

Лаборатория компьютерной
графики и мультимедиа
ВМК МГУ имени М.В. Ломоносова

Курс «Введение в компьютерное зрение
и глубокое обучение»

Лекция №8
«Перенос стиля и синтез изображений»

Антон Конушин

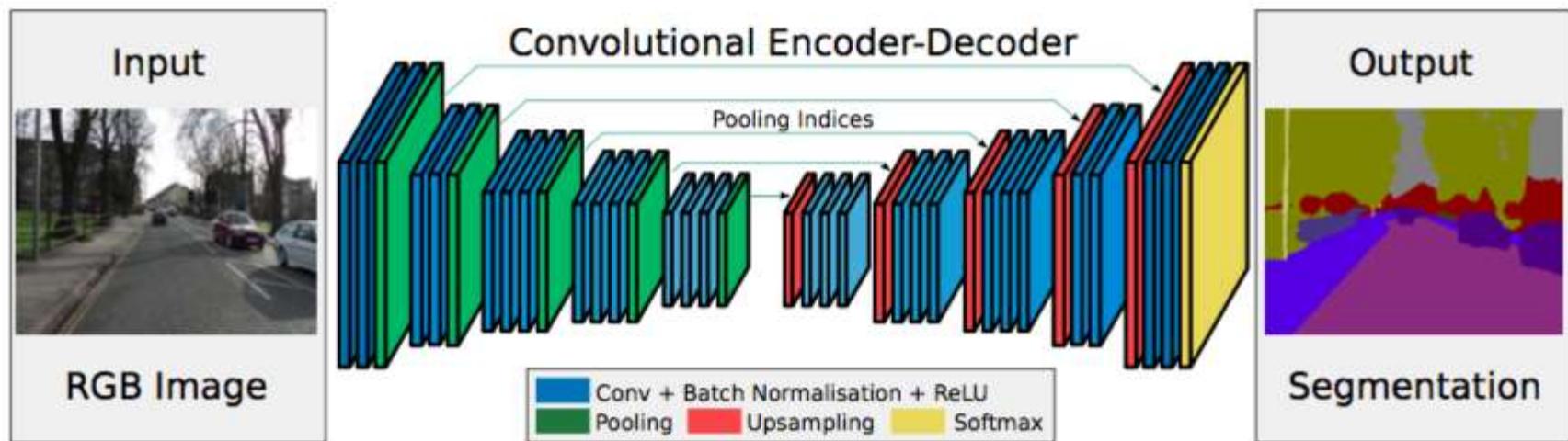
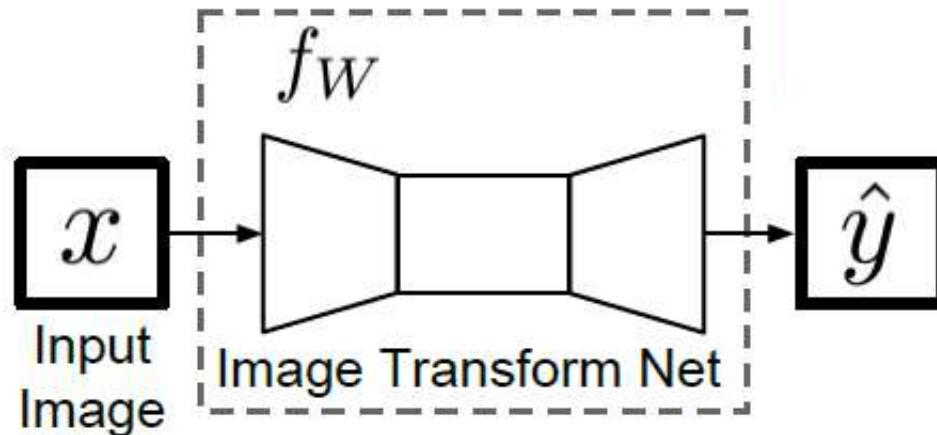
Заведующий лабораторией компьютерной графики и мультимедиа
ВМК МГУ

22 апреля 2019 года

Модификация изображений



Примеры решения - преобразователи



Перенос стиля изображения



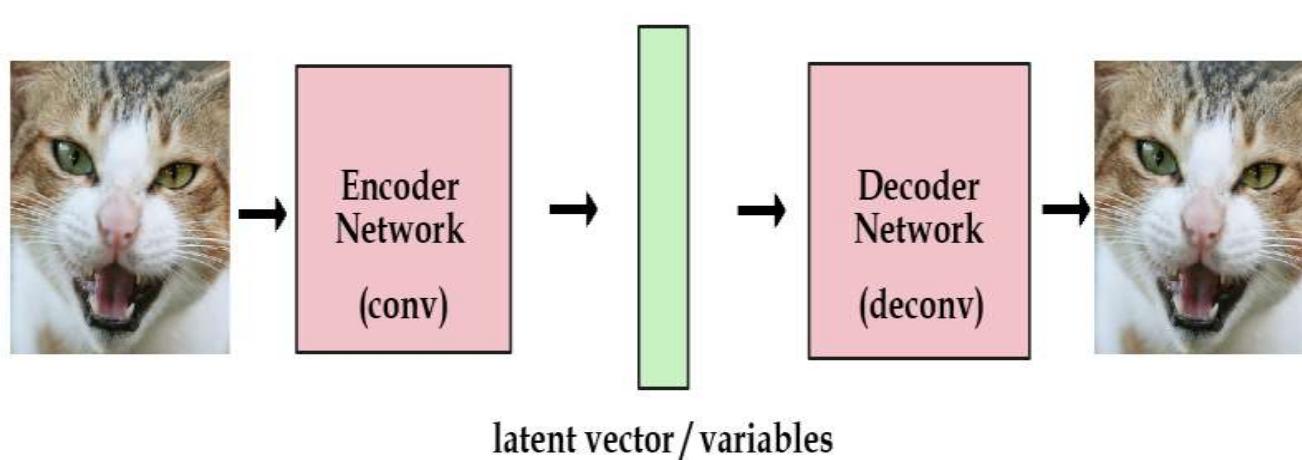
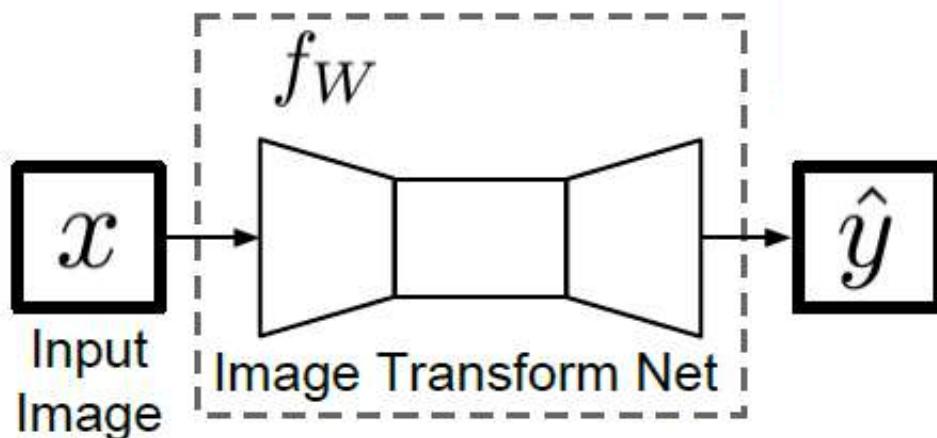
+



=



Подходы к решению



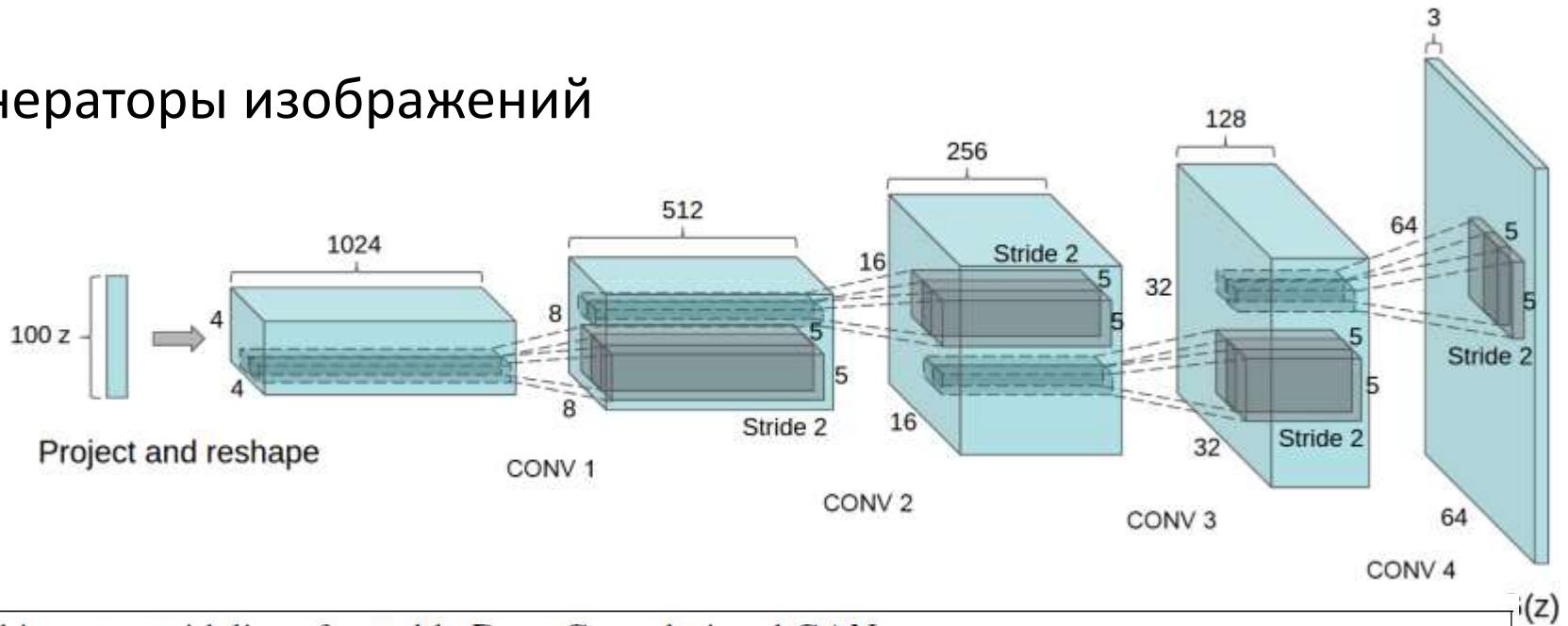
Синтез изображений



- Хотим сгенерировать случайное изображение с заданными свойствами
- В простом варианте – заданного класса (комнаты в квартире)
- Хотим использовать нейросети для этого

Пример решения - DCGAN

Генераторы изображений

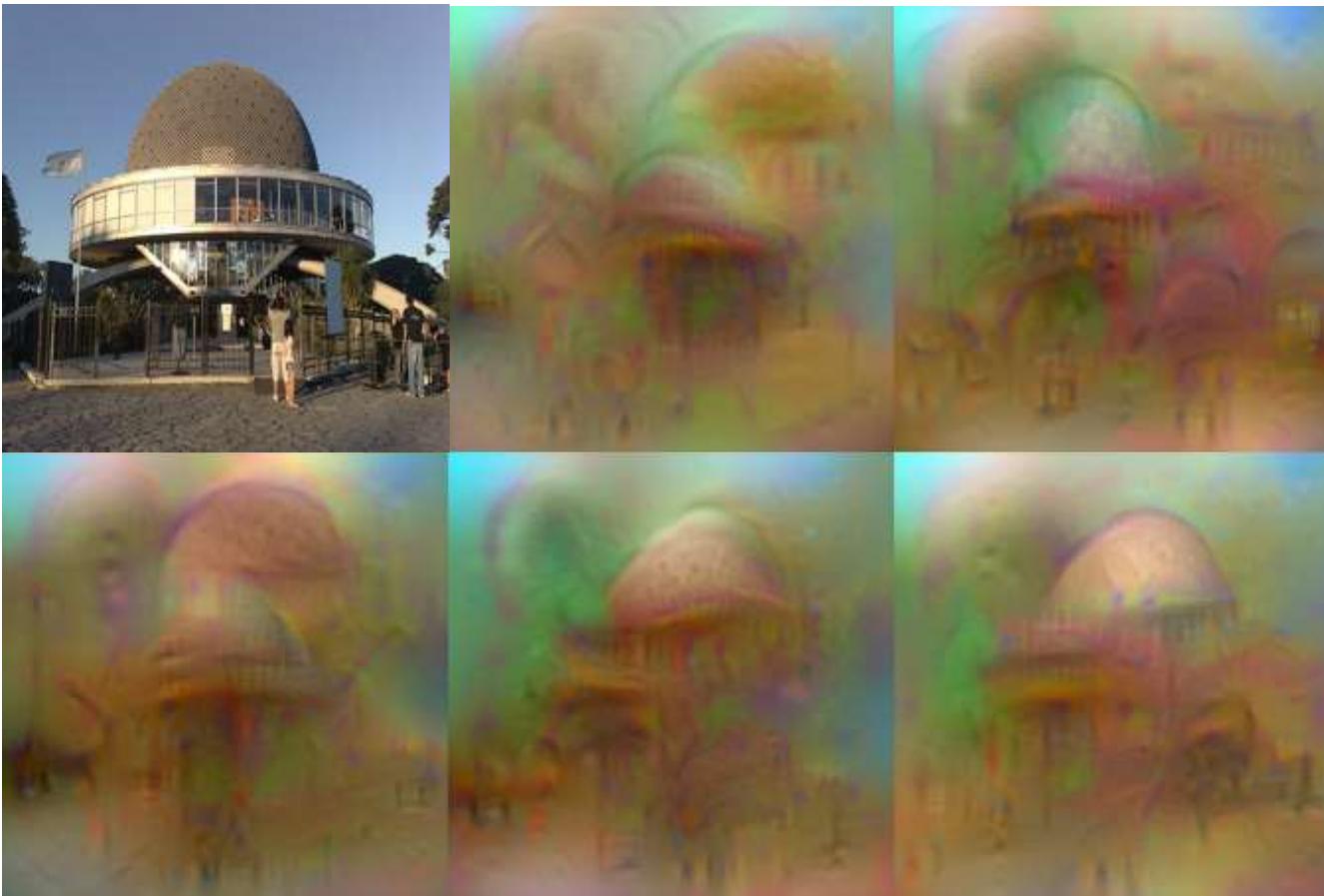


Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Визуализация вектор-признака

Найдём такое изображение x , которое даёт такой же вектор-признак $\Phi_0 = \Phi(x_0)$ от изображения x_0



Mahendran and Vedaldi. Understanding Deep Image Representations by Inverting Them, 2014

Визуализация изображения по выходам

Процедура поиска:

- Инициализируем x белым шумом
- Будем оптимизировать следующий функционал:

$$\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^{H \times W \times C}} \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x}) \quad R(x) - \text{регуляризатор}$$

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

Оптимизируем градиентным спуском

Реконструкция

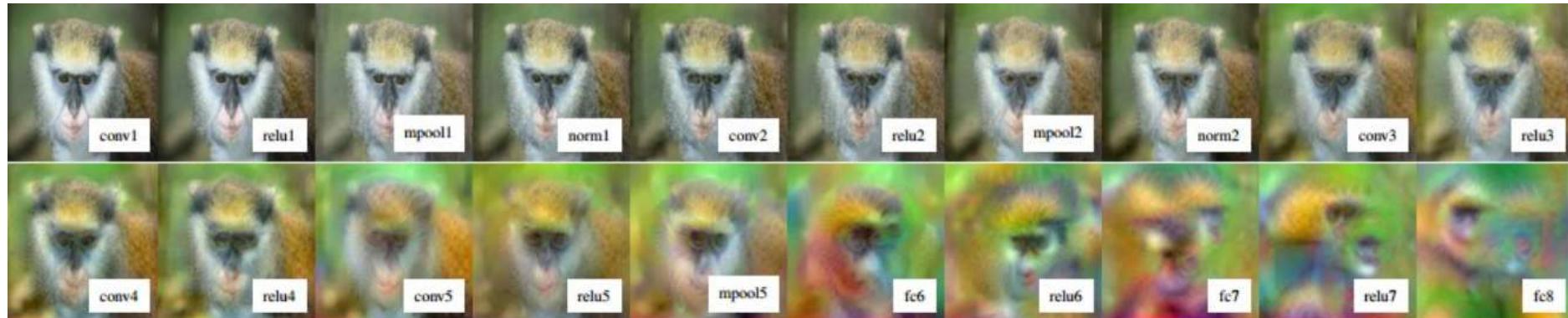
С последнего свёрточного слоя



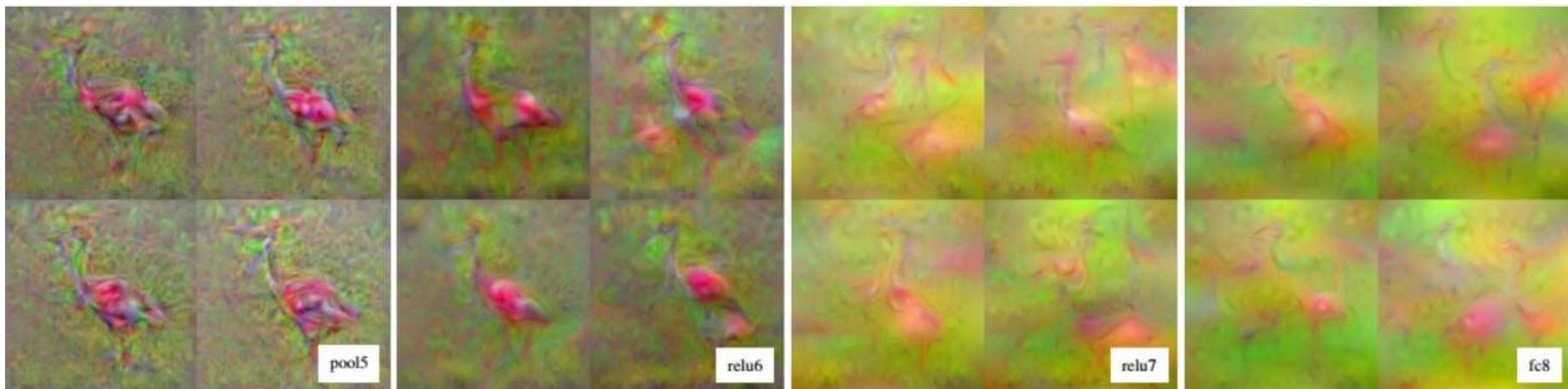
Пространственная информация во многом сохраняется

Реконструкция

С разных уровней



Множественные реконструкции (разные изображения с одинаковыми признаками)



