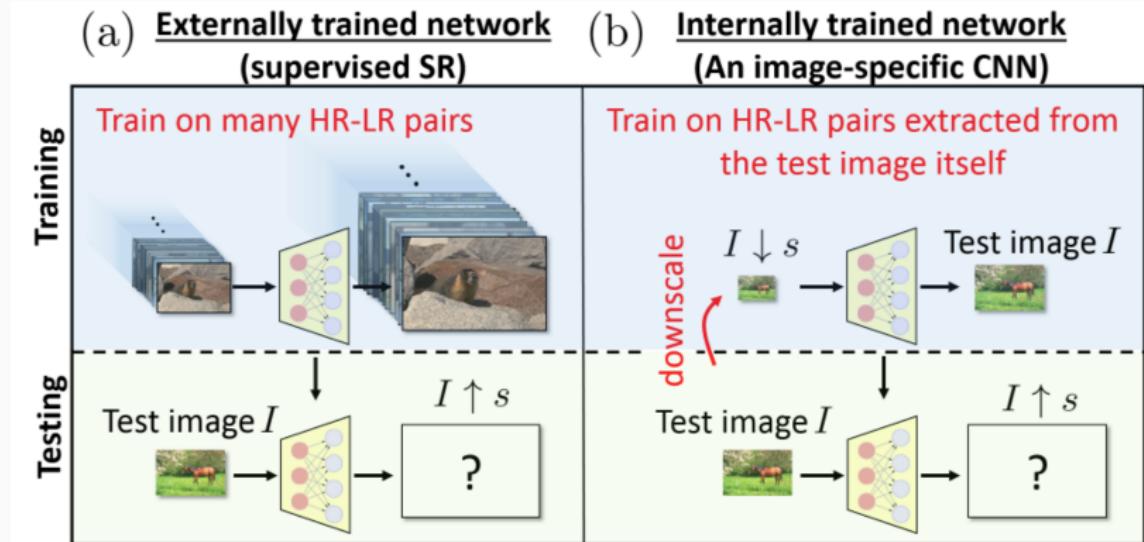
- Современная версия очень простого метода: давайте максимально ухудшим хорошие картинки, добавим кучу шума и аугментаций, и попросим восстановить
- Пишут, что пытались добавить GAN'ы, но оказалось, что в этом пайплайне GAN'ы не особенно и нужны; NoGAN approach
- <https://www.fast.ai/2019/05/03/decrappify/>



'Gypsy Camp, Maryland' (1925)

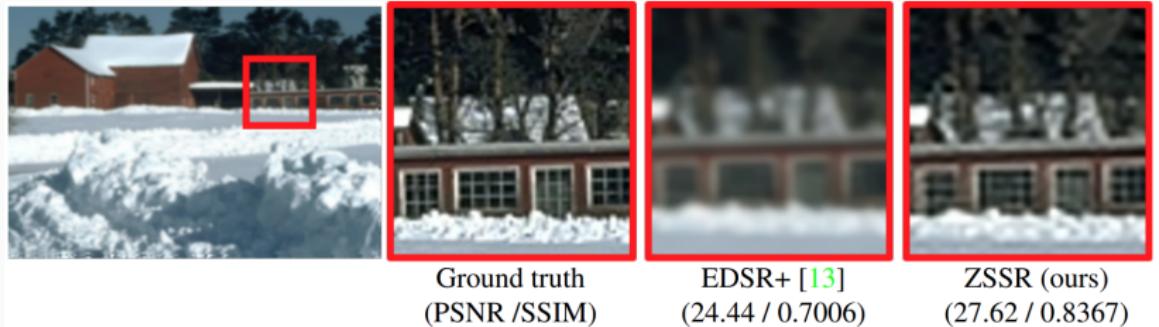
Superresolution

- Но всё это было в том или ином смысле non-blind. А как понять, какое ядро лучше всего восстанавливать?
- ZSSR (Zero-Shot Super-Resolution), Shocher et al. (2019): обучаем ядро прямо на текущей картинке, self-supervised



Superresolution

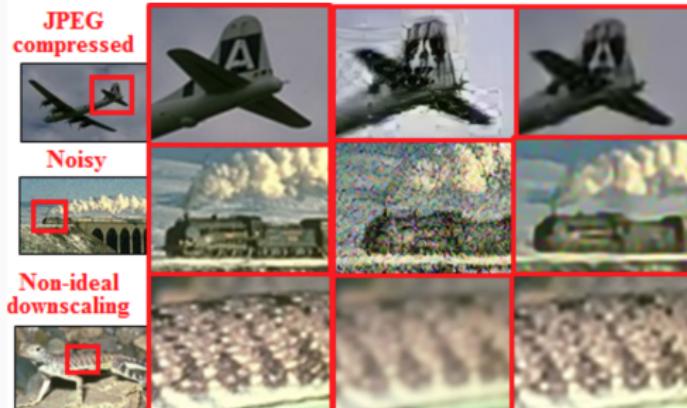
- Получается лучше в ситуациях, когда ядро неизвестно



Superresolution

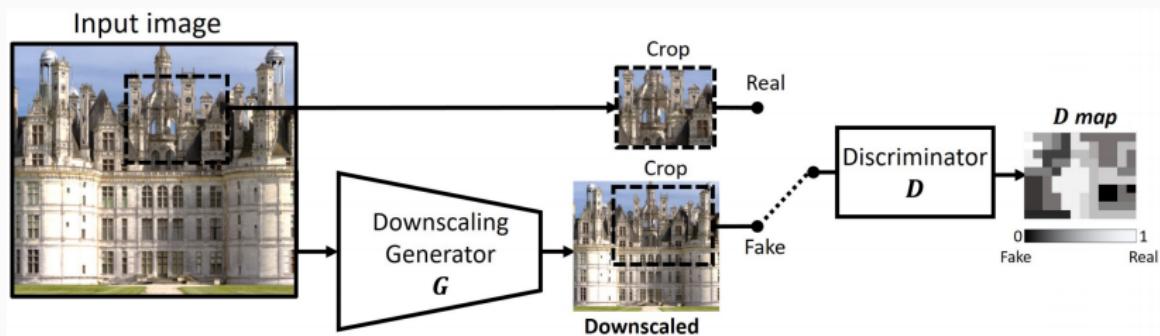
- И даже в идеальных ситуациях (с известным ядром) иногда может быть лучше, если есть повторяющиеся структуры, которые можно обучить:

ZSHC HSKRN CHKRVD HONSDCV OKHDNRC VHDONKUOSRC BODLCKZYHSROA HKGSCANDMPVESR PHEUSLZTJHCAZDI RAHCFOYZG	O S R C HSROA MPVESR IJHCAZDI RAHCFOYZG			
Ground Truth (PSNR, SSIM)	VDSR [9] (20.11, 0.9136)	EDSR+ [13] (25.29 / 0.9627)	ZSSR (ours) (25.68 / 0.9546)	



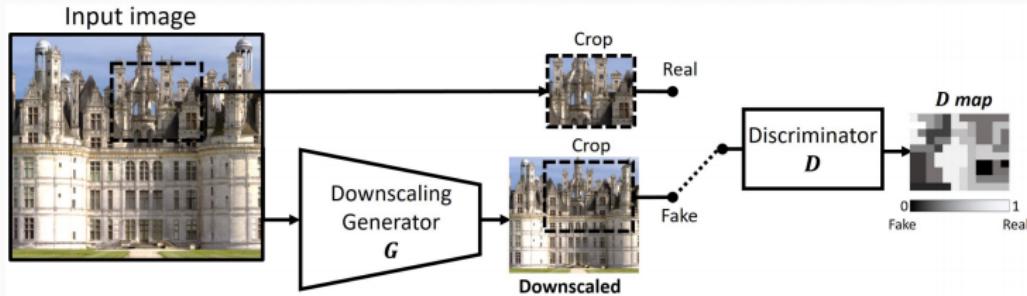
Superresolution

- А следующее развитие – это добавить сюда GAN
- KernelGAN (Bell-Kligler et al., 2019) обучает генератор делать downscaling так, чтобы кусочки картинки были неотличимы от исходных, а потом применяет это ядро фактически через ZSSR:

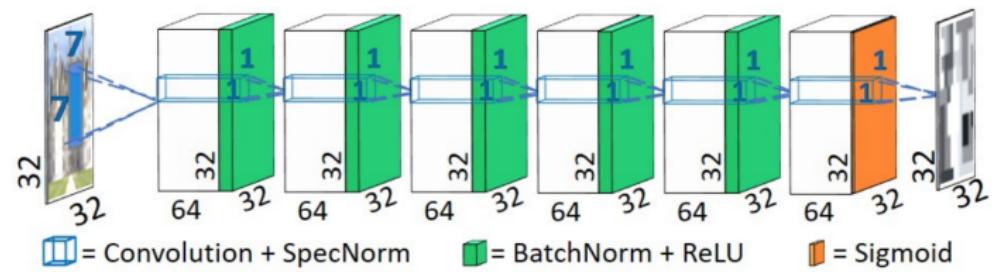


Superresolution

- Сети на самом деле довольно простые и маленькие: линейный генератор (чтобы был эквивалентен свёртке с ядром), LSGAN loss с L_1 -нормой плюс регуляризатор Ω

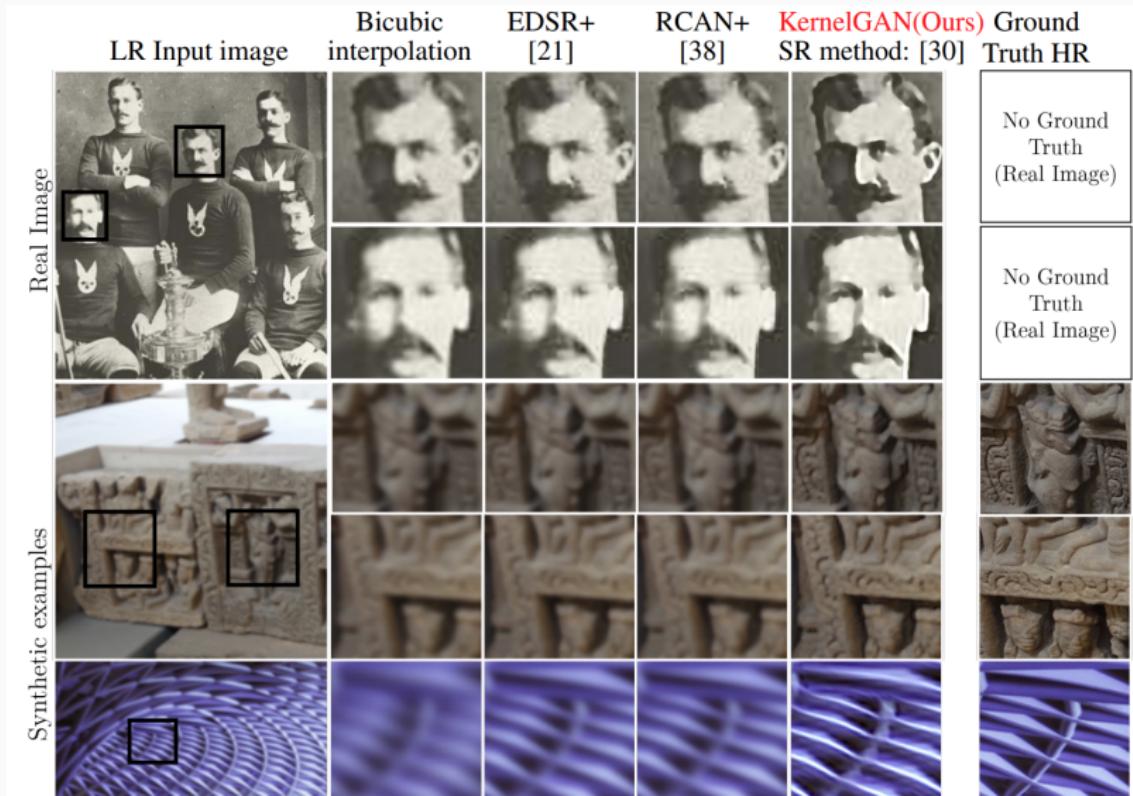


- Простой свёрточный дискриминатор:



Superresolution

- Получается хорошо:

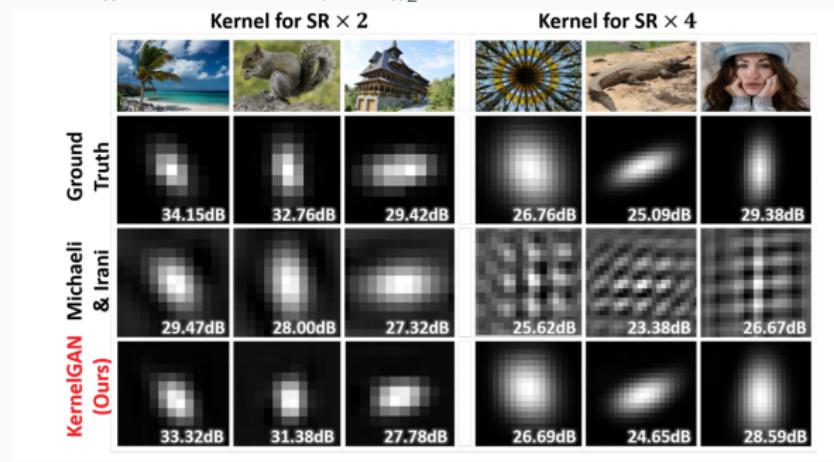


Superresolution

- Выделять ядро явно, кстати, полезно потому, что можно его регуляризовать; в KernelGAN:

$$\Omega = \alpha \mathcal{L}_{\text{sum-to-1}} + \beta \mathcal{L}_{\text{boundaries}} + \gamma \mathcal{L}_{\text{sparse}} + \delta \mathcal{L}_{\text{center}}, \text{ где}$$