Применим для переноса стиля



+



=



Простейшие методы переноса стиля

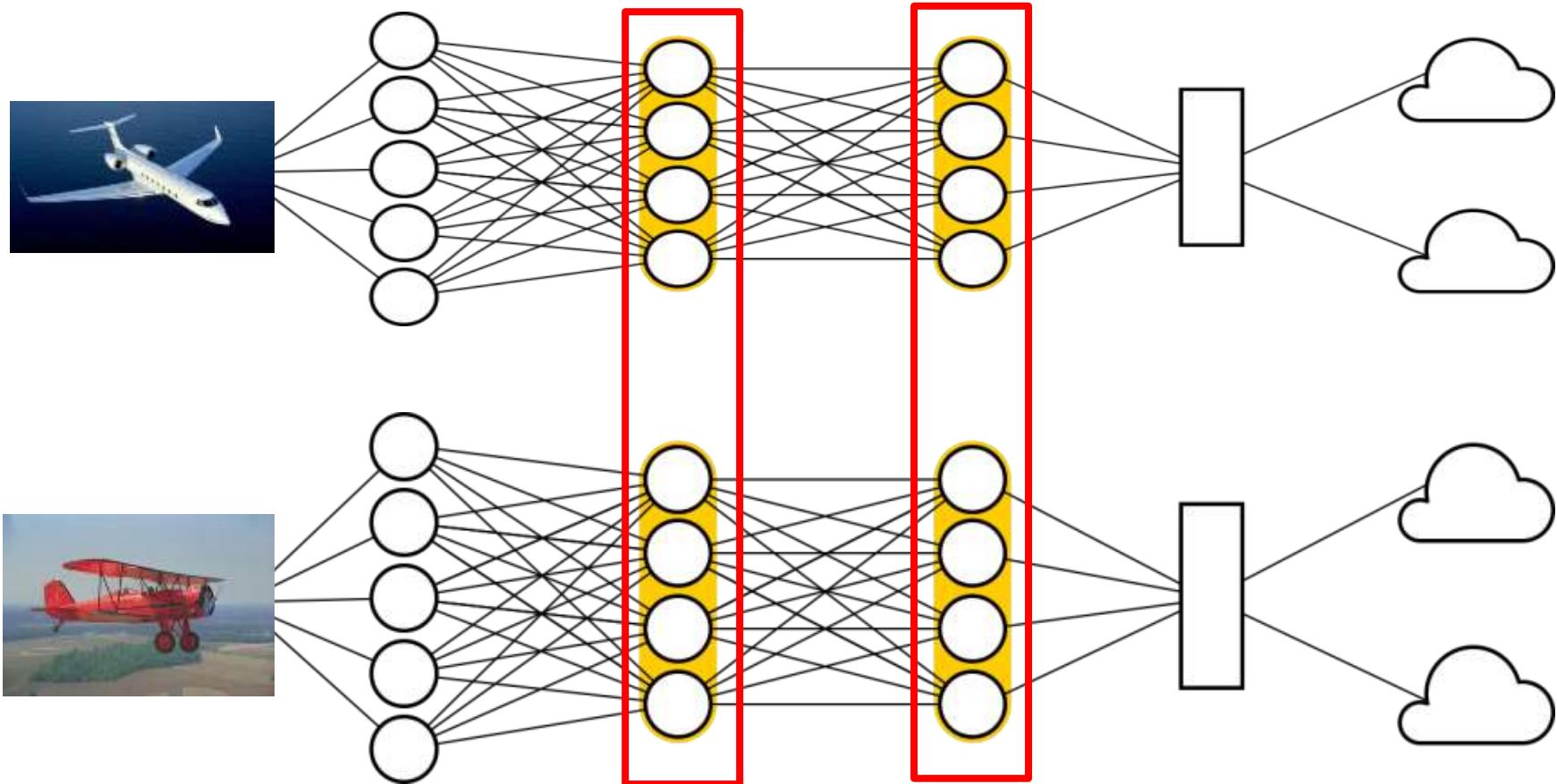
Базовые методы переноса цвета работают со свойствами всей картинки целиком. Пример такого метода – приравнивание среднего и дисперсии по каждому каналу в Lab



Основная идея – модификация изображения таким образом, чтобы по части признаков оно было похоже на исходное (содержание), а часть – на целевое (стиль)

Как описать похожесть по содержанию?

Сравнение нейросетевых признаков



Perception loss

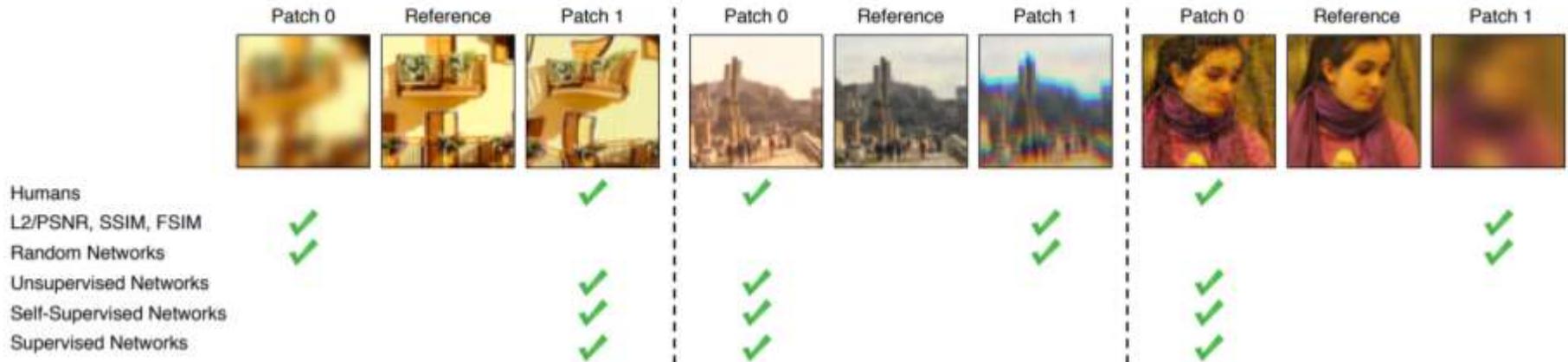


Figure 1: **Which patch (left or right) is “closer” to the middle patch in these examples?** In each case, the traditional metrics (L2/PSNR, SSIM, FSIM) disagree with human judgments. But deep networks, even across architectures (SqueezeNet [20], AlexNet [27], VGG [52]) and supervision type (supervised [47], self-supervised [13, 40, 43, 64], and even unsupervised [26]), provide an *emergent embedding* which agrees surprisingly well with humans. We further calibrate existing deep embeddings on a large-scale database of perceptual judgments; models and data can be found at <https://www.github.com/richzhang/PerceptualSimilarity>.

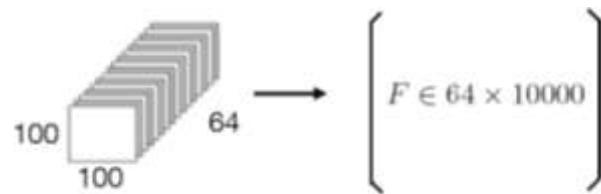
Описание стиля



- За описание «стиля» можно взять корреляцию откликов разных фильтров по всему изображению
- Можно вычислить по признакам первых слоёв «стиль» и попробовать реконструировать изображение с тем же стилем

Реконструкция стиля

- Стиль можно описать корреляцией откликов фильтров, записав матрицу Грама $G^l \in \mathbb{R}^{N_l \times N_l}$
- Где G_{ij}^l вычисляется как скалярное произведение откликов i-го и j-го фильтров:


$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$$

- Генерируем изображения стиля, минимизируя среднеквадратичную разницу между матрицами Грама исходного изображения G и сгенерированного изображения A (или суммами по L первых слоёв)

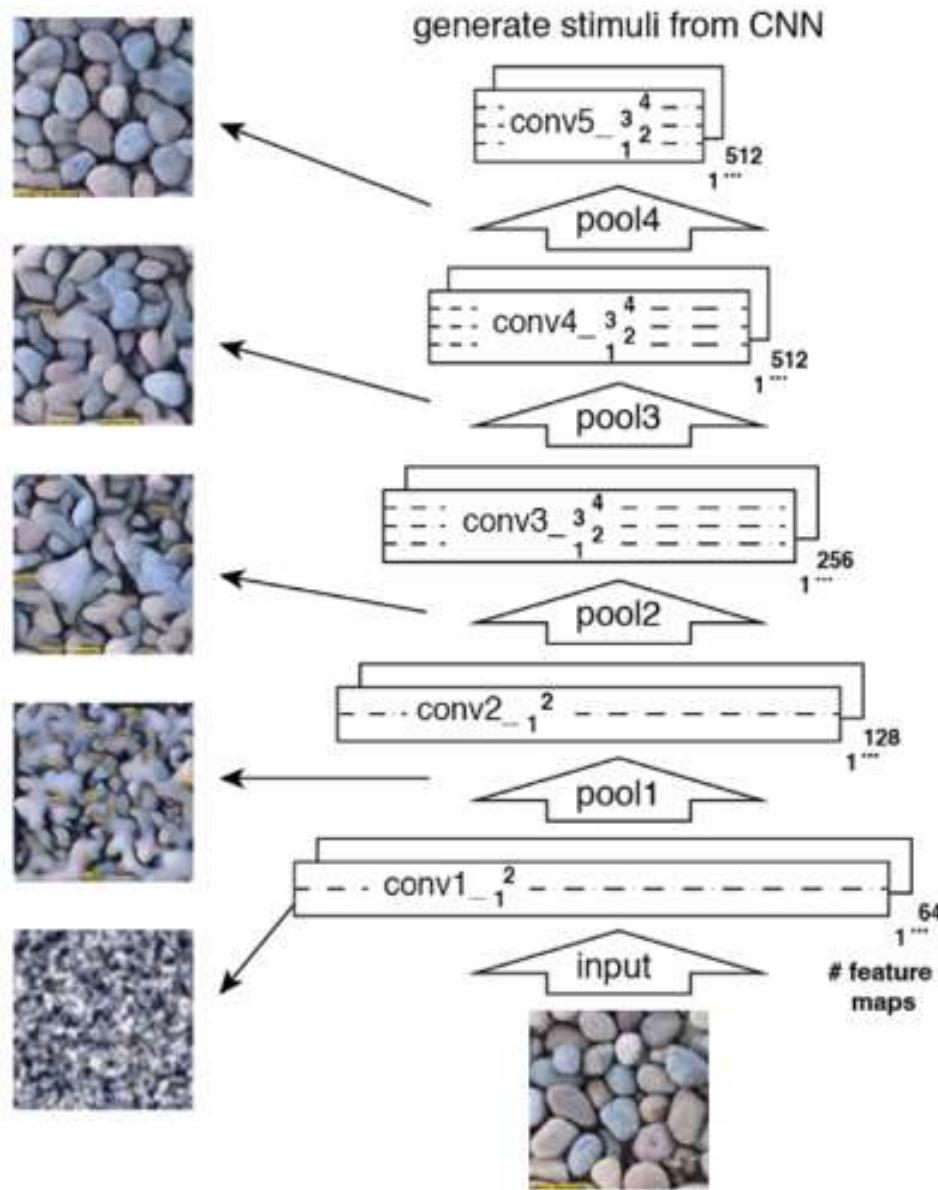
$$E_l = \frac{1}{Norm} \sum_{i,j} (G_{ij}^l - A_{i,j}^l)^2 \quad \text{Cost for style reconstruction}$$

$$Loss_{style} = \sum_{l=0}^L w_l E_l \quad \text{Accumulate cost for lower layers}$$

A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients. Portilla, J., & Simoncelli, E. P. *Int. J. Comput. Vis.* 40, 49-70 (2000)

Texture Synthesis and The Controlled Generation of Natural Stimuli Using Convolutional Neural Networks. Gatys, L. A., Ecker, A. S., & Bethge, M. *NIPS* (2015)

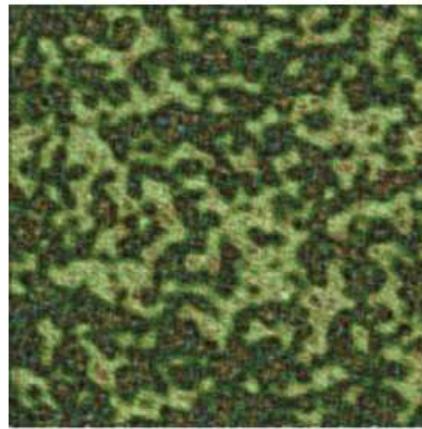
Реконструкция текстур



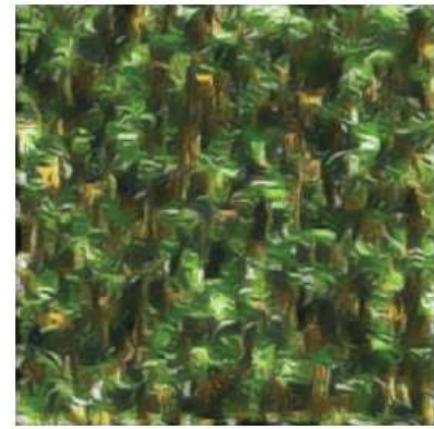
Пример реконструкции стиля



Original



Up to Conv1_1 layer



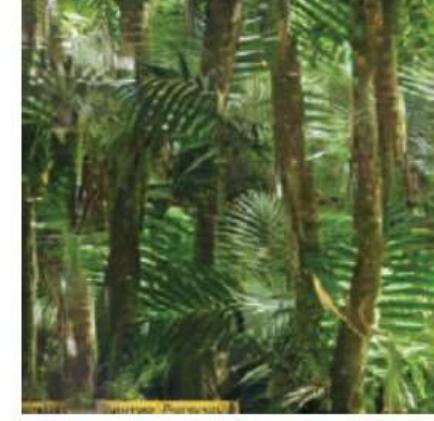
Up to Pool1 layer



Up to Pool2 layer



Up to Pool3 layer



Up to Pool4 layer

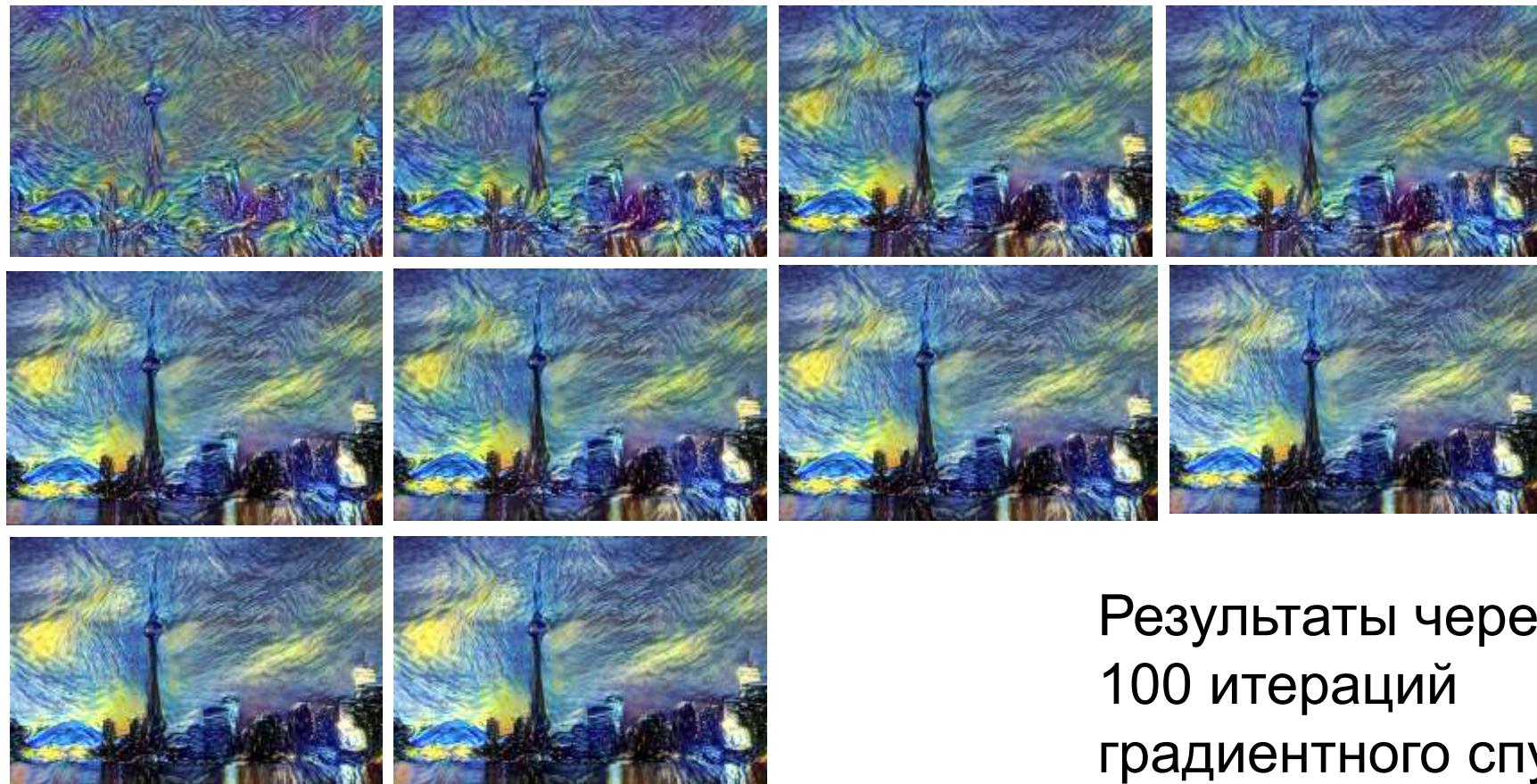
Ключевое наблюдение

- Содержание изображение и стиль оказываются разделимы
- Верхние слои целиком больше описывают содержание изображения
- Корреляция карт нижних слоёв – стиль изображения
- Можем сгенерировать изображения, начав с белого шума, минимизировав градиентным спуском функционал:

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." *arXiv preprint arXiv:1508.06576* (2015).