- $\mathcal{L}_{\text{sum-to-1}} = \left| 1 - \sum_{ij} k_{ij} \right|$
- $\mathcal{L}_{\text{boundaries}} = \sum_{ij} |k_{ij} \cdot m_{ij}|$, где m растёт от центра к краю;
- $\mathcal{L}_{\text{sparse}} = \sum_{ij} |k_{ij}|^{1/2}$
- $\mathcal{L}_{\text{center}} = \left\| (x_0, y_0) - \frac{\sum_{ij} k_{ij} \cdot (i, j)}{\sum_{ij} k_{ij}} \right\|_2$



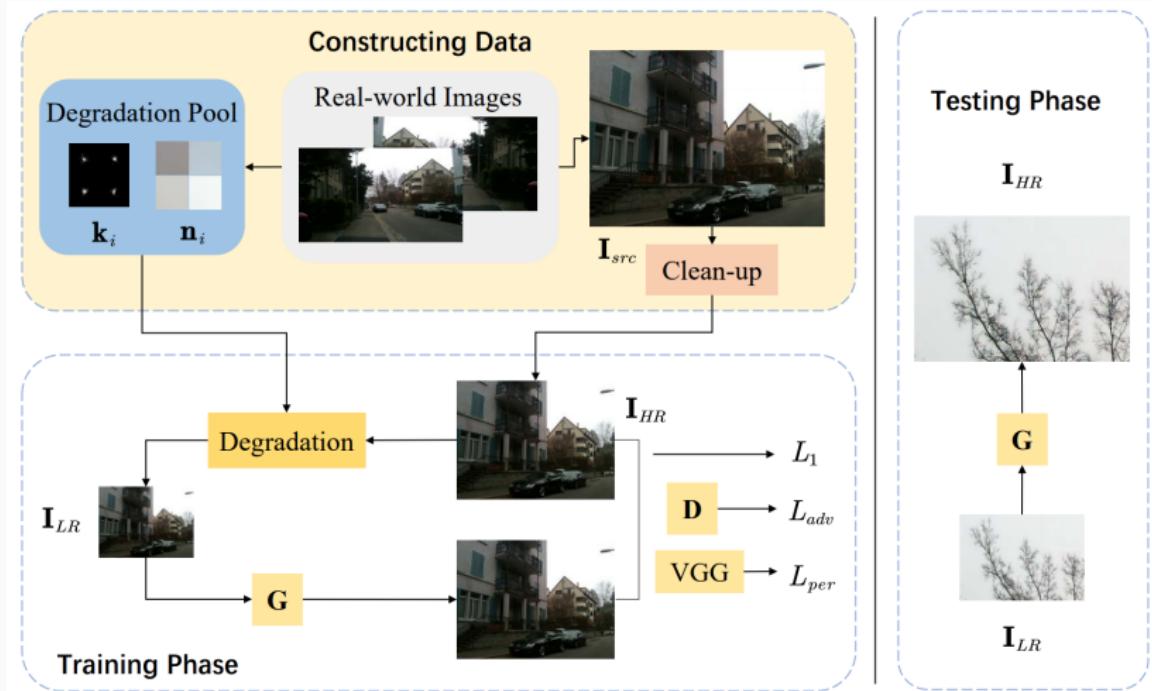
- KernelGAN хороший:
 - unsupervised
 - обучает хорошие ядра, причём для каждой картинки свои
 - давал наилучшие результаты на тот момент, когда был предложен
- Но и плохой:
 - очень медленный – надо, чтобы целое обучение сошлось для каждой картинки отдельно

Superresolution

- RealSR (Real-World Super-Resolution) – Ji et al., CVPR 2020:
 - идея в том, чтобы взять лучшее от обоих миров;
 - supervised SR отлично работает на идеальных датасетах, но на реальных картинках не очень, потому что ядро неизвестно;
 - KernelGAN обучает ядра, но это очень медленно и по отдельности на каждой картинке
 - так давайте обучим единую модель, но возьмём в неё много разных ядер!

Superresolution

- Общая идея:



Superresolution

- Т.е. деградация происходит при помощи ядра и шума:

$$I_{LR} = (I_{HR} * k) \downarrow_s + n$$

- Архитектуры по сути из ESRGAN, функция ошибки та же самая:

$$\mathcal{L} = \lambda_1 \mathcal{L}_1 + \lambda_{perc} \mathcal{L}_{perc} + \lambda_{adv} \mathcal{L}_{adv}$$

- Новое здесь только noise injection – собираем патчи шума, добавляем их как элемент деградации.

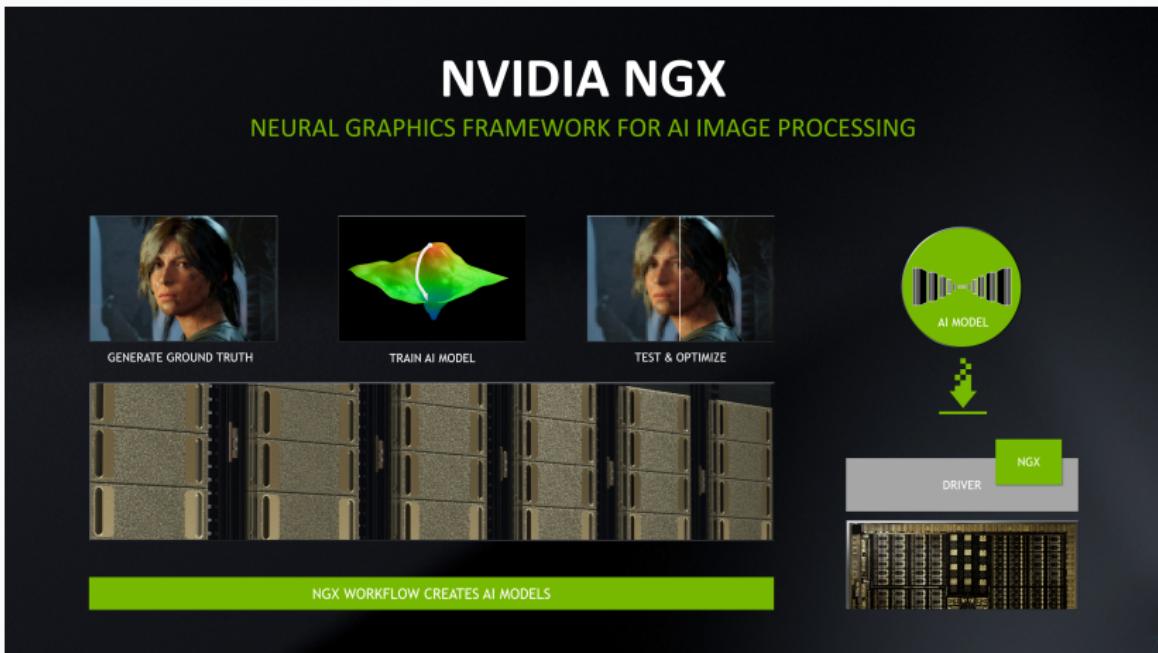
Superresolution

- Но получается прямо хорошо:



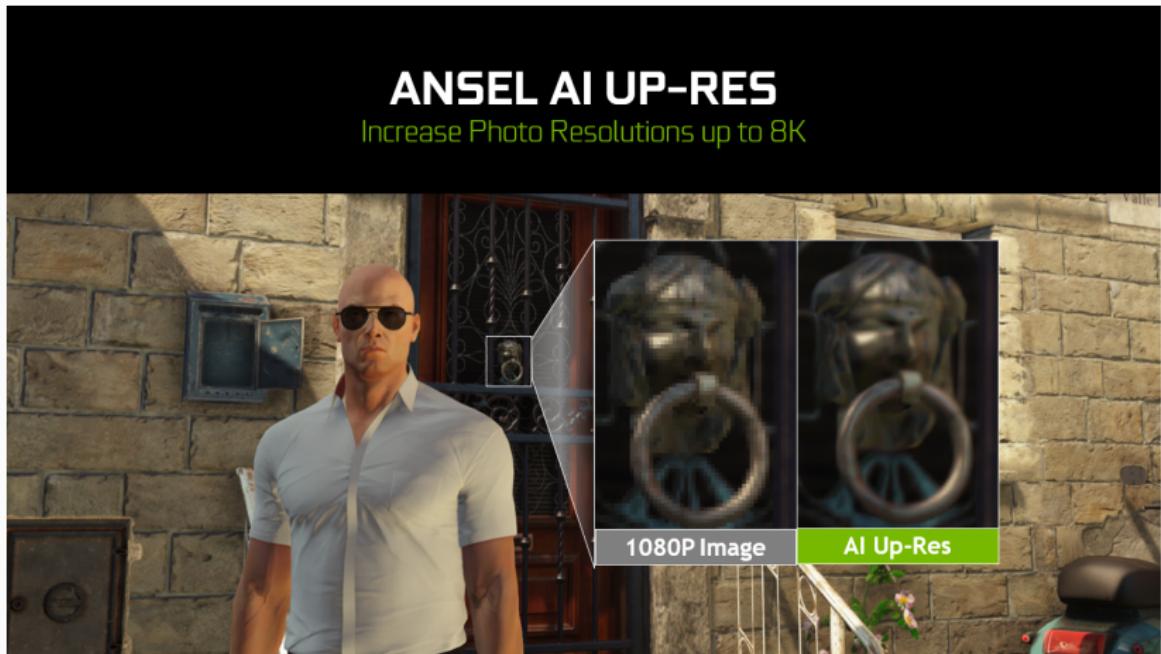
Superresolution

- NVIDIA DLSS (Deep Learning Super Sampling) – это не только и не столько superresolution, но интересное применение всего этого дела; первая версия вышла в 2018



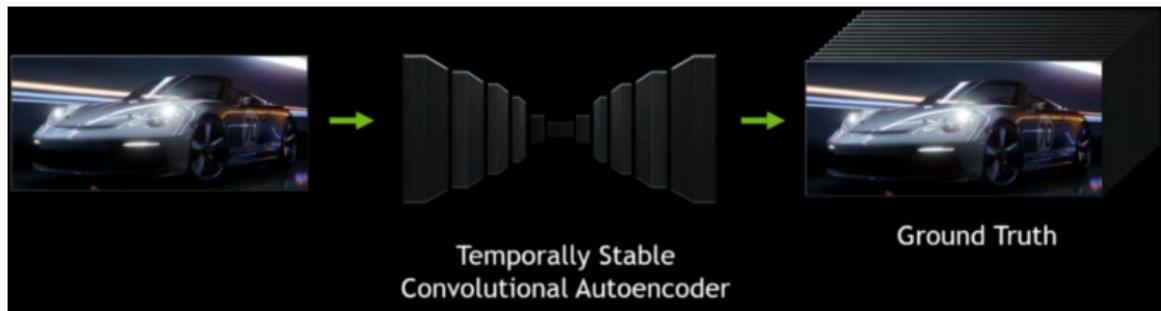
Superresolution

- Ещё раньше началось с NVIDIA Ansel, который делал крутые скриншоты в высоком разрешении; там уже был superresolution



Superresolution

- Первая версия DLSS – это свёрточный автокодировщик, обученный восстанавливать идеальные кадры, отрендеренные в высоком разрешении
- И, конечно, это non-blind SR, тут нет никакой камеры и можно обучать в реальных условиях