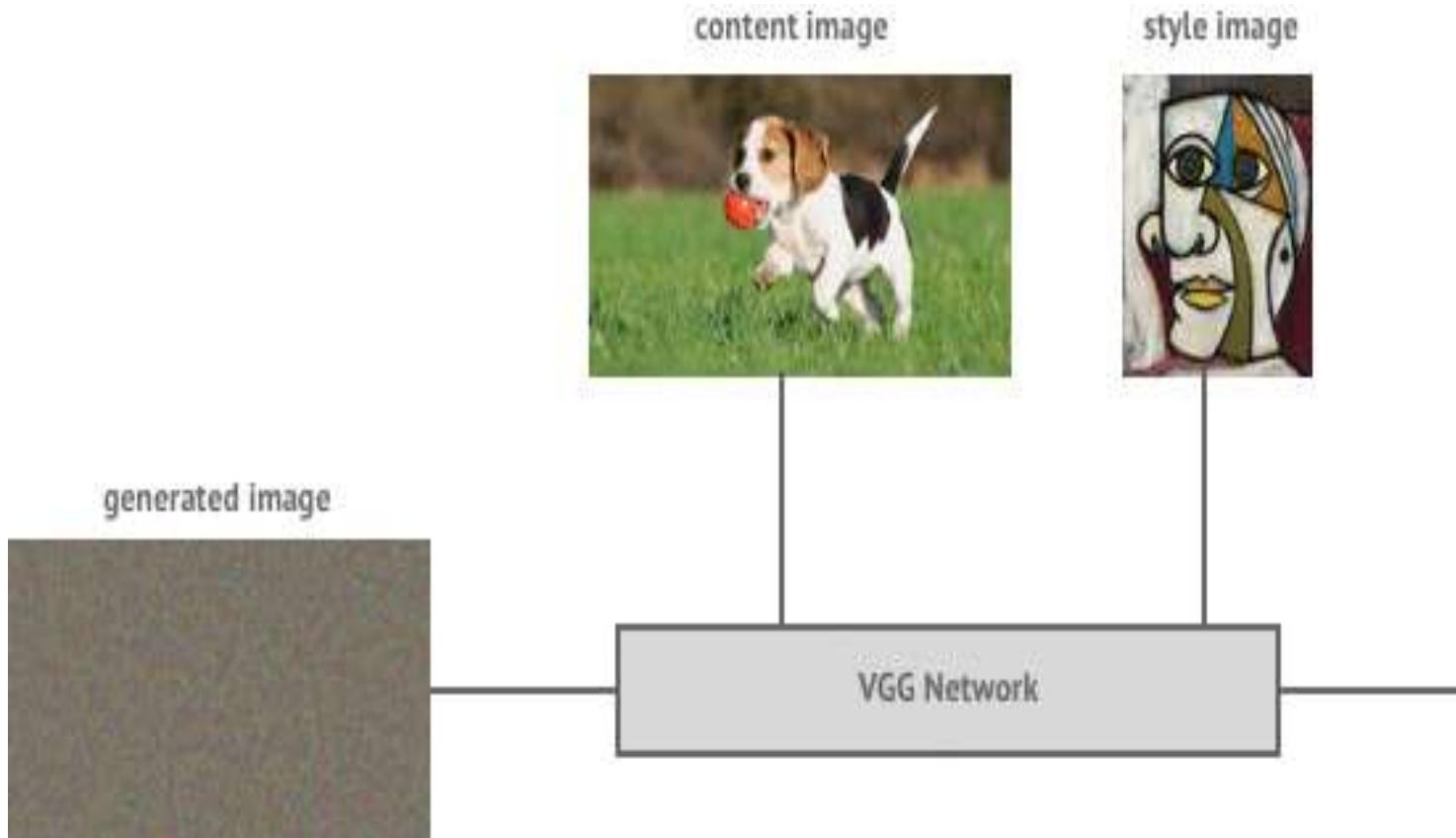
Визуализация работы



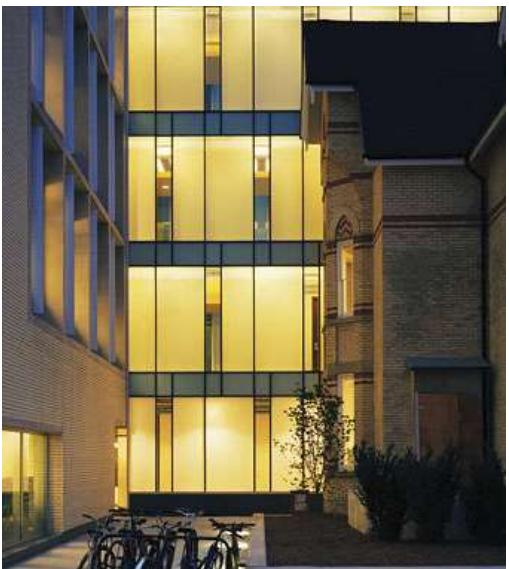
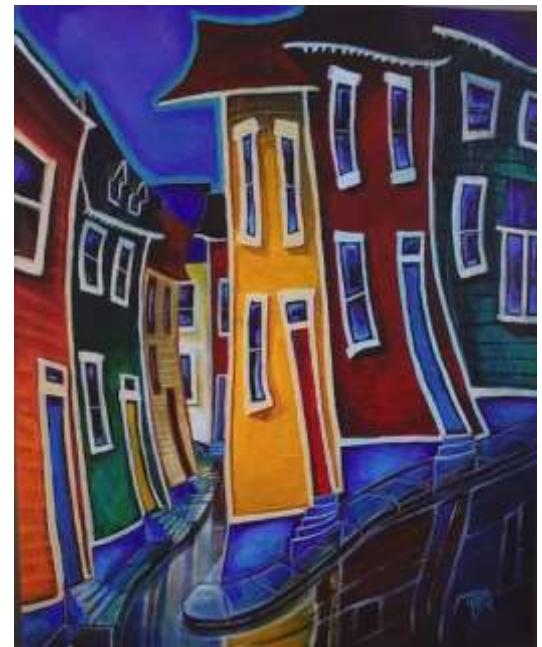
Результаты через
100 итераций
градиентного спуска



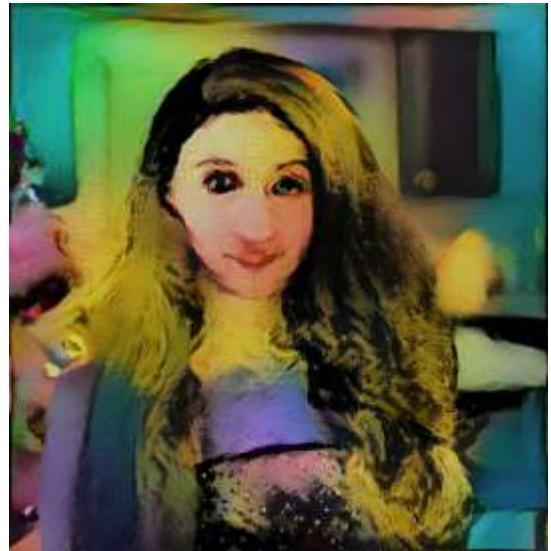
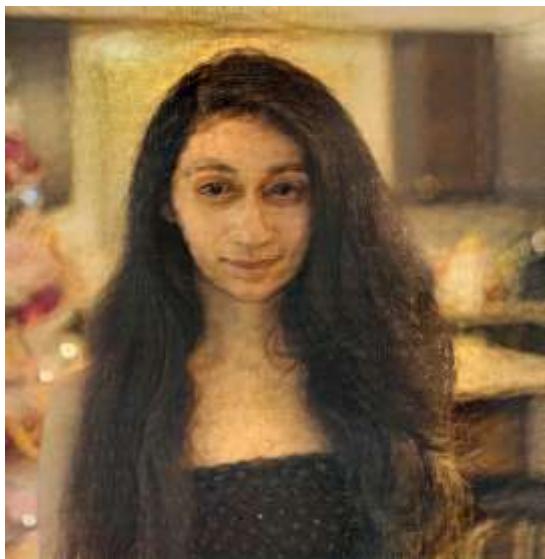
Визуализация



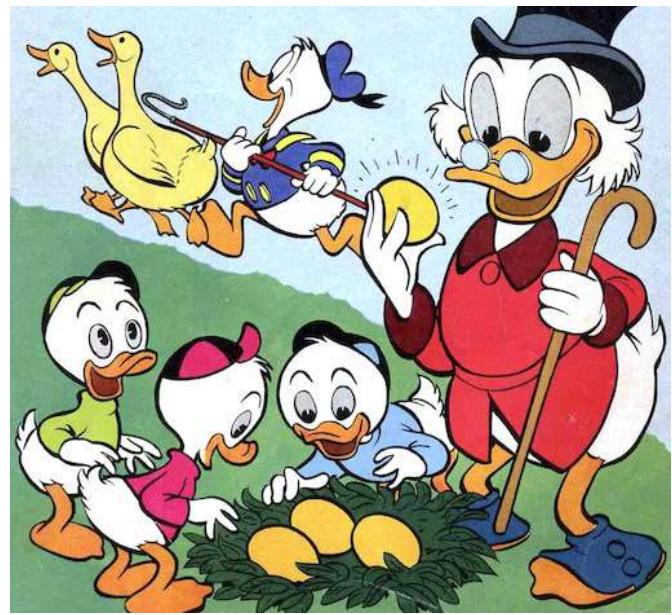
Пример работы



Пример работы



Пример работы



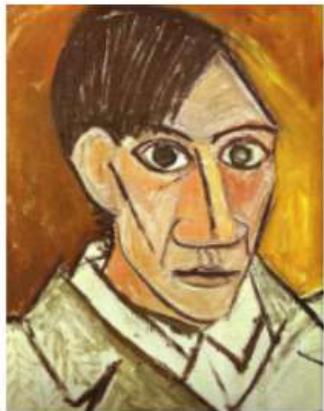
Соотношение стиля и содержания

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

Decrease α/β



Used for *Content*



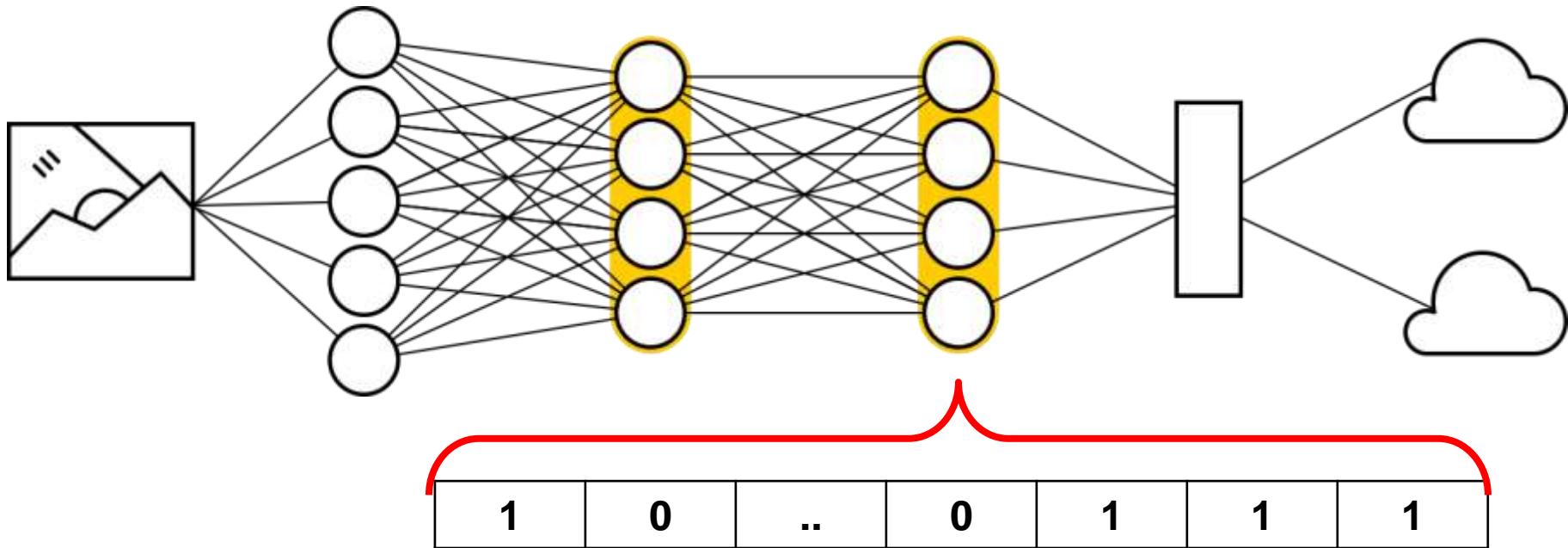
Used for *Style*

Deep Feature Interpolation



P. Upchurch et. al. Deep Feature Interpolation for Image Content Changes, CVPR 2017

Идея – манипуляция признаками



- Глубокие нейросети показывают отличные результаты на image classification, используя простые линейные классификаторы
- Значит признаковое пространство получается очень хорошее
- М.б. даже Евклидово?
- Можно работать напрямую в признаковом пространстве

Схема алгоритма

Deep Feature Interpolation

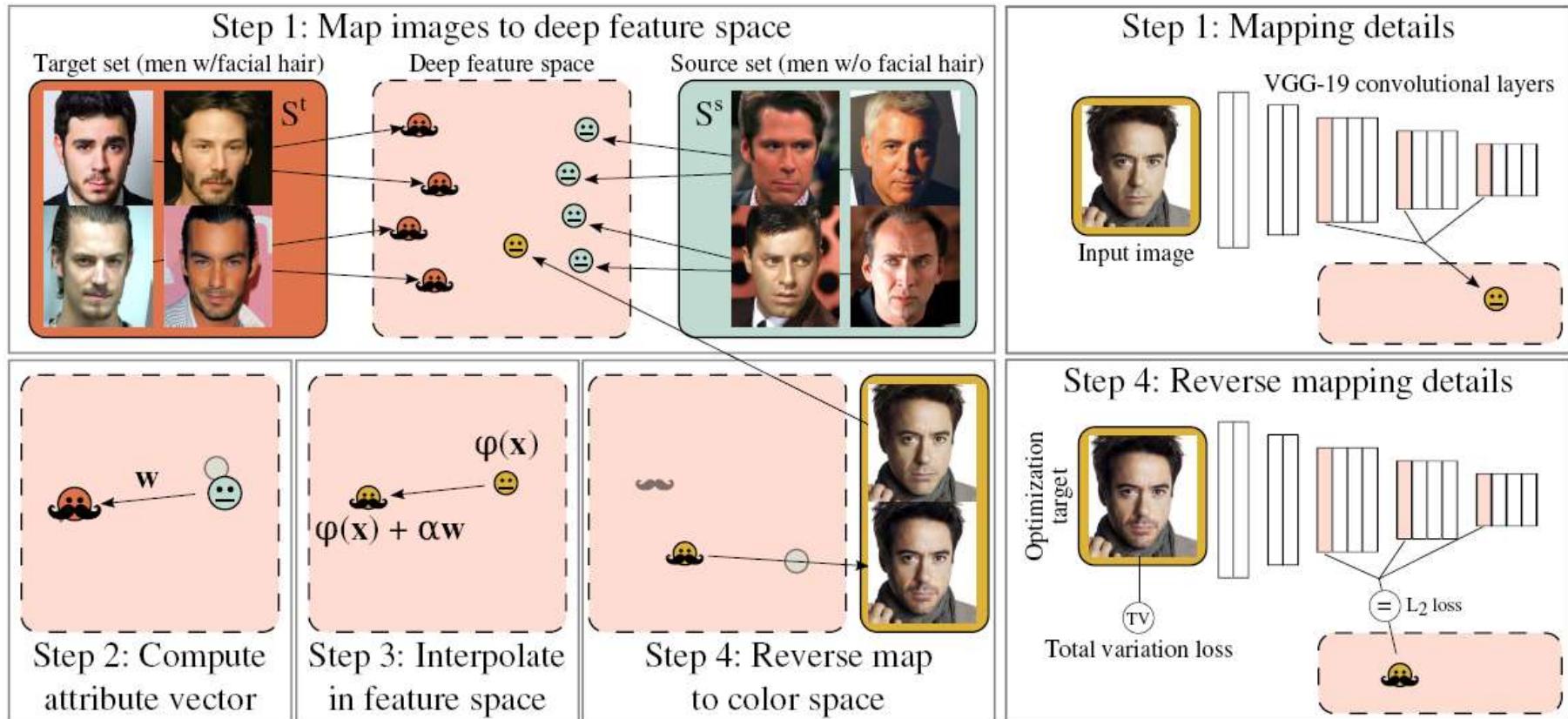
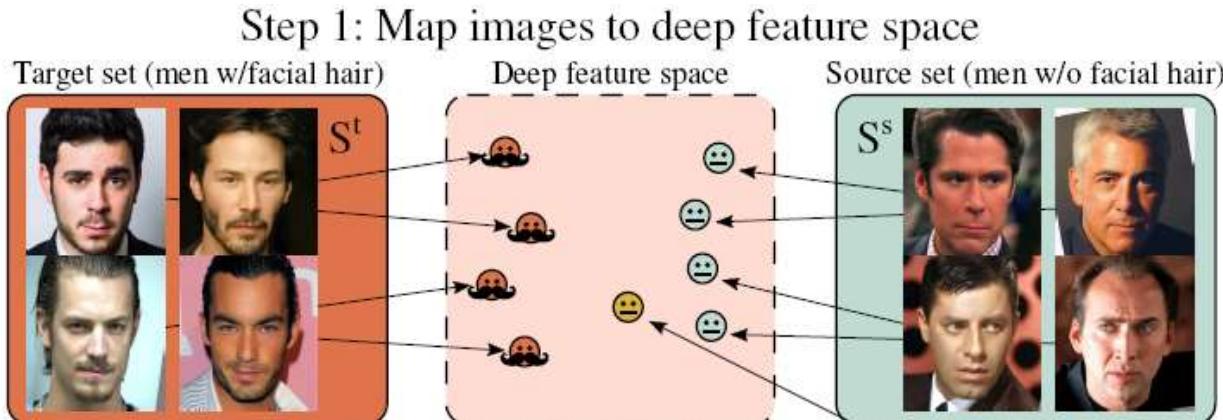


Figure 2. A schematic outline of the four high-level DFI steps.

Используем обычные VGG предобученные на ImageNet

Отображение и поиск похожих

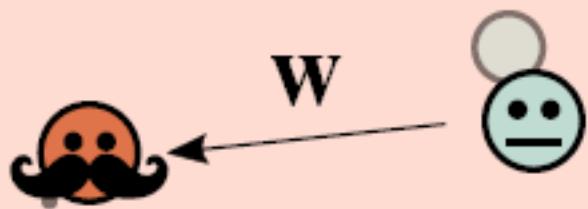


look believable. To ensure sufficient similarity we restrict \mathcal{S}^t and \mathcal{S}^s to the K nearest neighbors of \mathbf{x} . Let \mathcal{N}_K^t denote the K nearest neighbors of \mathcal{S}^t to $\phi(\mathbf{x})$; we define

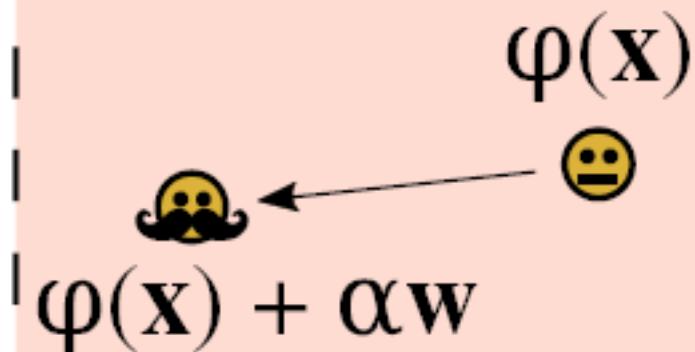
$$\bar{\phi}^t = \frac{1}{K} \sum_{\mathbf{x}^t \in \mathcal{N}_K^t} \phi(\mathbf{x}^t) \text{ and } \bar{\phi}^s = \frac{1}{K} \sum_{\mathbf{x}^s \in \mathcal{N}_K^s} \phi(\mathbf{x}^s). \quad (3)$$

These neighbors can be selected in two ways, depending on the amount of information available. When attribute labels are available, we find the nearest images by counting the number of matching attributes (e.g., matching gender, race, age, hair color). When attribute labels are unavailable, or as a second selection criterion, we take the nearest neighbors by cosine distance in deep feature space.

Модификация



Step 2: Compute
attribute vector

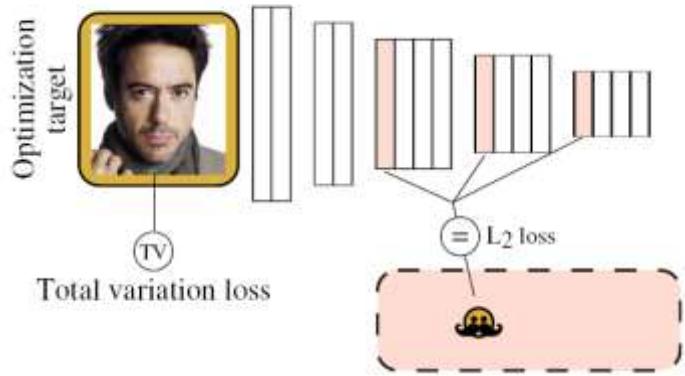


$\varphi(x)$
 $\varphi(x) + \alpha w$

Step 3: Interpolate
in feature space

Детали реконструкции изображения

Step 4: Reverse mapping details



Reverse mapping. The final step of DFI is to reverse map the vector $\phi(\mathbf{x}) + \alpha\mathbf{w}$ back into pixel space to obtain an output image \mathbf{z} . Intuitively, \mathbf{z} is an image that corresponds to $\phi(\mathbf{z}) \approx \phi(\mathbf{x}) + \alpha\mathbf{w}$ when mapped into deep feature space. Although no closed-form inverse function exists for the VGG mapping, we can obtain a color image by adopting the approach of [28] and find \mathbf{z} with gradient descent:

$$\mathbf{z} = \arg \min_{\mathbf{z}} \frac{1}{2} \|(\phi(\mathbf{x}) + \alpha\mathbf{w}) - \phi(\mathbf{z})\|_2^2 + \lambda_{V^\beta} R_{V^\beta}(\mathbf{z}), \quad (4)$$

where R_{V^β} is the Total Variation regularizer [28] which encourages smooth transitions between neighboring pixels,

$$R_{V^\beta}(\mathbf{z}) = \sum_{i,j} ((z_{i,j+1} - z_{i,j})^2 + (z_{i+1,j} - z_{i,j})^2)^{\frac{\beta}{2}} \quad (5)$$

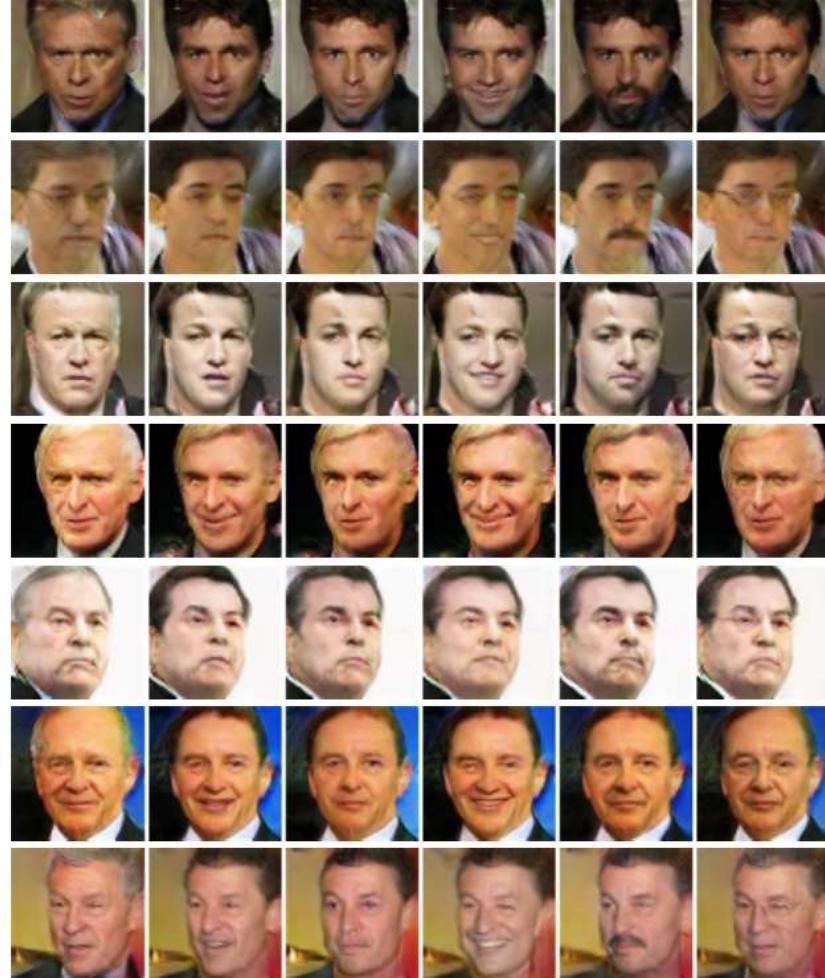
Here, $z_{i,j}$ denotes the pixel in location (i, j) in image \mathbf{z} . Throughout our experiments, we set $\lambda_{V^\beta} = 0.001$ and $\beta = 2$. We solve (4) with the standard hill-climbing algorithm L-BFGS [26].

Примеры

Older Mouth Open Eyes Open Smiling Moustache Glasses



Older Mouth Open Eyes Open Smiling Moustache Glasses



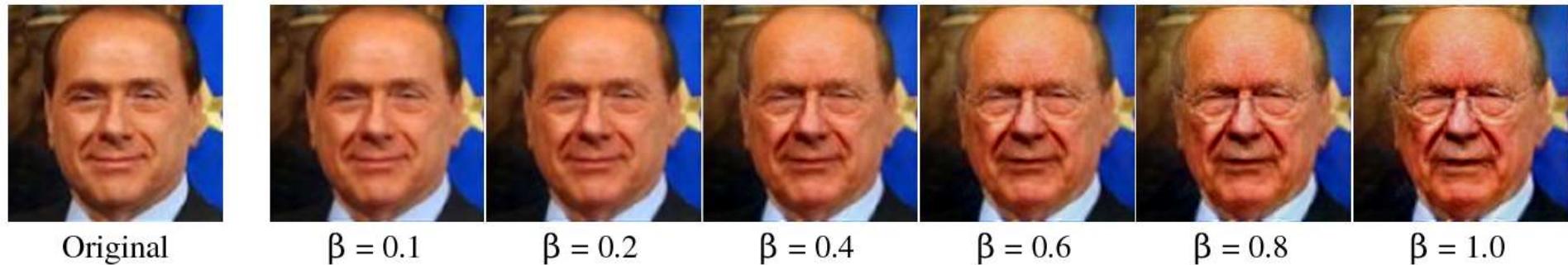
Aging



Facial hair



Варьирование параметра



Ещё пара моментов



Выравнивание лиц и других атрибутов (мужчина, Caucasian)

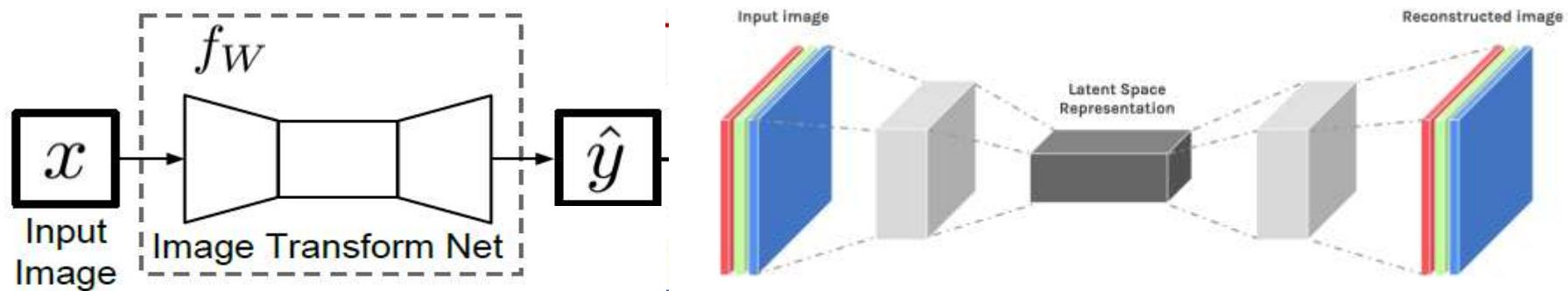
- Низкое разрешение – выравнивание глаз и рта на этапе предобработки коллекции
- Высокое разрешение – подгон лиц к тестовому перед сбором source & target набором для расчета вектора преобразования

Тип используемой сети для PerceptionLoss

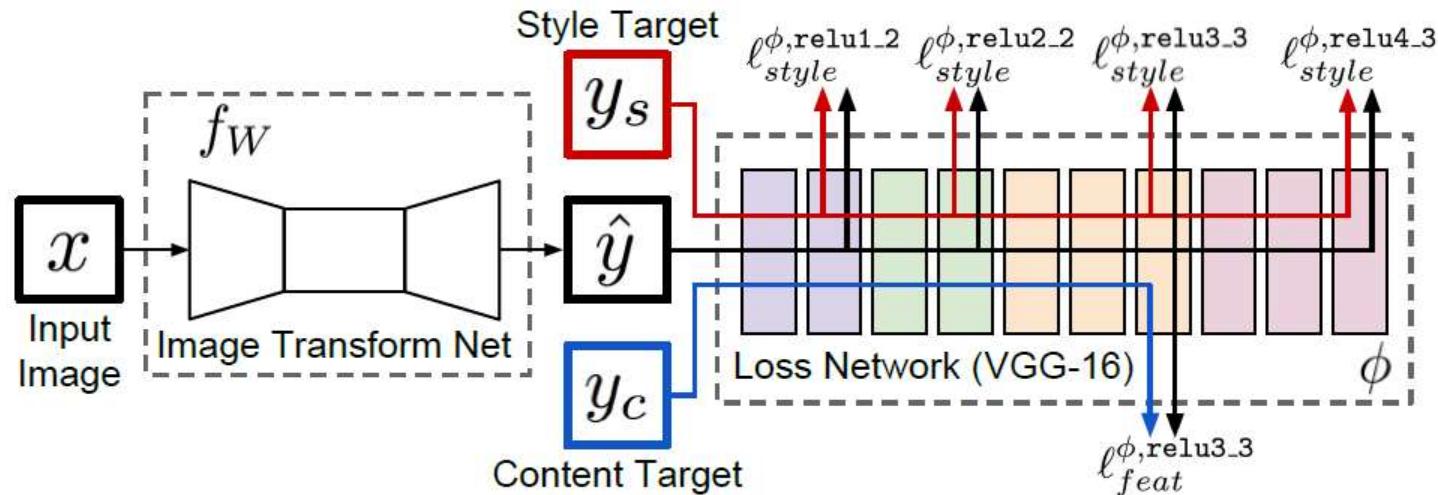
- VGG-face работала хуже VGG на ImageNet. «Внешность», включая атрибуты, она хуже передает
- Если мы хотим сохранить «личность», то можем использовать IdentityLoss, считая признаки на специальной сети, используемой для идентификации объектов/человека

Стилизация с Perception Loss

- Оптимизация с белого шума идёт медленно
- Идея: воспользоваться сетью-преобразователем
- Оценивать качество будем через perception loss
- Для каждого стиля нужно обучать свою собственную нейросеть

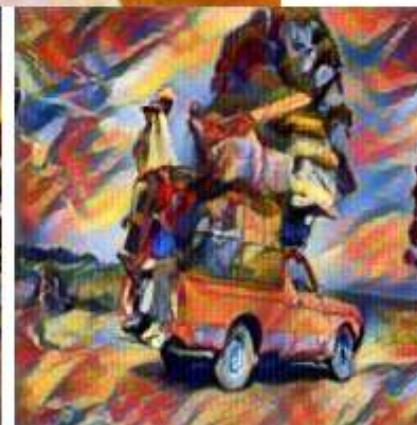


Обучение такой сети



- Воспользуемся предобученной на задаче классификации на imagenet сетью VGG-16
- Будем использовать её для извлечения признаков, и она будет зафиксирована при обучении трансформационной сети
- Через неё прогоняем изображения y_s (стиля), и $y_c=x$ (содержание)

Composition VII,
Wassily
Kandinsky,
1913

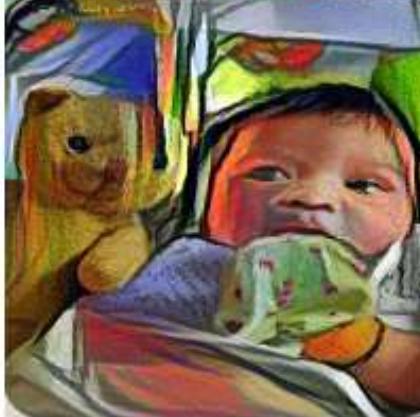


Original

Gatys et al.

Ours

The Muse,
Pablo Picasso,
1935



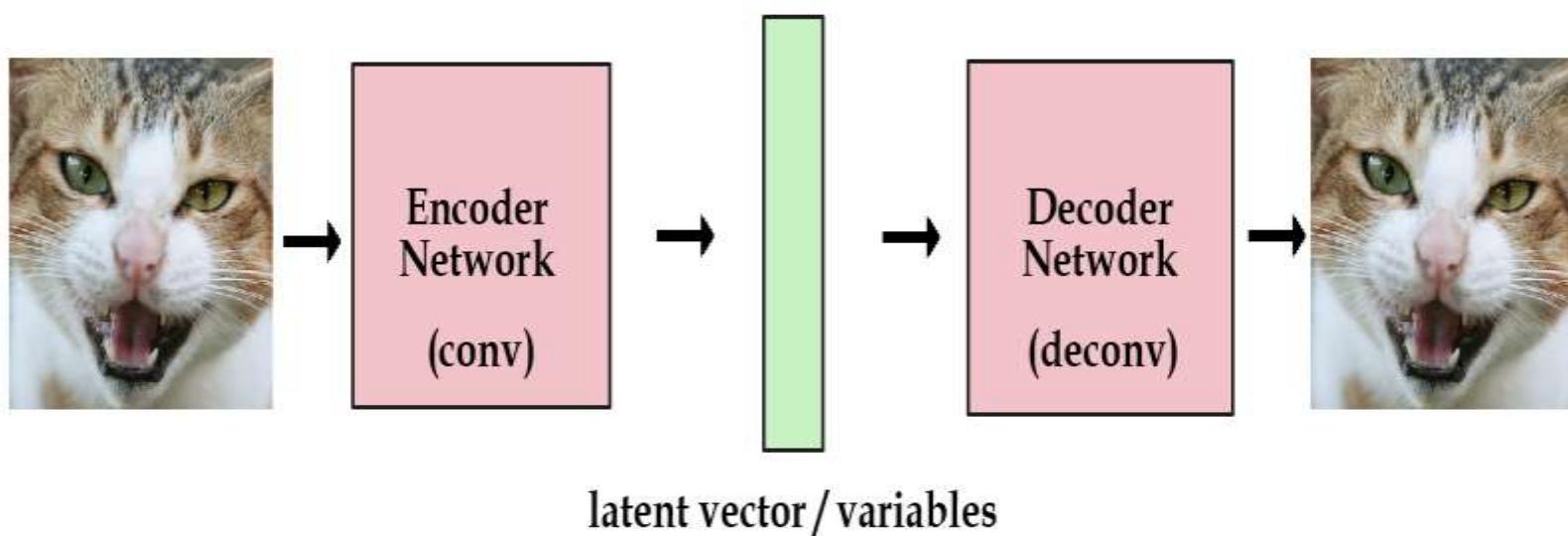
Original

Gatys et al.

Ours

Universal Style Transfer [Li et al, 2017]

- Хотим научиться быстро переносить произвольный стиль с помощью сети – преобразователя изображений
- Будем манипулировать нейросетевыми признаками изображения из сети encoder-decoder
- Обозначим признаки двух картинок F_c (содержание) и F_s (стиль). Пусть это матрицы размера [CxA], где C – количество признаков



Преобразование признаков

- Мы хотим, чтобы у новой картинки матрица Грама совпадала с $F_S F_S^T$
- Делаем в 2 этапа:
 - сначала уберем корреляцию исходных признаков
 - затем приравняем нужные матрицы



$$FF^T = I$$

$$FF^T = F_S F_S^T$$

Шаг 1

1. Для матрицы $F_c F_c^T$ найдем собственные векторы и значения Q и Λ
2. Для любой симметричной матрицы верно: $A = Q\Lambda Q^{-1}$
3. Первое преобразование: $whiten(F_c) = Q\Lambda^{-\frac{1}{2}}QF_c$, где Q и Λ – как раз и есть эти найденные матрицы
4. Пусть $whiten(F_c) = F_{cw}$. Тогда легко убедиться, что $F_{cw}F_{cw}^T = I$

Пример шага 1



Шаг 2

1. Второй шаг будет очень похож на первый:
посчитаем теперь векторы и лямбды для $F_S F_S^T$. Пусть
они равны P и Σ соответственно.
2. Построим второе преобразование:

$$\text{color}(F_{cw}) = P \Sigma^{\frac{1}{2}} P^T F_{cw}$$

3. Докажем, что полученное преобразование верно:

$$\begin{aligned} \text{color}(F_{cw})^* \text{color}(F_{cw})^T &= P \Sigma^{\frac{1}{2}} P^T F_{cw} F_{cw}^T P \Sigma^{\frac{1}{2}} P^T = P \Sigma P^T = \\ &= F_S F_S^T \end{aligned}$$

Итоговое преобразование: $\text{color}(\text{whiten}(F_c)) =$
 $P \Sigma^{\frac{1}{2}} P^T Q \Lambda^{-\frac{1}{2}} Q^T F_c$