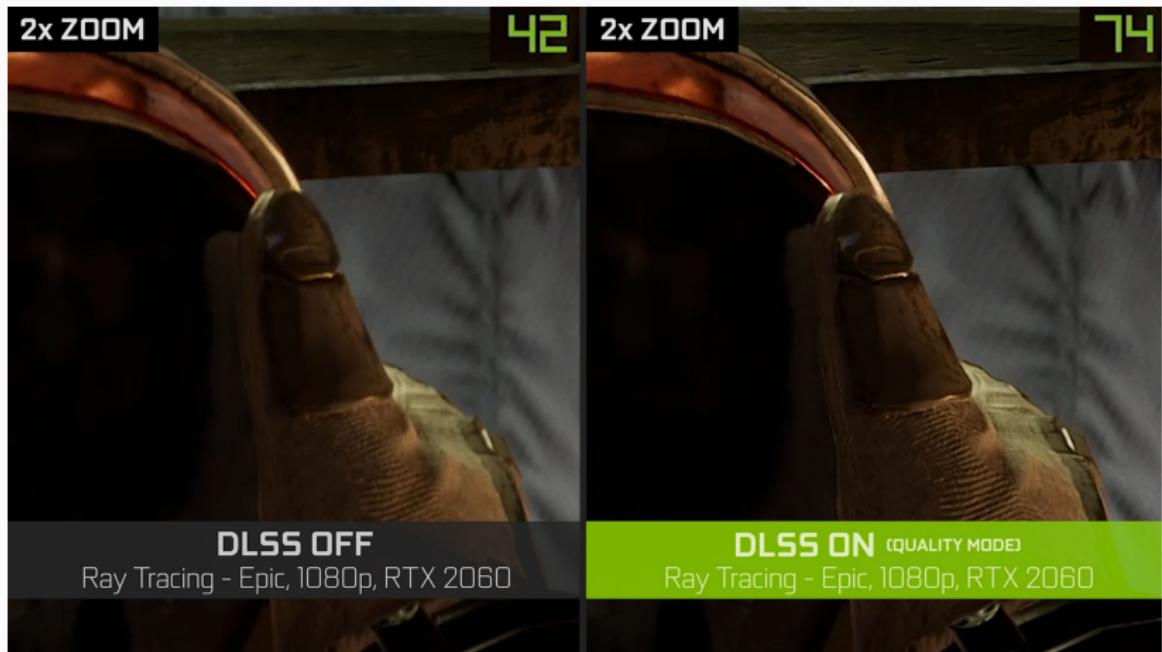
Superresolution

- Получалось хорошо, но обратите внимание, что DLSS в каждой игре вводился отдельно, т.е. модель нужно было обучать отдельно в каждом случае
- Так что это каждый раз была новость: Battlefield V, Metro Exodus, Final Fantasy XV...



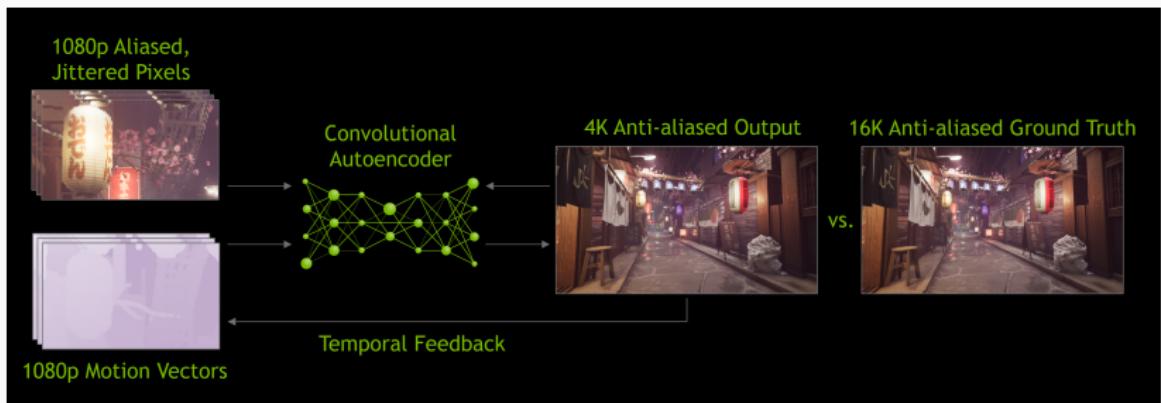
Superresolution

- Апрель 2020 – DLSS v 2.0



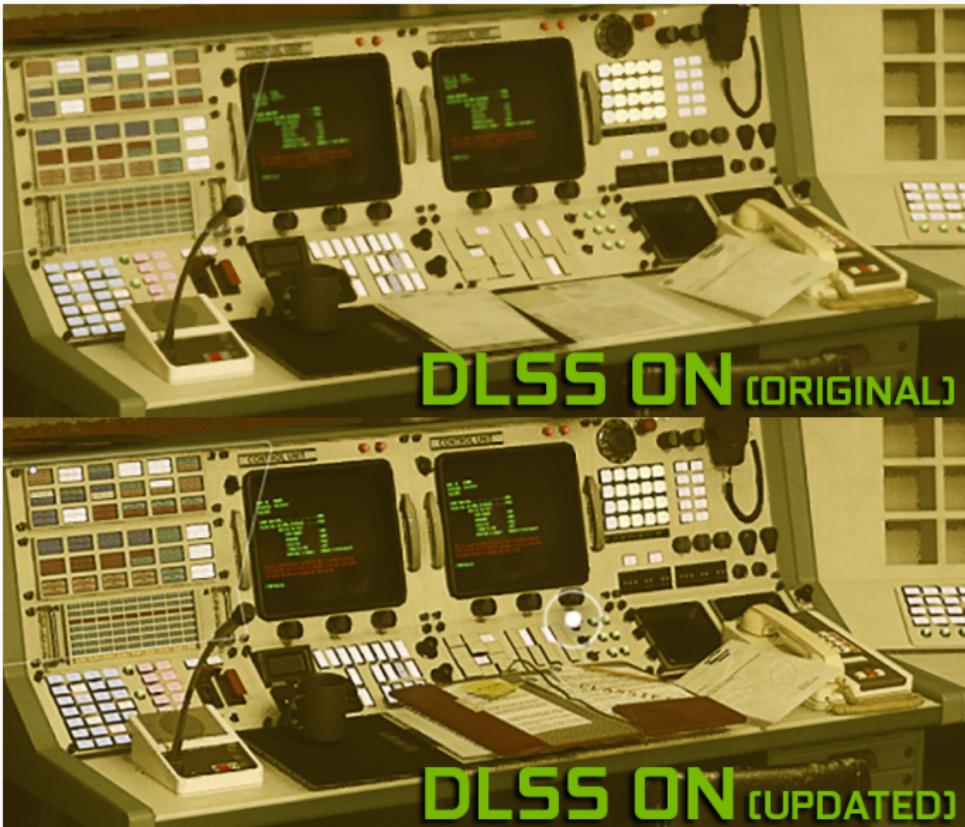
Superresolution

- Игровые движки могут дать не только картинку, но и motion vectors, т.е. то, куда двигаются сейчас объекты, и это тоже может помочь улучшить картинку