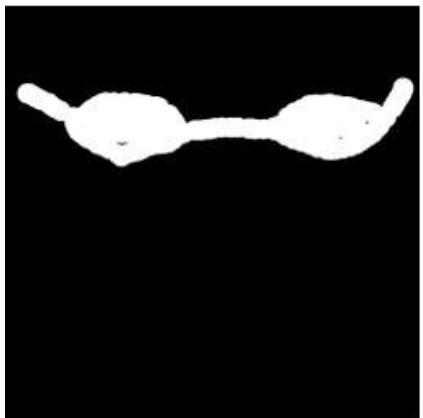
Пример работы



Смешение стилей



Content



Mask



Style I



Style II



Style transfer result

Генерация текстур

Замена изображения – содержания на шум



Texture



Scale 256

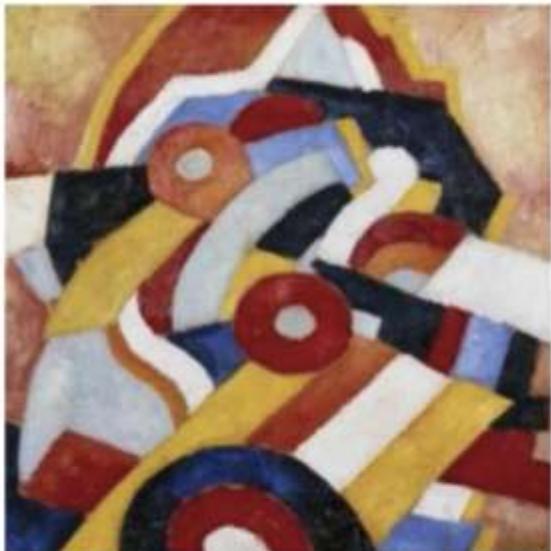


Scale 512



Scale 768

Пример работы на текстурах



Texture I



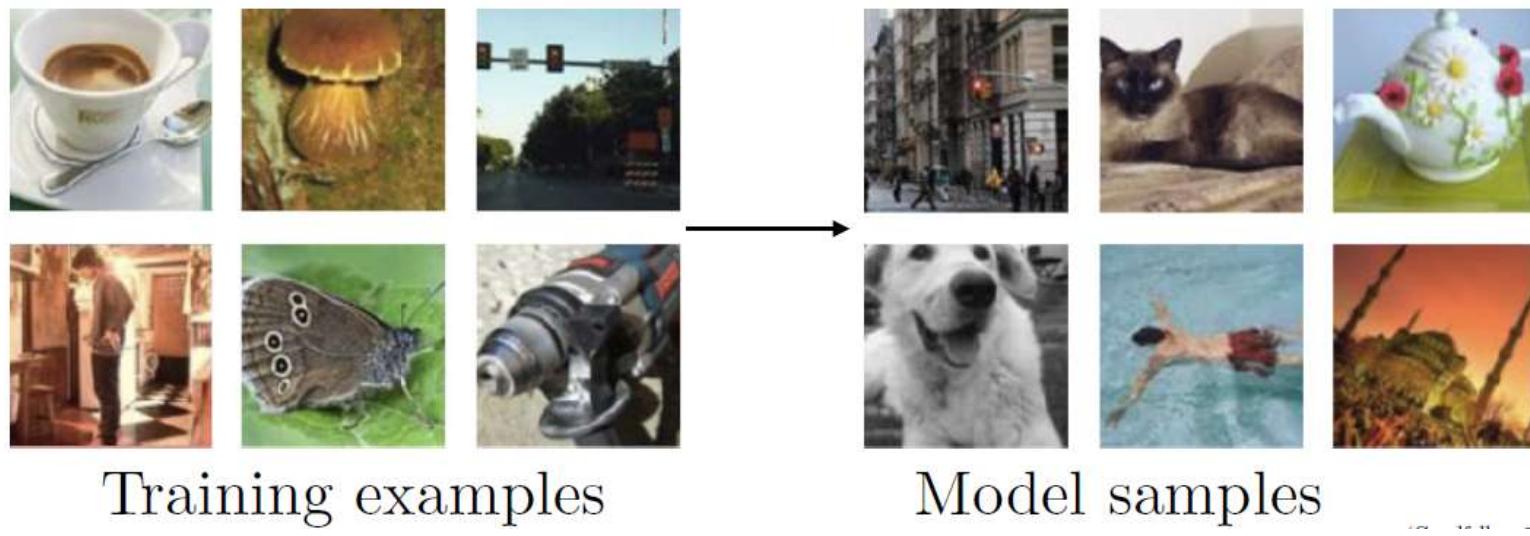
Interpolated synthesis result



Texture II

Генерация изображений

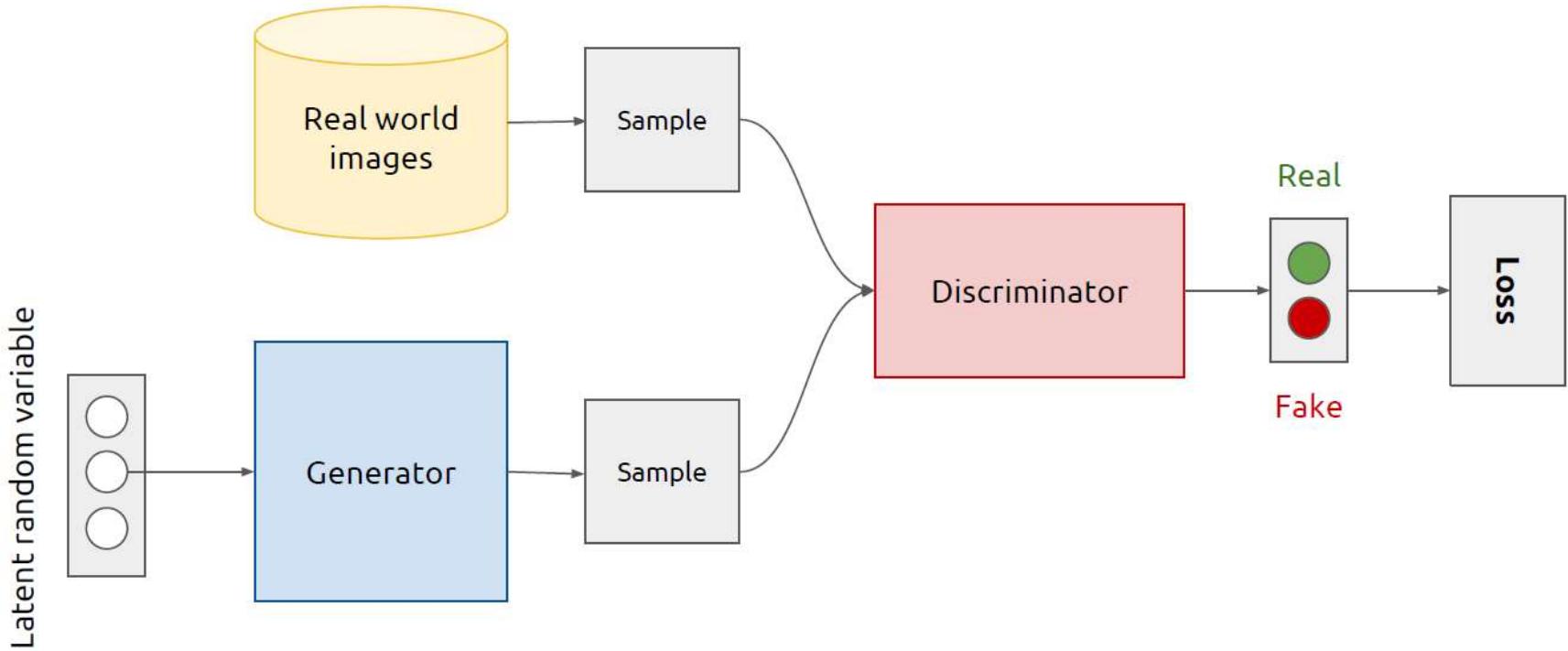
Хотим научиться генерировать изображения, похожие на обучающую выборку



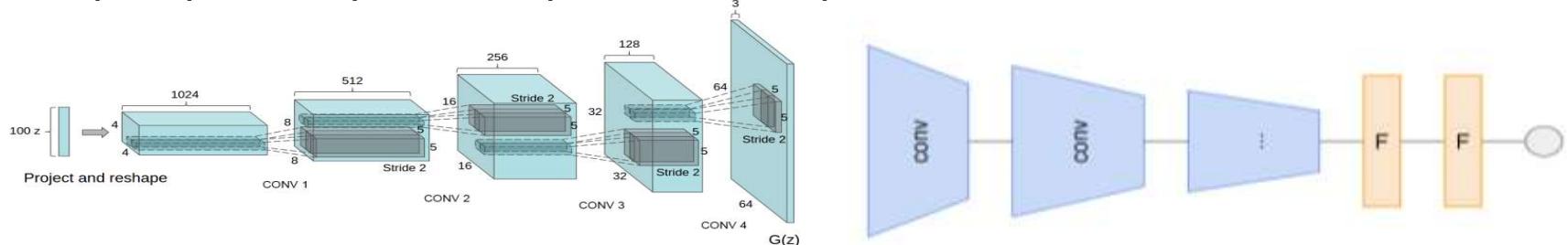
Как проверить, что у нас получилось хорошее, реалистичное изображение?

Общая схема Generative Adversarial Networks (GAN)

Сталкиваем генератор изображений из шума и дискриминатор:



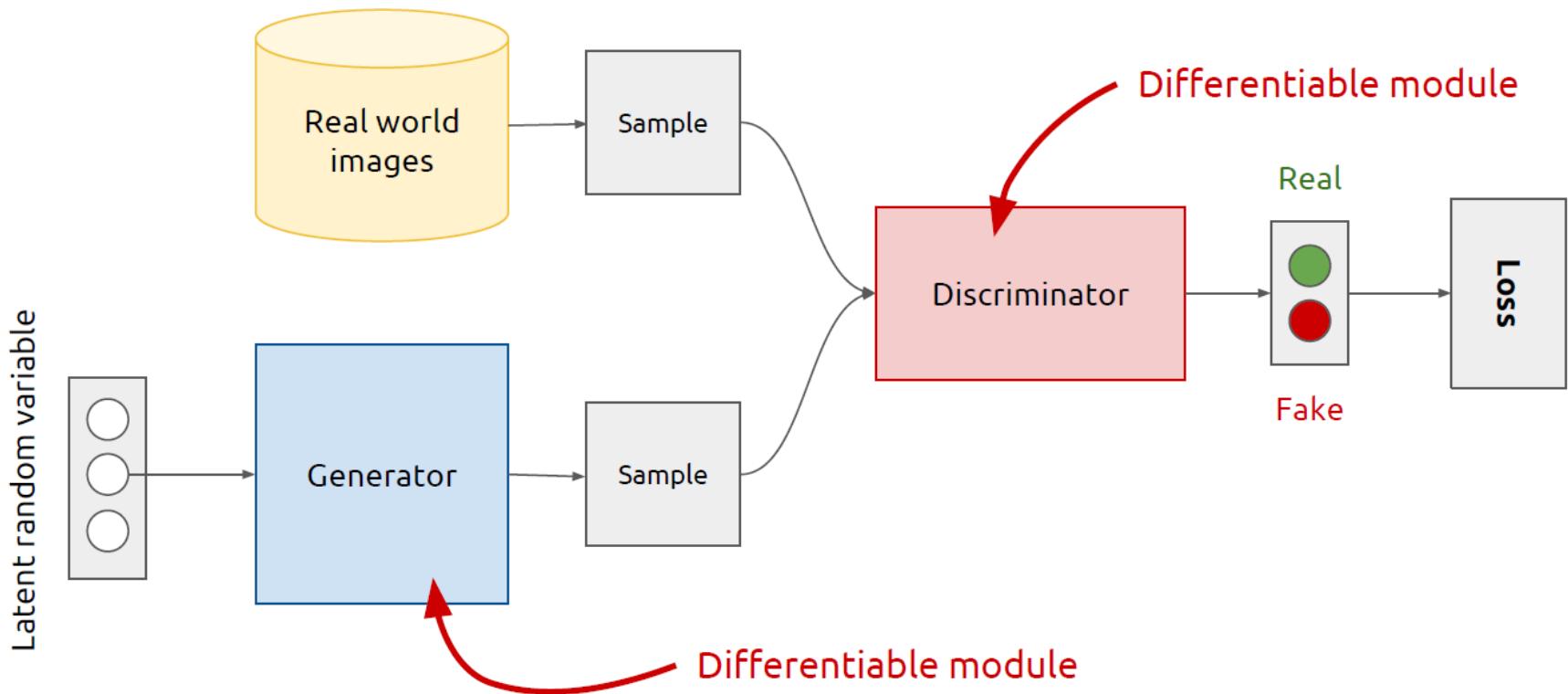
Генератор и дискриминатор обычно нейросети:



Ian Goodfellow et al, “Generative Adversarial Networks”, 2014

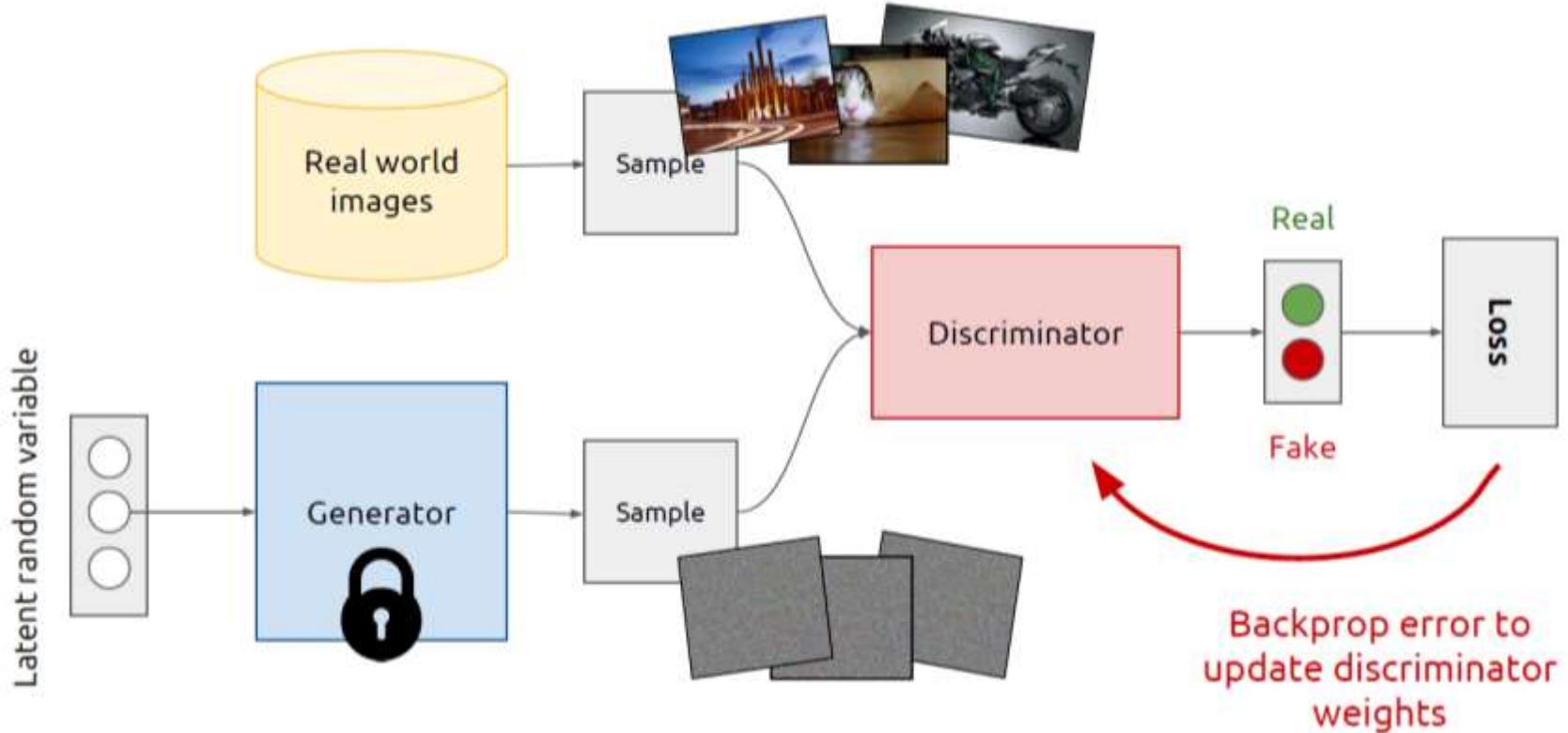
Обучение GAN

Будем обучать совместно, чередуя обучение дискриминатора и генератора



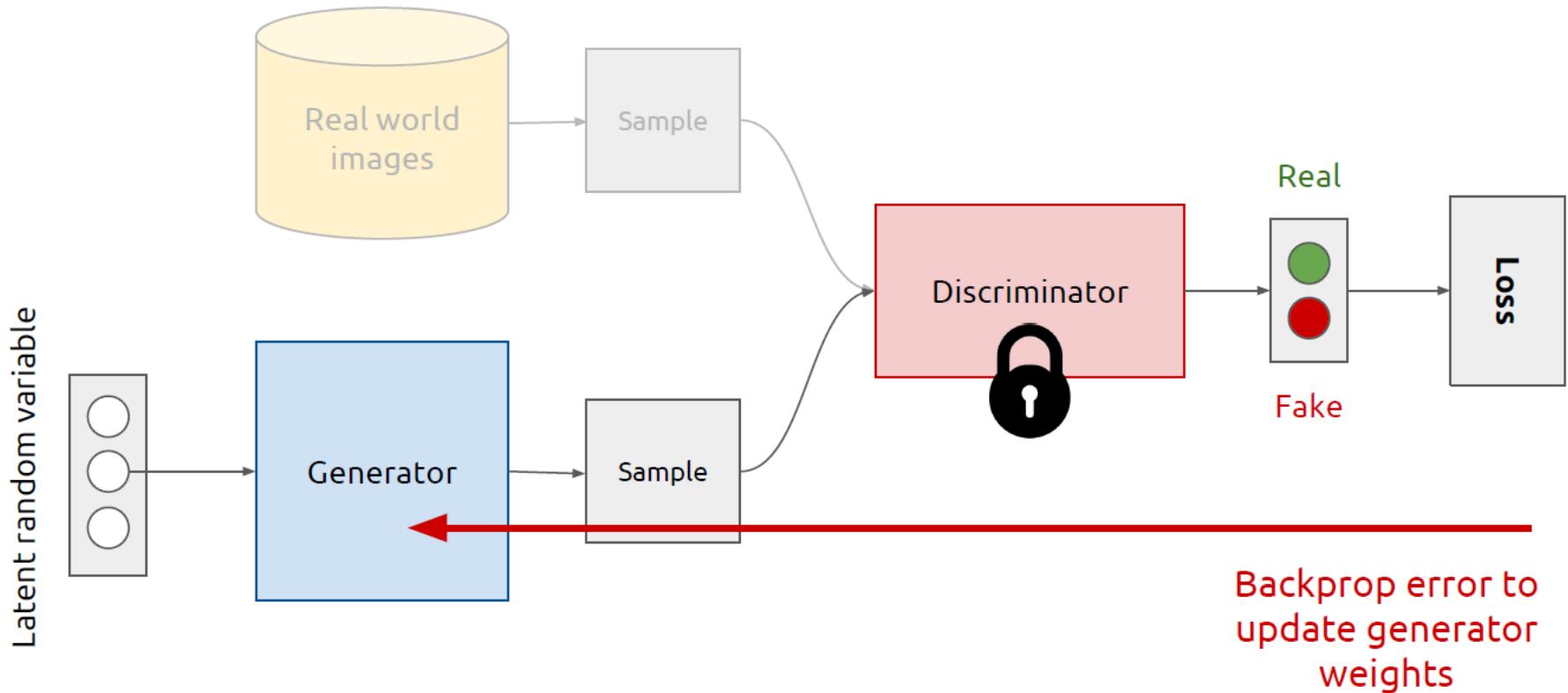
Почему нельзя взять готовый дискриминатор?