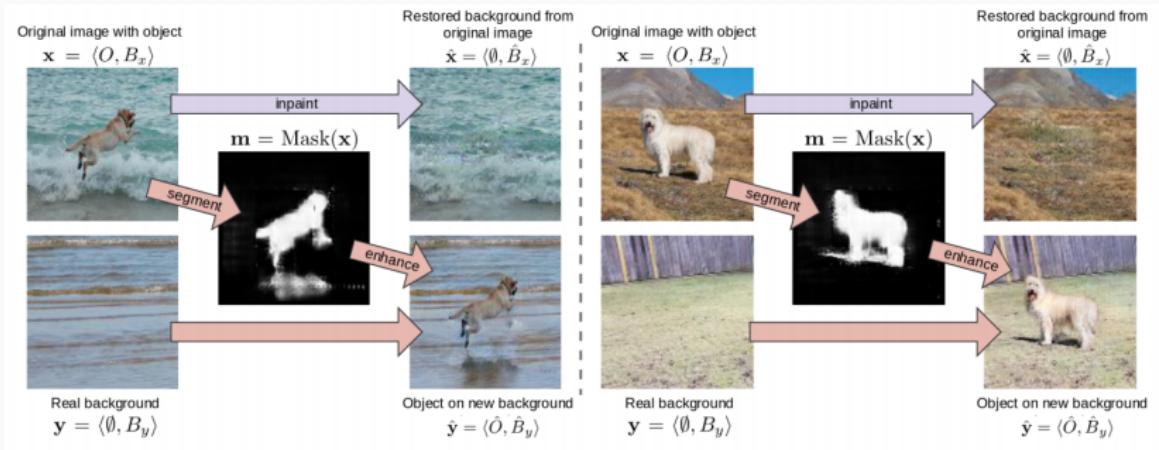
Superresolution

- Получается ещё лучше:



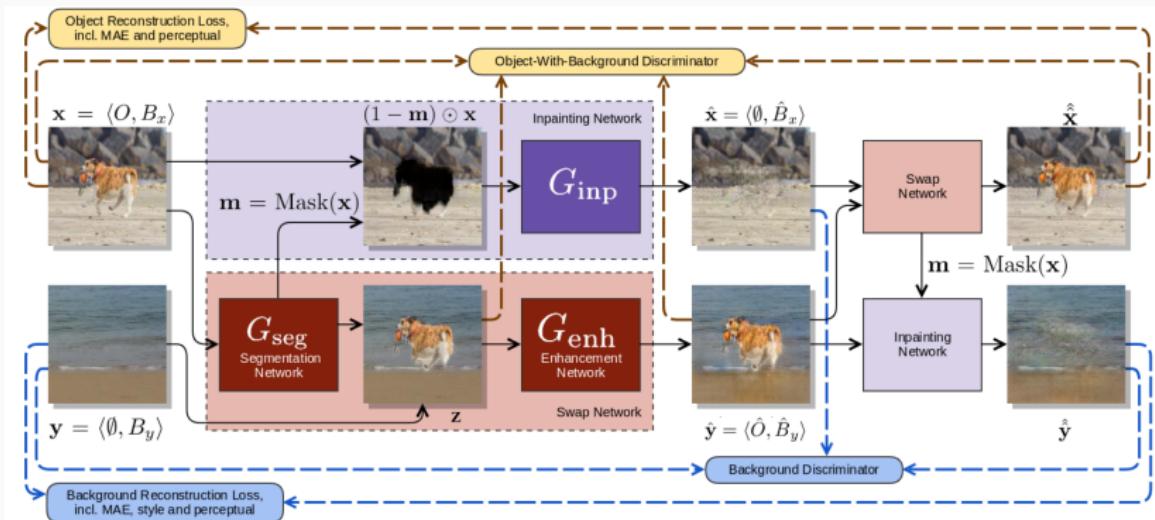
И ещё пара примеров

- SEIGAN (Ostyakov et al., 2018):



- Разбиваем задачу на три части:
 - **SEGMENT**: найти объект O на картинке $x = \langle O, B_x \rangle$, предсказав маску сегментации $m = \text{Mask}(x)$; по m можно построить грубый бленд как $z = m \odot x + (1 - m) \odot y$;
 - **ENHANCE**: по z и m построить улучшенную картинку $\hat{y} = \langle \hat{O}, \hat{B}_y \rangle$;
 - **INPAINT**: по $(1 - m) \odot x$ с удалённым m , восстановить фон $\hat{x} = \langle \emptyset, \hat{B}_x \rangle$.
- Зачем, кстати, фон восстанавливать?

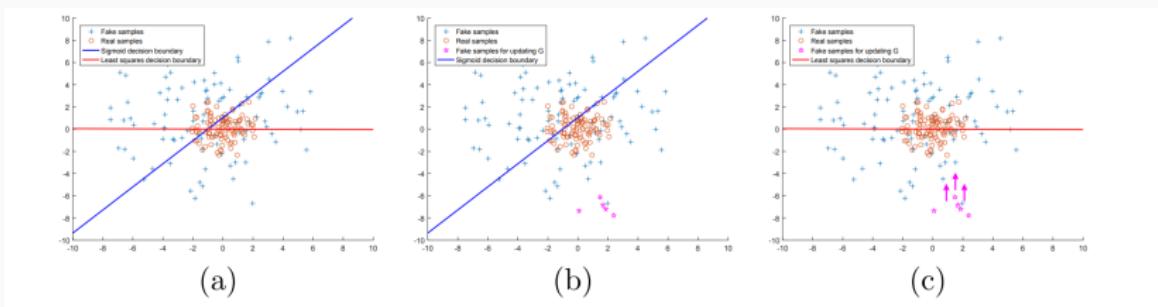
- Чтобы можно было добавить cycle consistency loss:



- В архитектуре пять сетей, три генератора и два дискrimинатора:
 - G_{seg} сегментирует: по картинке x предсказывает маску объекта $\text{Mask}(x)$;
 - G_{inp} делает inpainting: по $(1 - m) \odot x$ предсказывает $\hat{x} = \langle \emptyset, \hat{B}_x \rangle$;
 - G_{enh} улучшает картинки: по $z = m \odot x + (1 - m) \odot y$ предсказывает $\hat{y} = \langle \hat{O}, \hat{B}_y \rangle$;
 - D_{bg} – дискриминатор фона, различает картинки с фоном без объектов; выход $D_{\text{bg}}(x)$ должен быть близок к 1 для настоящих x и к 0 для поддельных x ;
 - D_{obj} – дискриминатор объектов, различает картинки с объектами; $D_{\text{obj}}(x)$ должен быть близок к 1 для настоящих x и к 0 для поддельных x .