Обучение GAN



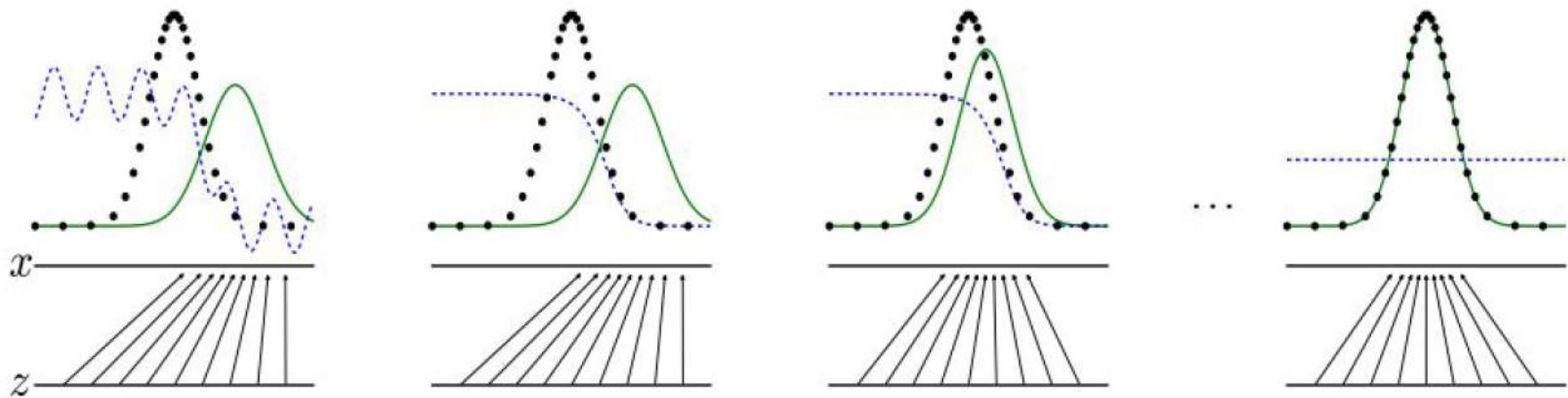
- Фиксируем веса генератора
- Берём примеры реальных изображений и синтезированных изображений
- Учим дискриминатор из различать

Обучение GAN



- Фиксируем веса дискриминатора
- Генерируем примеры генератором
- Распространяем ошибку от дискриминатора

Обучение GAN



- Чередование обучения дискриминатора повторяется до сходимости, которая может и не состояться
- В конце мы надеемся, что генератор научиться делать такие изображения, что дискриминатор не сможет их различить

Результаты



a)



b)



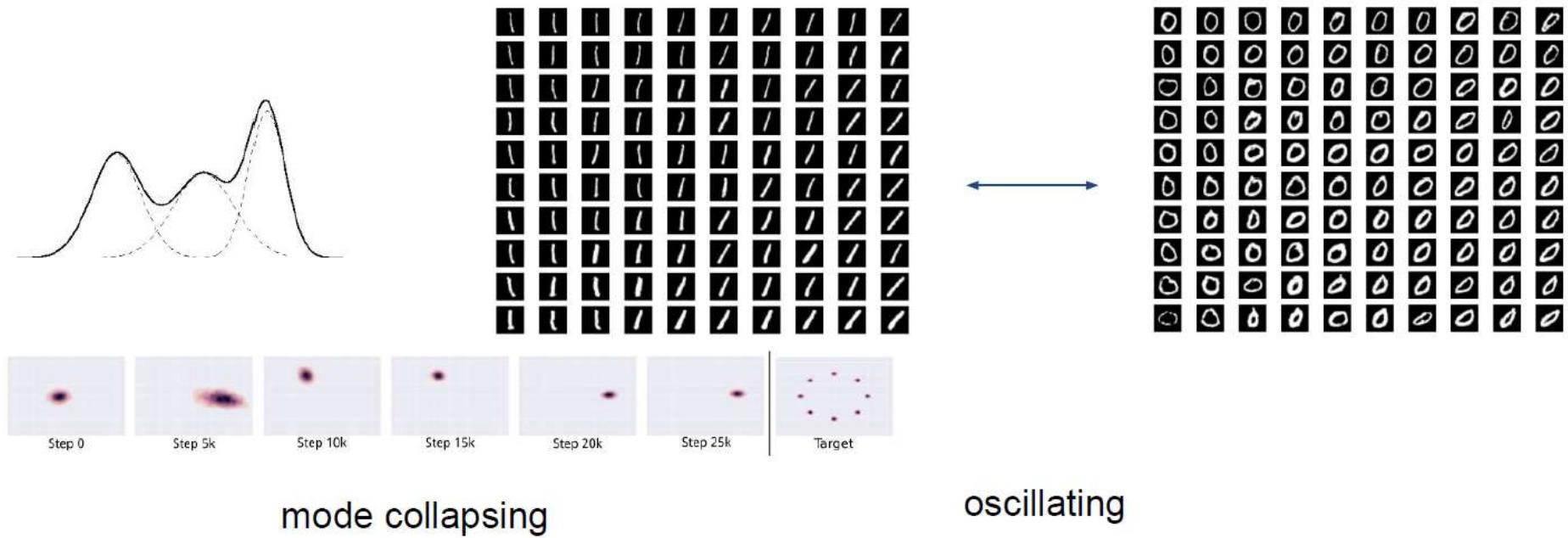
c)



d)

Основные проблемы GAN

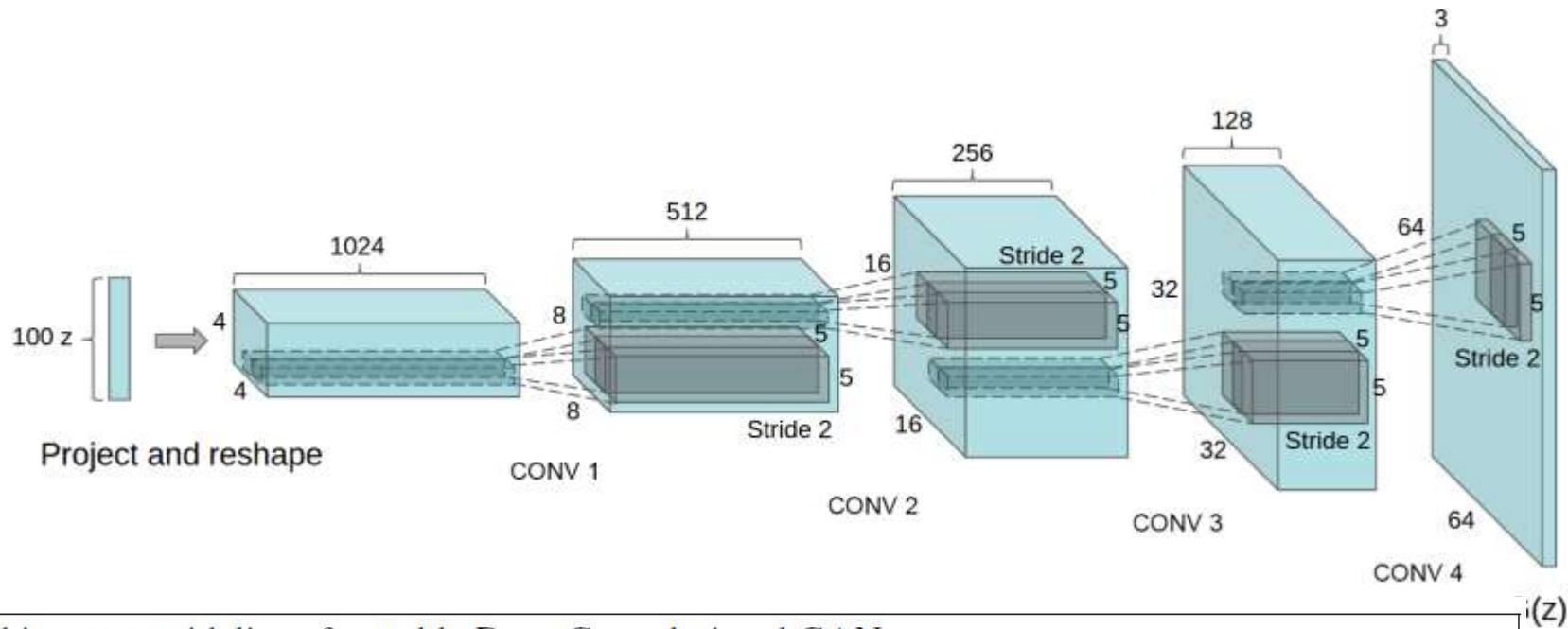
- Может не сходится – осциляция, схлопывание



- Сложно оценивать качество работы

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

DCGAN



Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Примеры - спальни



Примеры - лица



Примеры – обложки альбомов

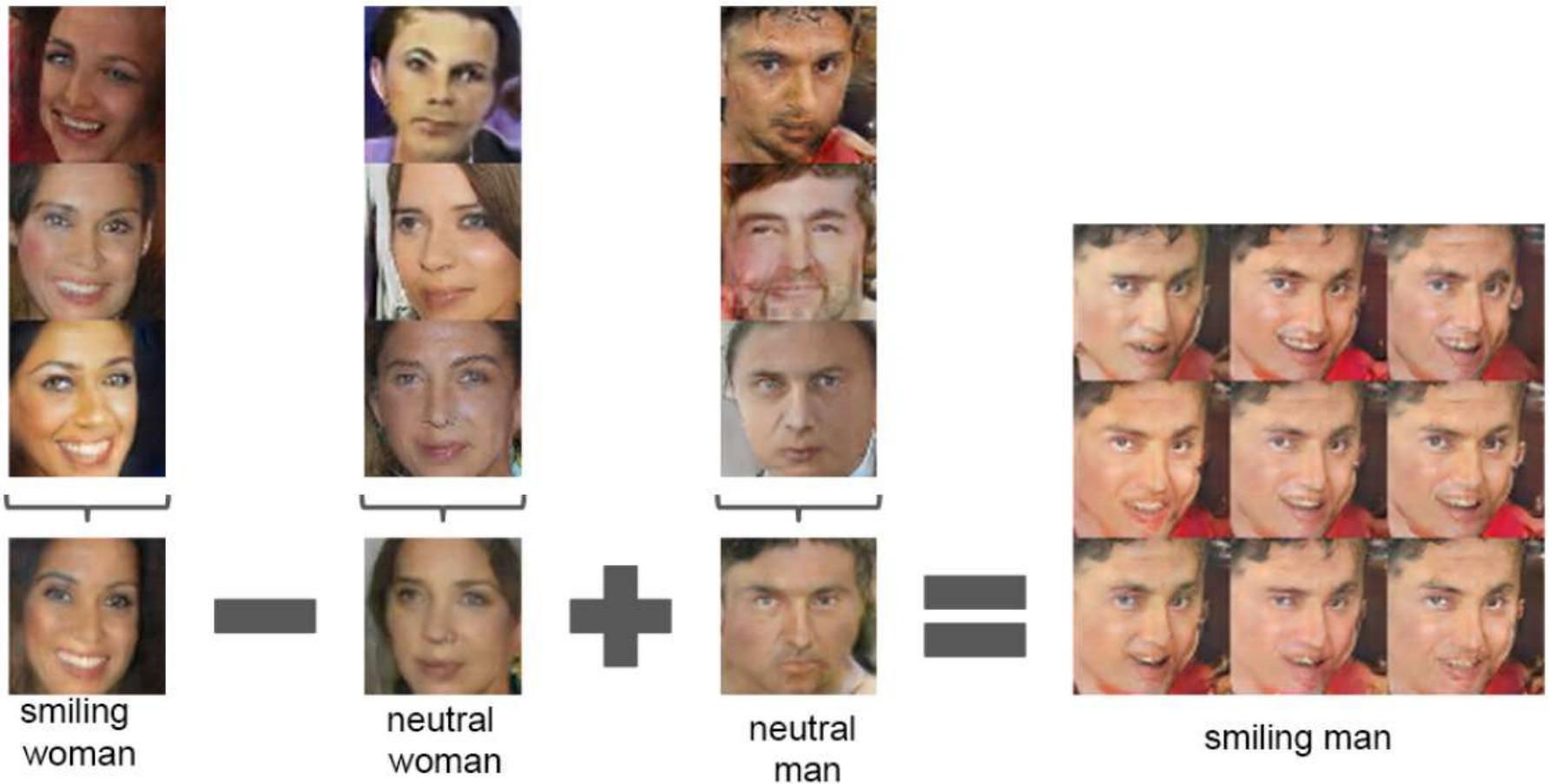


Изучение представления



- Поиск фильтров, которые соответствуют определённым объектам
 - Логистическая регрессия по активациям фильтров внутри нарисованных областей
- Обнуление фильтров, которые были выбраны как соответствующие объектам
- На изображениях «исчезли» окна при некотором ухудшении качества
- Мораль – получаем представление, в котором за разные объекты отвечают разные фильтры

Векторная арифметика



- Работа с z-векторами, которые использовались для генерации изображений
- Отдельные вектора нестабильны, поэтому сумма по 3м изображениям

Векторная арифметика



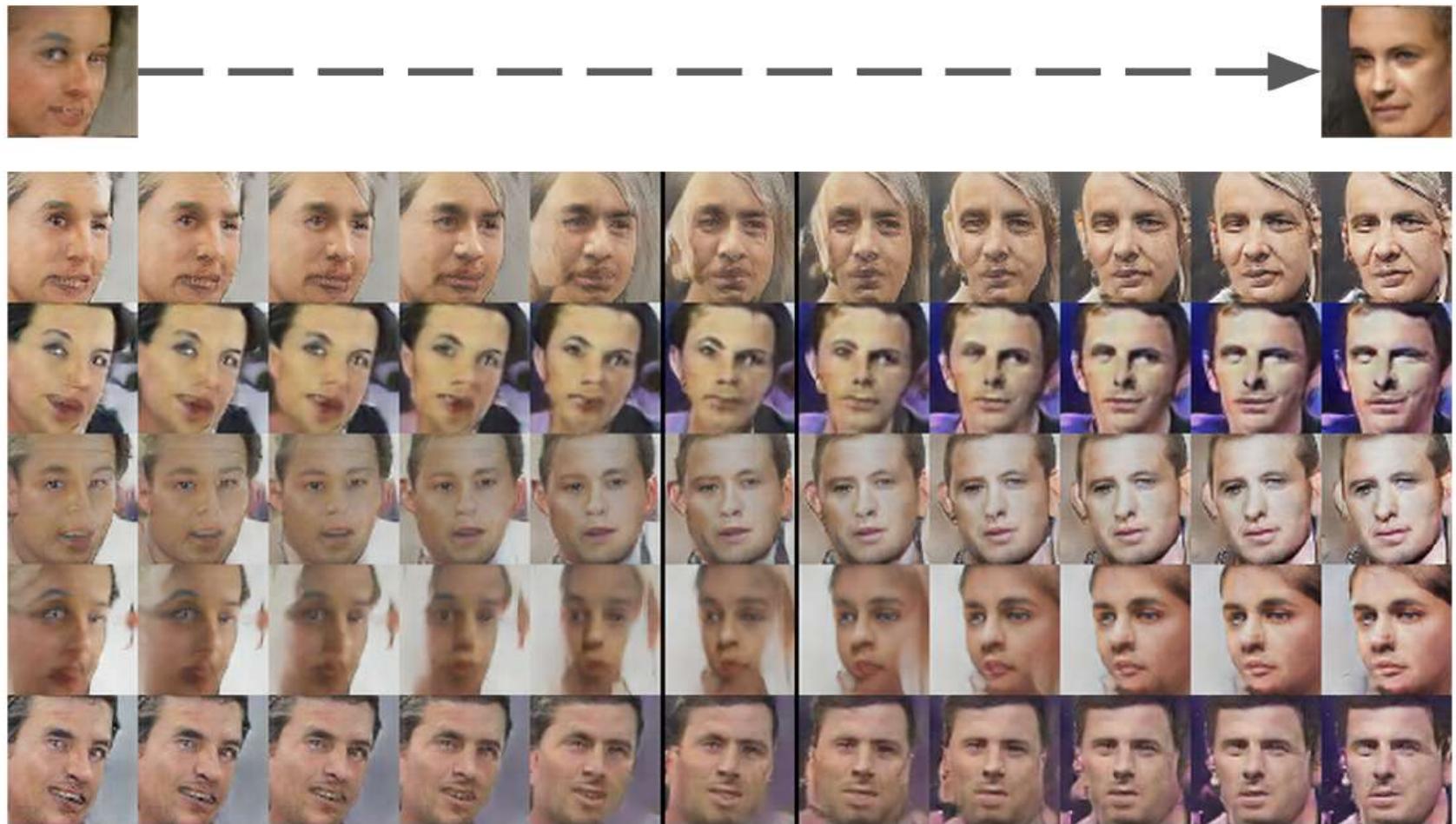
man
with glasses

man
without glasses

woman
without glasses

woman with glasses

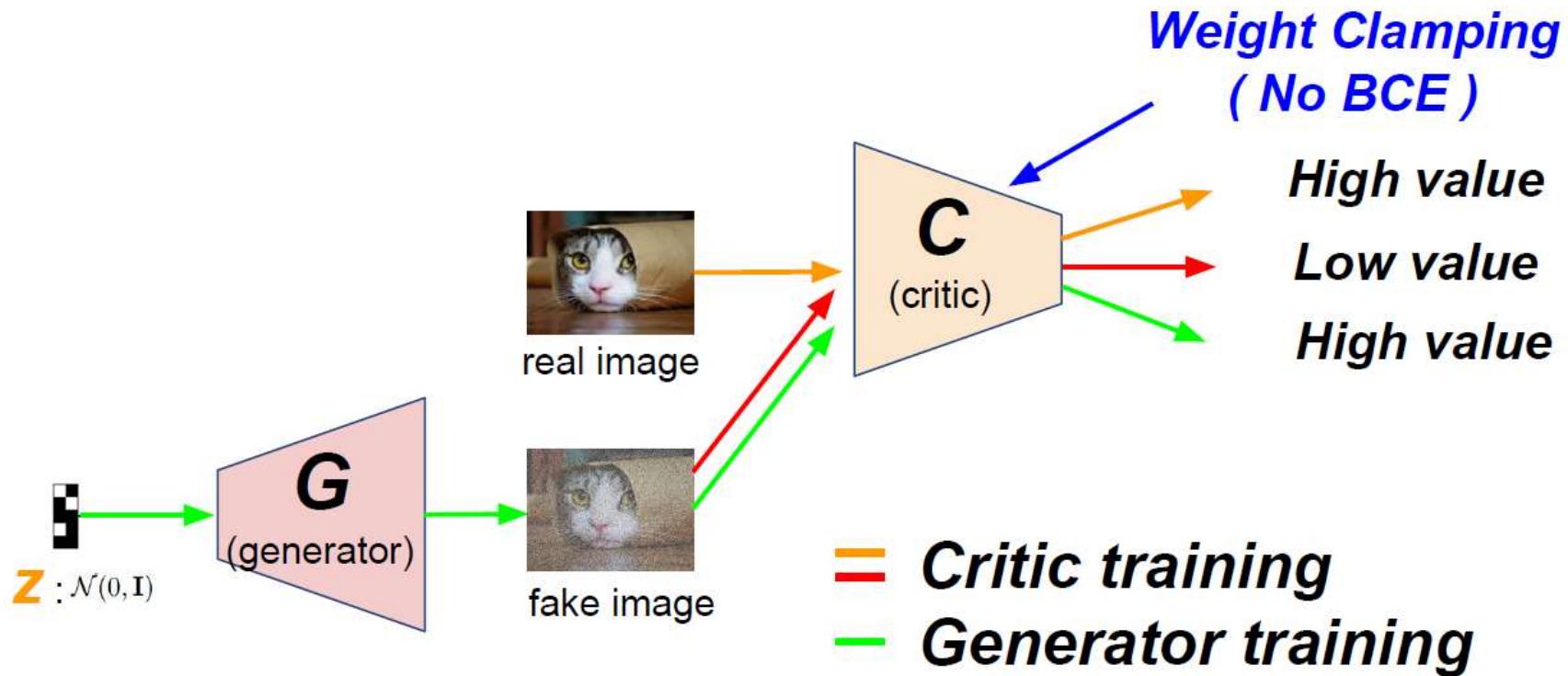
Векторная арифметика



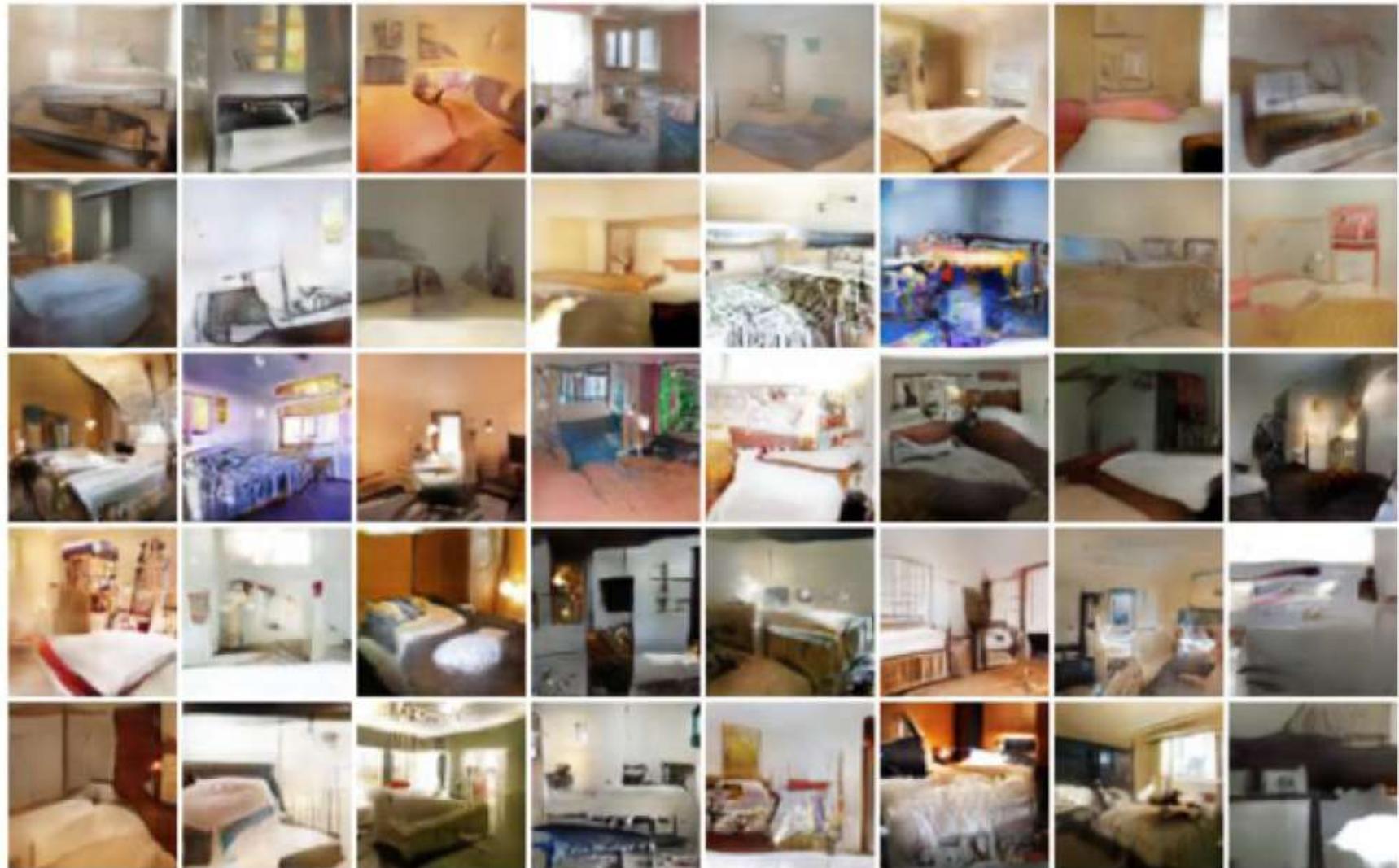
- Вектор «поворота» как разница между усреднёнными векторами 4х изображений с лицами повернутыми налево и 4мя направо

Wasserstein GAN

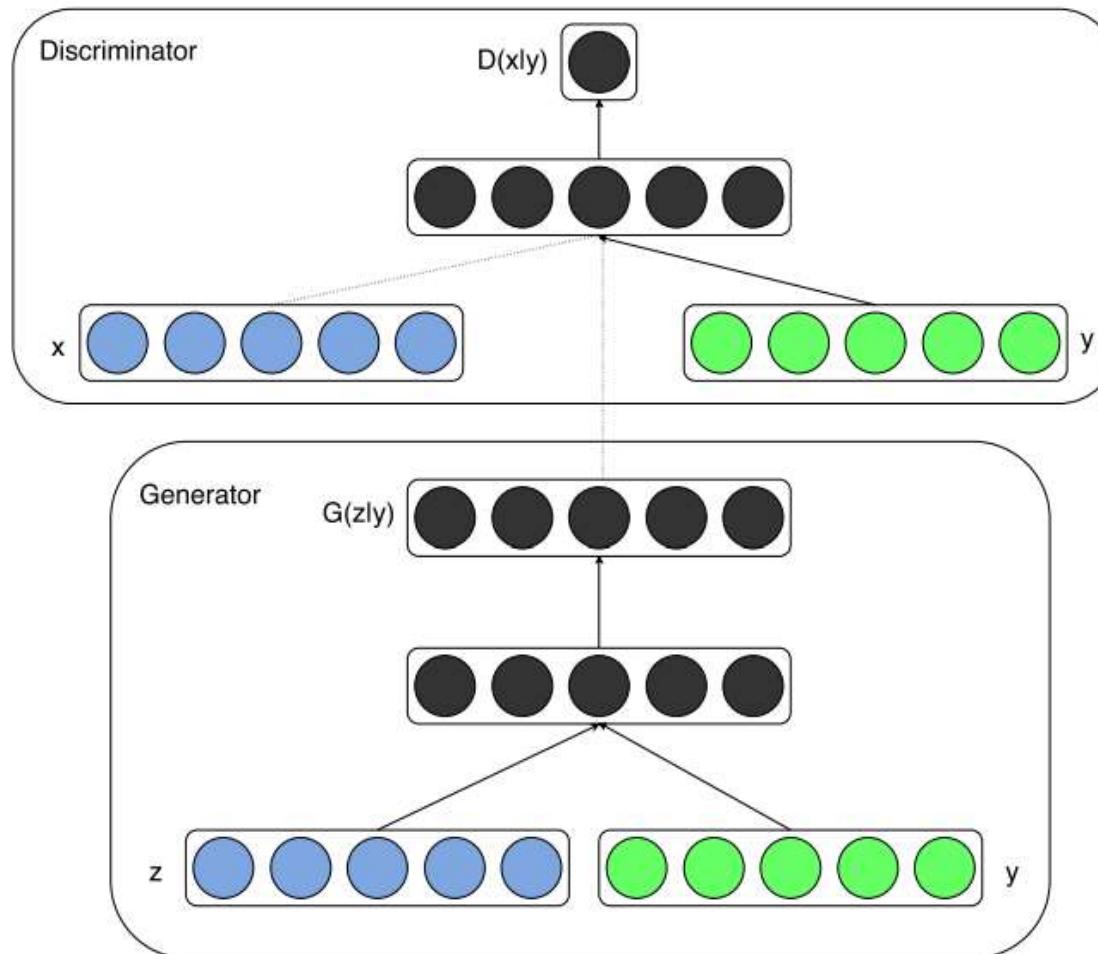
- Martin et al, Wasserstein GAN, 2017



Результаты

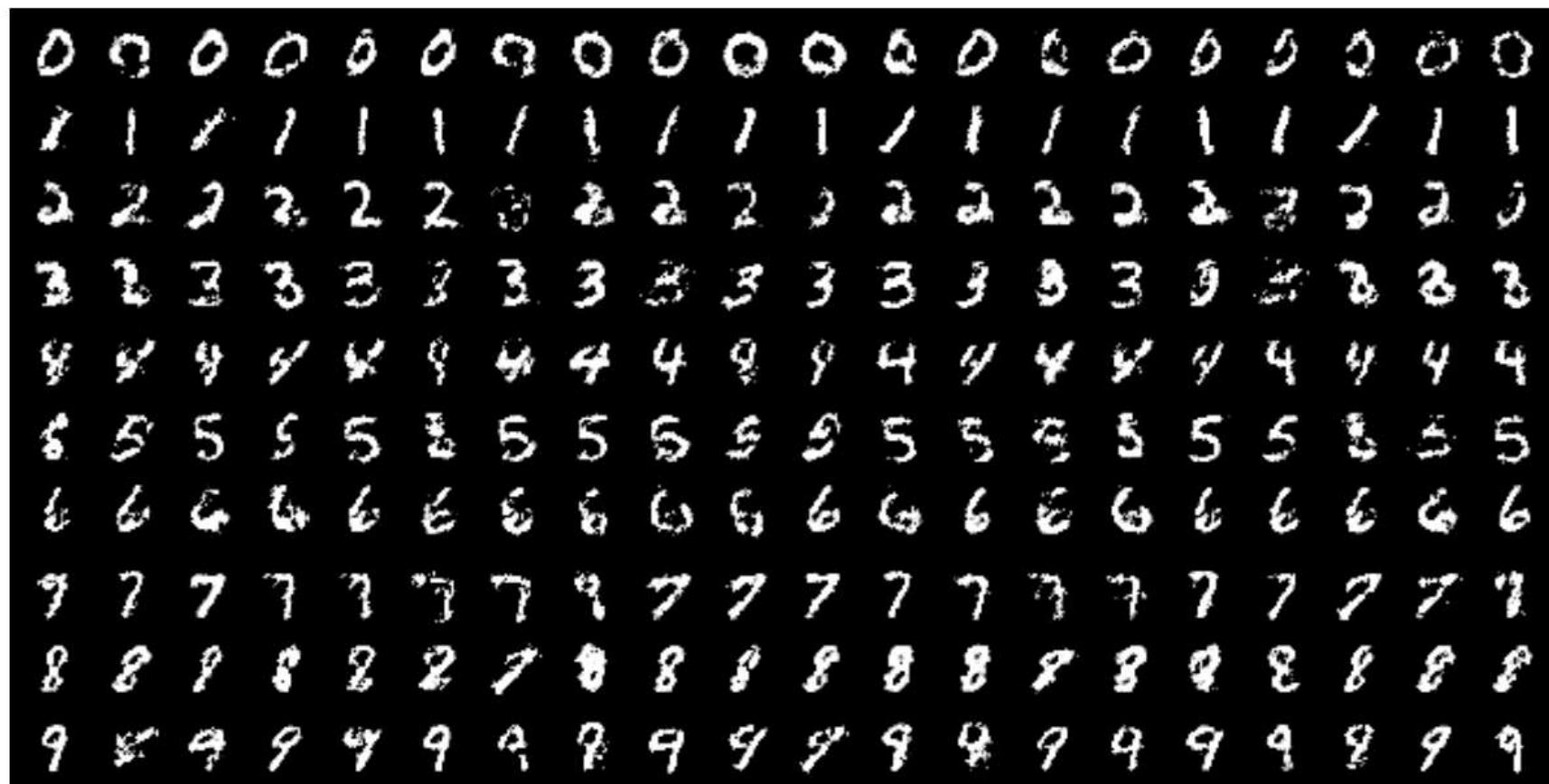


Conditional GAN

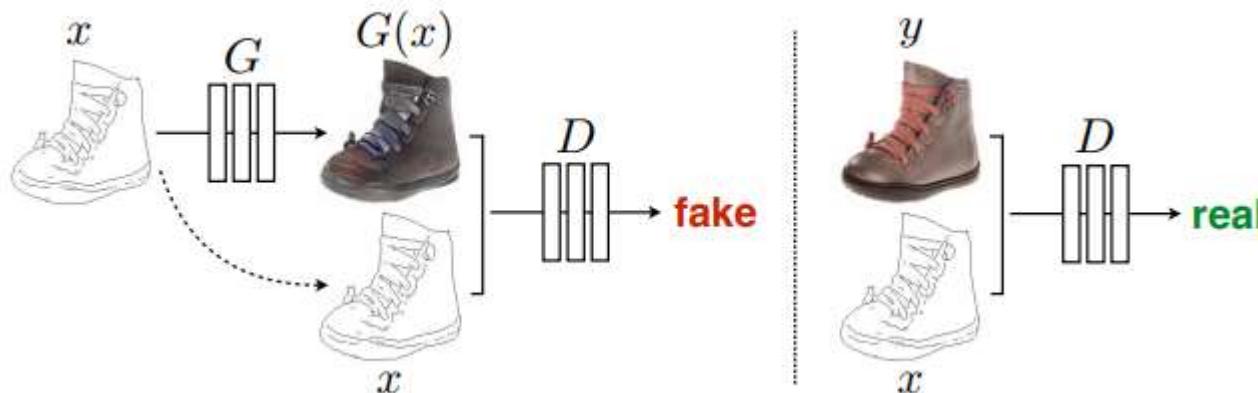


Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." *arXiv preprint arXiv:1411.1784* (2014).

Conditional GAN



Pix2Pix

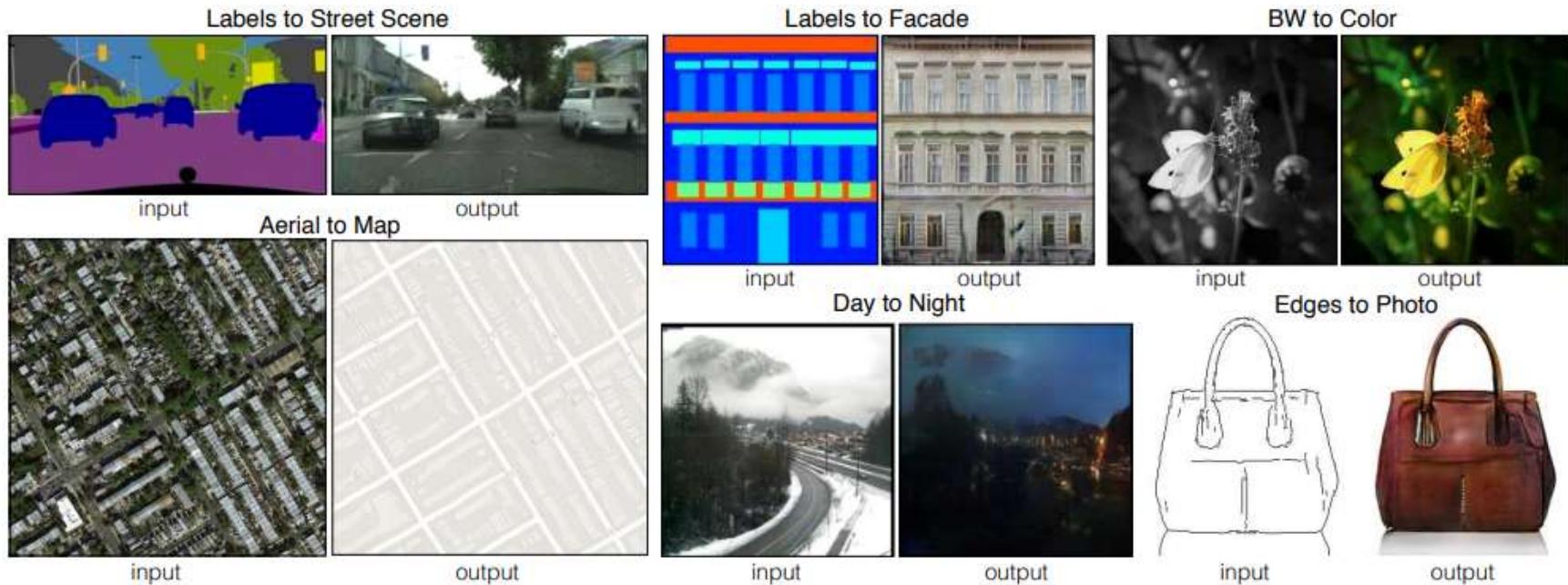


$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

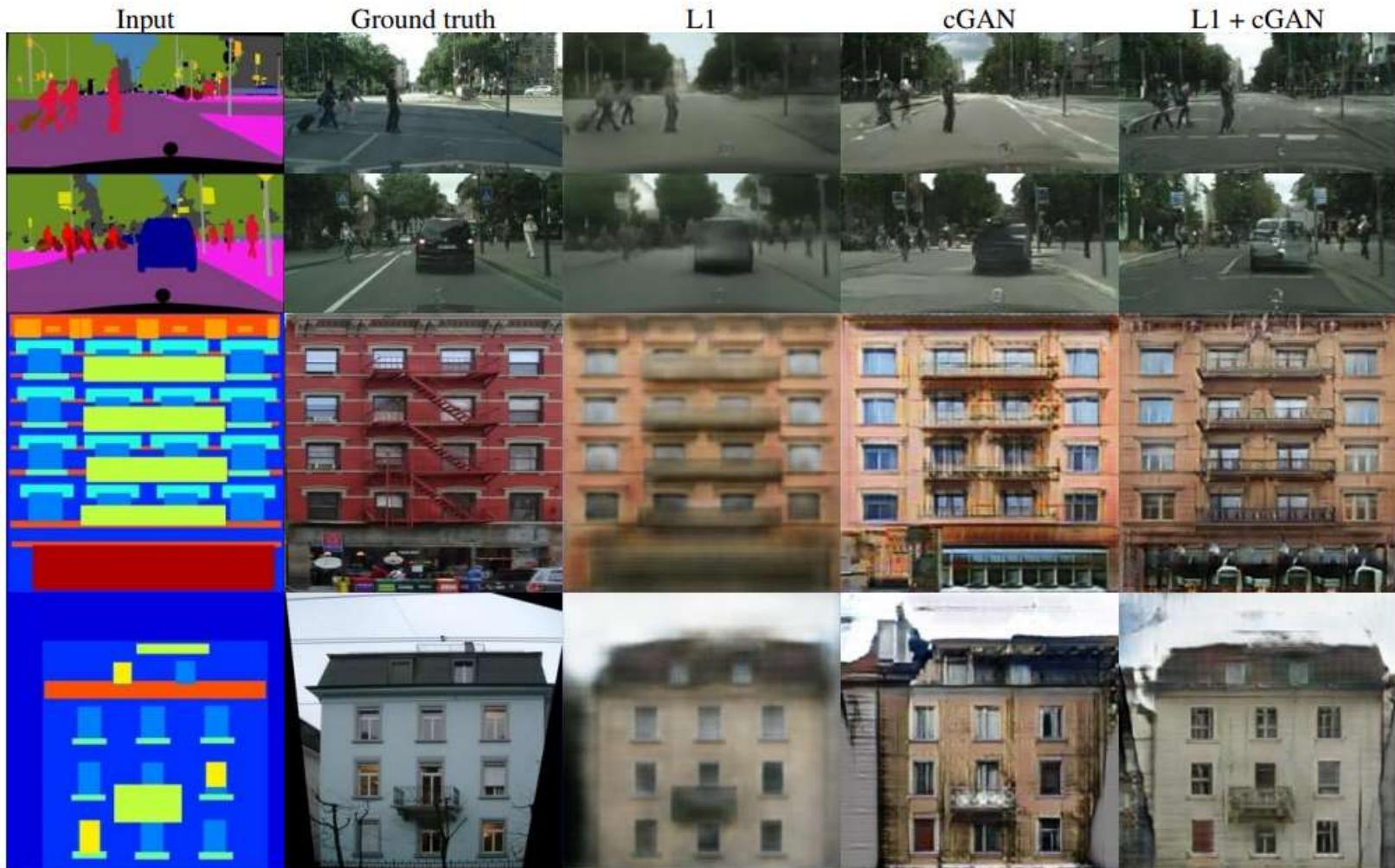
$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1] \quad \mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))]$$

Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." *arXiv preprint* (2017).