- Adversarial background loss – функция потерь MSE, как в Least Squares GAN (LSGAN):

$$\begin{aligned}\mathcal{L}_{\text{inp}}^{\text{GAN}} &= (1 - D_{\text{bg}}(\hat{\mathbf{x}}))^2, \quad \mathcal{L}_{\text{inp}2}^{\text{GAN}} = (1 - D_{\text{bg}}(\hat{\mathbf{y}}))^2, \\ \mathcal{L}_{\text{bg}}^{\text{GAN}} &= \lambda_1 \mathcal{L}_{\text{inp}}^{\text{GAN}} + \lambda_2 \mathcal{L}_{\text{inp}2}^{\text{GAN}},\end{aligned}$$



- Background reconstruction loss $\mathcal{L}_{\text{bg}}^{\text{rec}}$ сохраняет информацию об исходном фоне B_x ; основан на texture loss:

$$\begin{aligned}\mathcal{L}_{\text{bg}}^{\text{texture}} &= |\text{Gram}(\text{VGG}_1(y)) - \text{Gram}(\text{VGG}_1(\hat{y}))|, \\ \mathcal{L}_{\text{bg}}^{\text{perc}} &= |\text{VGG}_2(y) - \text{VGG}_2(\hat{y})|, \quad \mathcal{L}_{\text{bg}}^{\text{MAE}} = |y - \hat{y}|, \\ \mathcal{L}_{\text{bg}}^{\text{rec}} &= \lambda_3 \mathcal{L}_{\text{bg}}^{\text{texture}} + \lambda_4 \mathcal{L}_{\text{bg}}^{\text{perc}} + \lambda_5 \mathcal{L}_{\text{bg}}^{\text{MAE}},\end{aligned}$$

где $\text{VGG}_1(y)$ – признаки после первых 5 свёрточных слоёв VGG-19, $\text{VGG}_2(y)$ – после последних 5 свёрточных слоёв, и $\text{Gram}(A)_{ij} = \sum_k A_{ik}A_{jk}$ – матрица Грама.

- Object reconstruction loss $\mathcal{L}_{\text{obj}}^{\text{rec}}$ – реконструкция объекта, сумма perceptual loss и MAE между исходным $\mathbf{x} = \langle O, B_x \rangle$ и циклом $\hat{\mathbf{x}} = G_{\text{enh}}(G_{\text{seg}}(\hat{\mathbf{y}}) \odot \hat{\mathbf{y}} + (1 - G_{\text{seg}}(\hat{\mathbf{y}})) \odot \hat{\mathbf{x}})$:

$$\begin{aligned}\mathcal{L}_{\text{obj}}^{\text{perc}} &= |\text{VGG}_2(\mathbf{x}) - \text{VGG}_2(\hat{\mathbf{x}})|, \quad \mathcal{L}_{\text{obj}}^{\text{MAE}} = |\mathbf{x} - \hat{\mathbf{x}}|, \\ \mathcal{L}_{\text{obj}}^{\text{rec}} &= \lambda_6 \mathcal{L}_{\text{obj}}^{\text{perc}} + \lambda_7 \mathcal{L}_{\text{obj}}^{\text{MAE}},\end{aligned}$$

где $\hat{\mathbf{y}} = G_{\text{enh}}(z)$ и $\hat{\mathbf{x}} = G_{\text{inp}}((1 - G_{\text{seg}}(\mathbf{x})) \odot \mathbf{x})$, т.е., $\hat{\mathbf{x}}$ – это результат двойного применения swap network.

- Adversarial object loss $\mathcal{L}_{\text{obj}}^{\text{GAN}}$ увеличивает правдоподобность $\hat{\mathbf{y}} = \langle \hat{O}, \hat{B}_y \rangle$ и z ; LSGAN с дискриминатором D_{obj} :

$$\begin{aligned}\mathcal{L}_{\text{coarse}}^{\text{GAN}} &= (1 - D_{\text{obj}}(z))^2, \quad \mathcal{L}_{\text{enh}}^{\text{GAN}} = (1 - D_{\text{obj}}(\hat{\mathbf{y}}))^2, \\ \mathcal{L}_{\text{obj}}^{\text{GAN}} &= \lambda_8 \mathcal{L}_{\text{coarse}}^{\text{GAN}} + \lambda_9 \mathcal{L}_{\text{enh}}^{\text{GAN}}.\end{aligned}$$

- *Identity loss* \mathcal{L}^{id} , as in CycleGAN.
 - object enhancement identity loss $\mathcal{L}_{\text{obj}}^{\text{id}}$ делает результат enhancement network G_{enh} близким к тождественному на настоящих картинках;
 - background identity loss $\mathcal{L}_{\text{bg}}^{\text{id}}$ требует, чтобы процедура cut-and-inpaint оставляла без изменений картинки без объектов:

$$\mathcal{L}_{\text{bg}}^{\text{id}} = |G_{\text{inp}}((1 - G_{\text{seg}}(\mathbf{y})) \odot \mathbf{y}) - \mathbf{y}|,$$

$$\mathcal{L}_{\text{obj}}^{\text{id}} = |G_{\text{enh}}(\mathbf{x}) - \mathbf{x}|, \quad \mathcal{L}^{\text{id}} = \lambda_{10}\mathcal{L}_{\text{obj}}^{\text{id}} + \lambda_{11}\mathcal{L}_{\text{bg}}^{\text{id}}.$$

- ИТОГО:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{inp}}^{\text{GAN}} + \lambda_2 \mathcal{L}_{\text{inp2}}^{\text{GAN}} + \lambda_3 \mathcal{L}_{\text{bg}}^{\text{texture}} + \lambda_4 \mathcal{L}_{\text{bg}}^{\text{perc}} + \lambda_5 \mathcal{L}_{\text{bg}}^{\text{MAE}} + \lambda_6 \mathcal{L}_{\text{obj}}^{\text{perc}} + \lambda_7 \mathcal{L}_{\text{obj}}^{\text{MAE}} + \lambda_8 \mathcal{L}_{\text{coarse}}^{\text{GAN}} + \lambda_9 \mathcal{L}_{\text{enh}}^{\text{GAN}} + \lambda_{10} \mathcal{L}_{\text{obj}}^{\text{id}} + \lambda_{11} \mathcal{L}_{\text{bg}}^{\text{id}}, \text{ где}$$

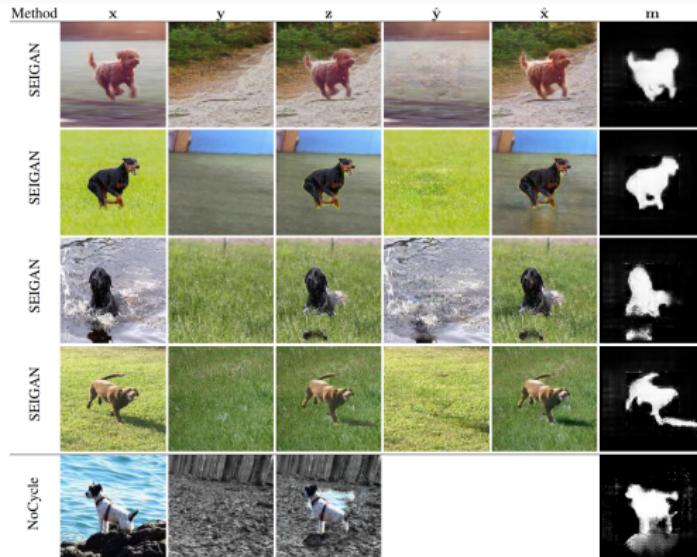
$$\begin{aligned}
 \mathcal{L}_{\text{inp}}^{\text{GAN}} &= (1 - D_{\text{bg}}(\hat{x}))^2, & \mathcal{L}_{\text{inp2}}^{\text{GAN}} &= (1 - D_{\text{bg}}(\hat{y}))^2, & \mathcal{L}_{\text{bg}}^{\text{id}} &= |G_{\text{inp}}((1 - G_{\text{seg}}(y)) \odot y) - y|, \\
 \mathcal{L}_{\text{enh}}^{\text{GAN}} &= (1 - D_{\text{obj}}(\hat{y}))^2, & \mathcal{L}_{\text{bg}}^{\text{MAE}} &= |y - \hat{y}|, & \mathcal{L}_{\text{obj}}^{\text{perc}} &= |VGG_2(x) - VGG_2(\hat{x})|, \\
 \mathcal{L}_{\text{obj}}^{\text{MAE}} &= |x - \hat{x}|, & \mathcal{L}_{\text{coarse}}^{\text{GAN}} &= (1 - D_{\text{obj}}(z))^2, & \mathcal{L}_{\text{bg}}^{\text{perc}} &= |VGG_2(y) - VGG_2(\hat{y})|, \\
 \mathcal{L}_{\text{obj}}^{\text{id}} &= |G_{\text{enh}}(x) - x|, & \mathcal{L}_{\text{bg}}^{\text{texture}} &= |\text{Gram}(VGG_1(y)) - \text{Gram}(VGG_1(\hat{y}))|.
 \end{aligned}$$

- Discriminator losses:

$$\mathcal{L}_{\text{bg}}^{\text{disc}} = (1 - D_{\text{bg}}(y))^2 + \frac{D_{\text{bg}}(\hat{x})^2 + D_{\text{bg}}(\hat{y})^2}{2},$$

$$\mathcal{L}_{\text{obj}}^{\text{disc}} = (1 - D_{\text{obj}}(x))^2 + \frac{D_{\text{obj}}(\hat{y})^2 + D_{\text{obj}}(z)^2}{2}.$$

- Итого получается:



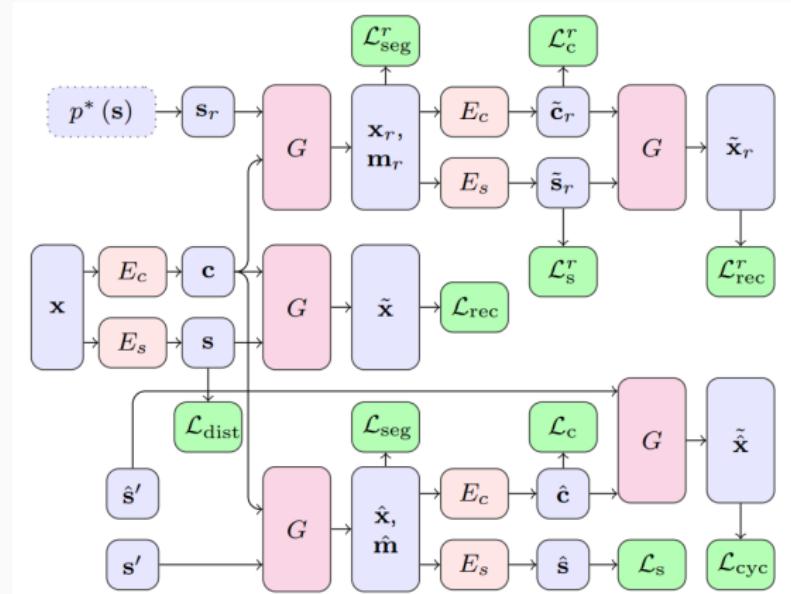
High-resolution daytime translation

- (Anokhin et al., 2020): HiDT – перенос стиля (смена времени суток) для больших ландшафтов



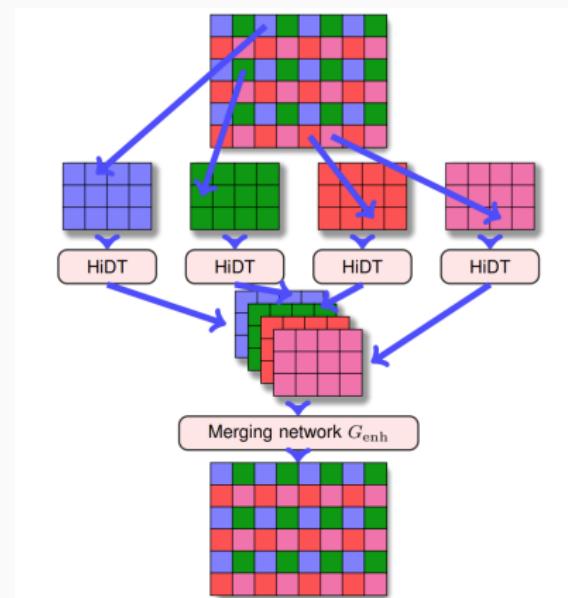
High-resolution daytime translation

- Базовый смысл в том, чтобы разделить картинку на контент и стиль, а потом поменять их местами
- Но этот смысл приходится выражать довольно хитро...



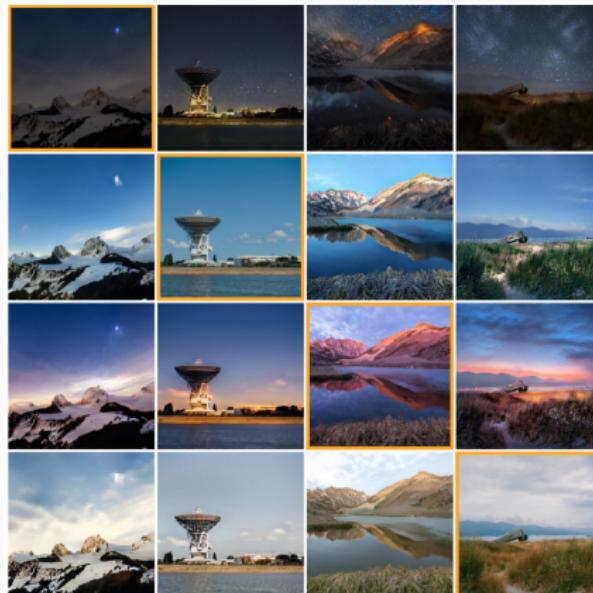
High-resolution daytime translation

- Плюс есть enhancement до более высокого разрешения, потому что иначе в HD не особенно работает



High-resolution daytime translation

- Но в итоге получается хорошо:



Спасибо!

Спасибо за внимание!