Pix2Pix



Pix2Pix

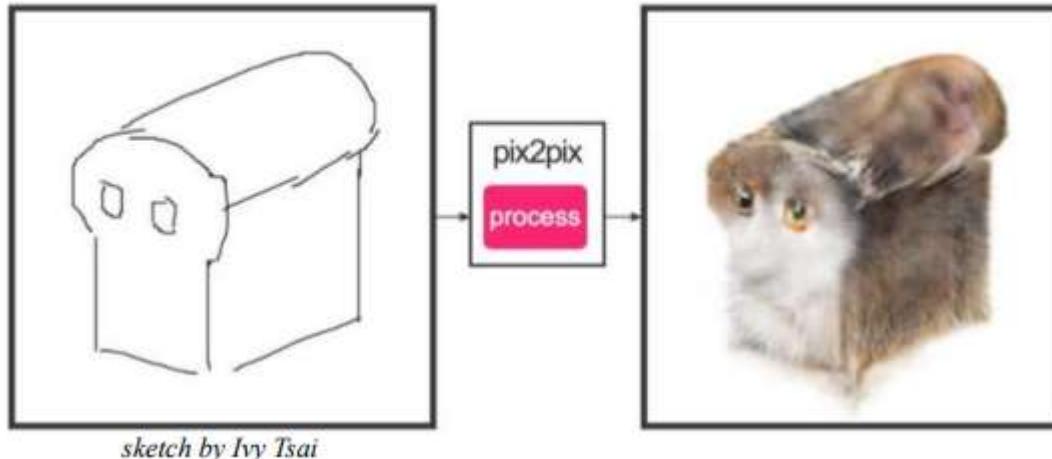


Pix2Pix



Pix2Pix

#edges2cats by Christopher Hesse



Background removal



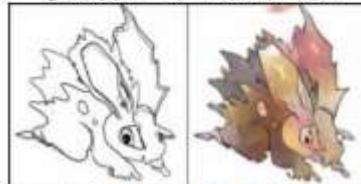
by Kaihu Chen

Palette generation



by Jack Qiao

Sketch → Pokemon



by Bertrand Gondouin

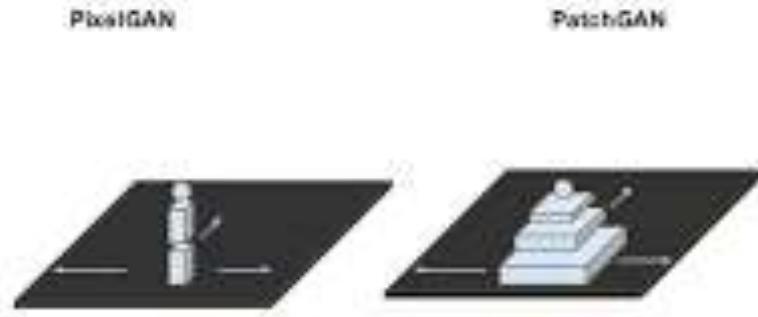
“Do as I do”



by Brannon Dorsey

Pix2Pix

Discriminator - Patch GAN



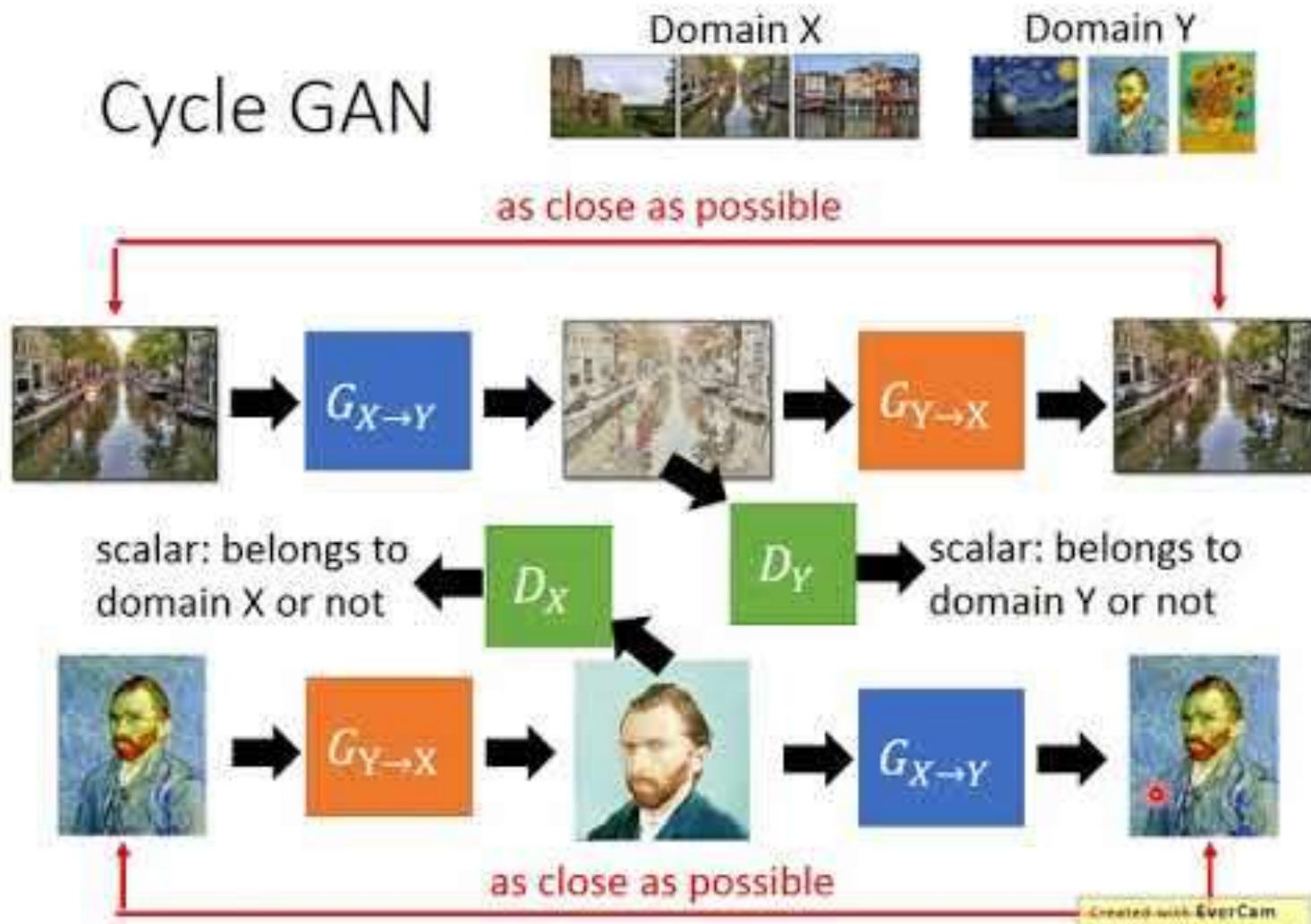
Архитектура:

- Генератор: U-net
- Дискриминатор: PatchGAN

Ограничение:

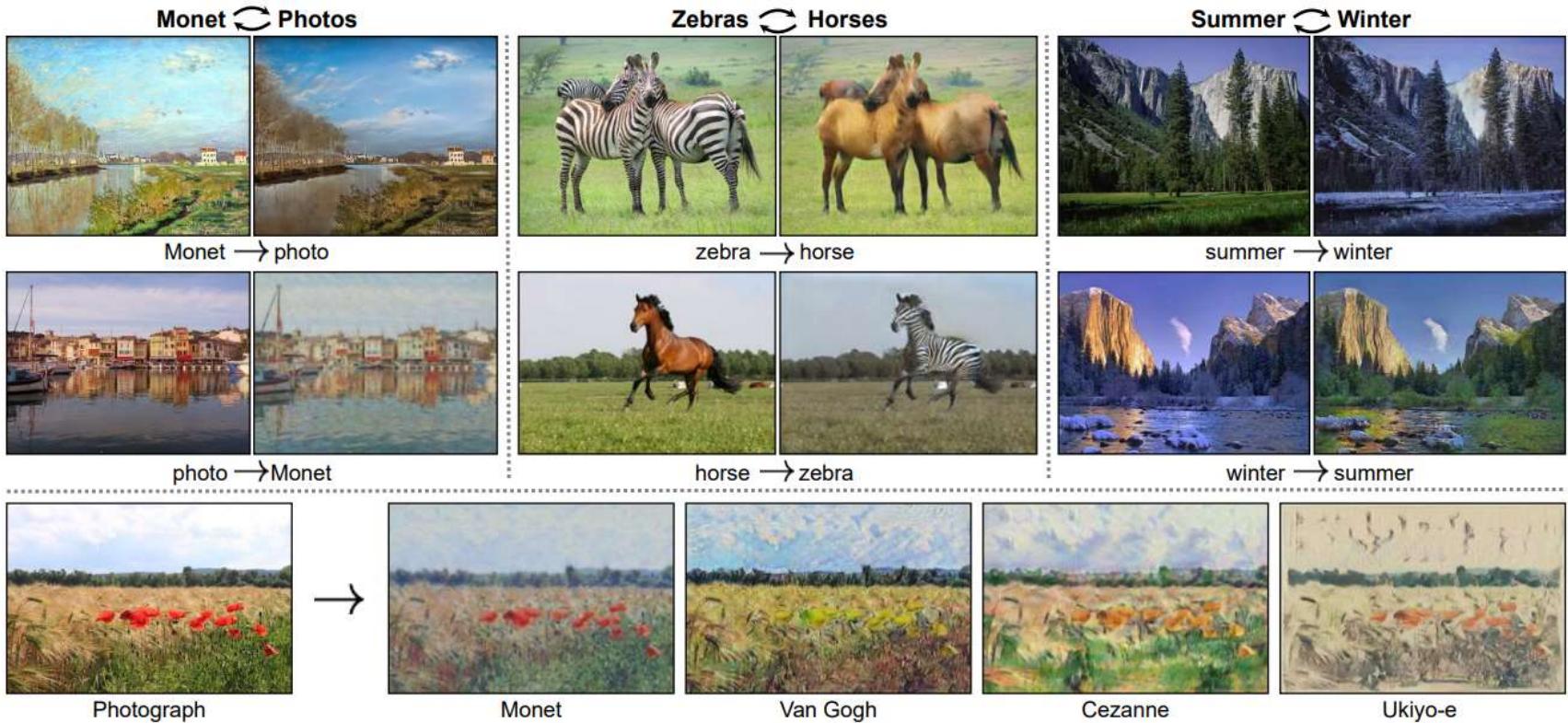
- Требует размеченные попарно данные

CycleGAN

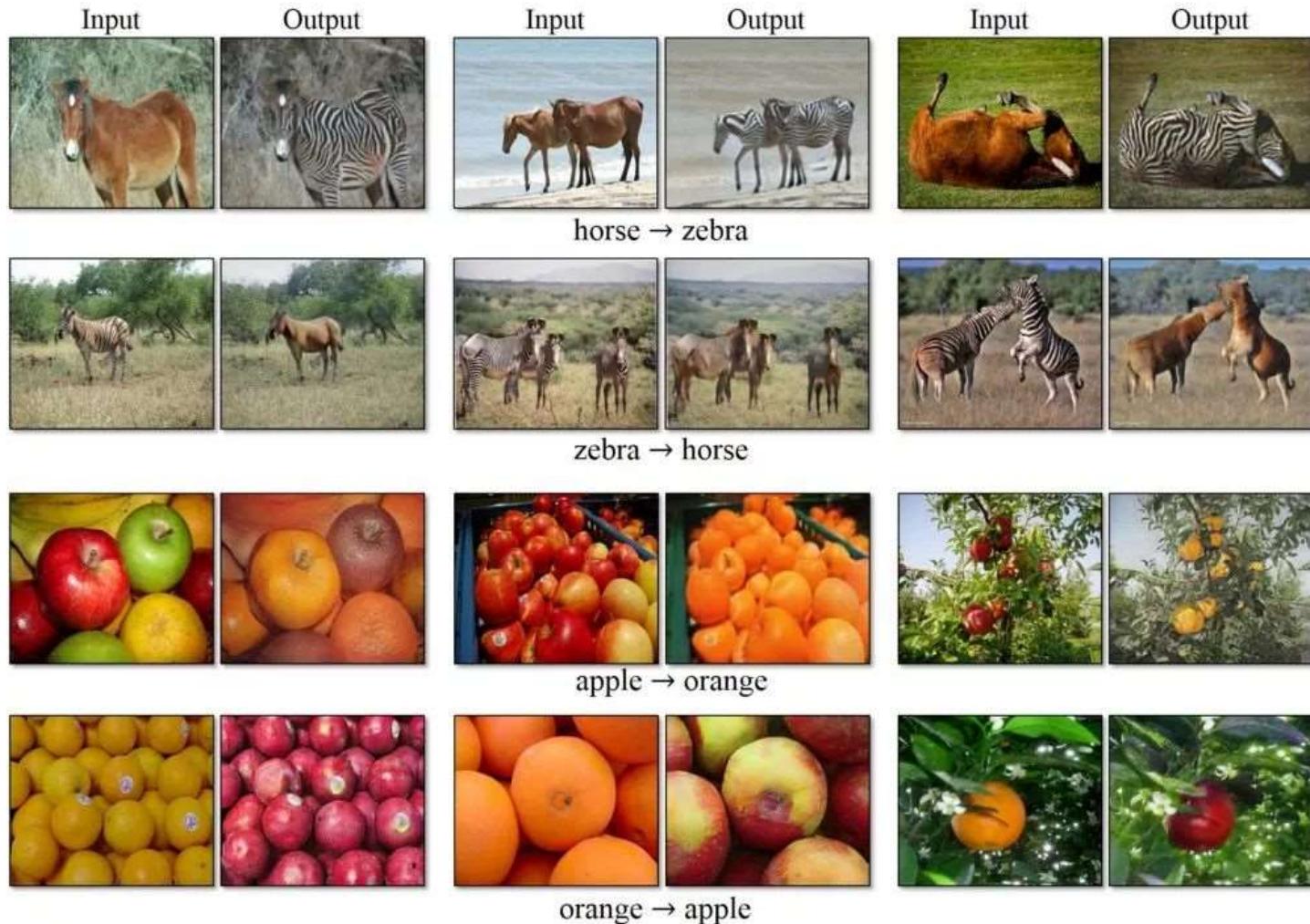


Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." *arXiv preprint*(2017).

CycleGAN



CycleGAN



CycleGAN



CycleGAN



CycleGAN



Детали реализации CycleGAN

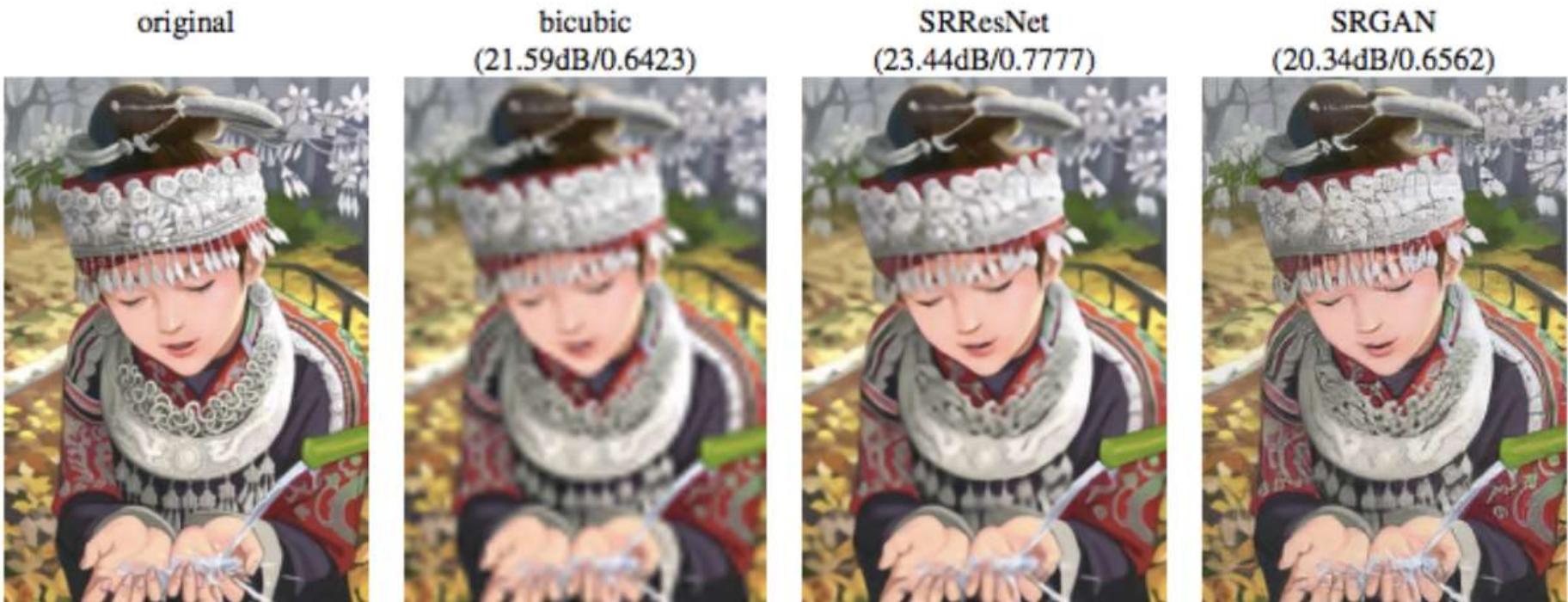
Архитектура:

- Можно использовать разные генераторы (сети преобразования)
- CycleLoss – в CycleGAN это попиксельная L1 ошибка
 - Но можно использовать и другие! Какие?
- Дискриминатор: PatchGAN

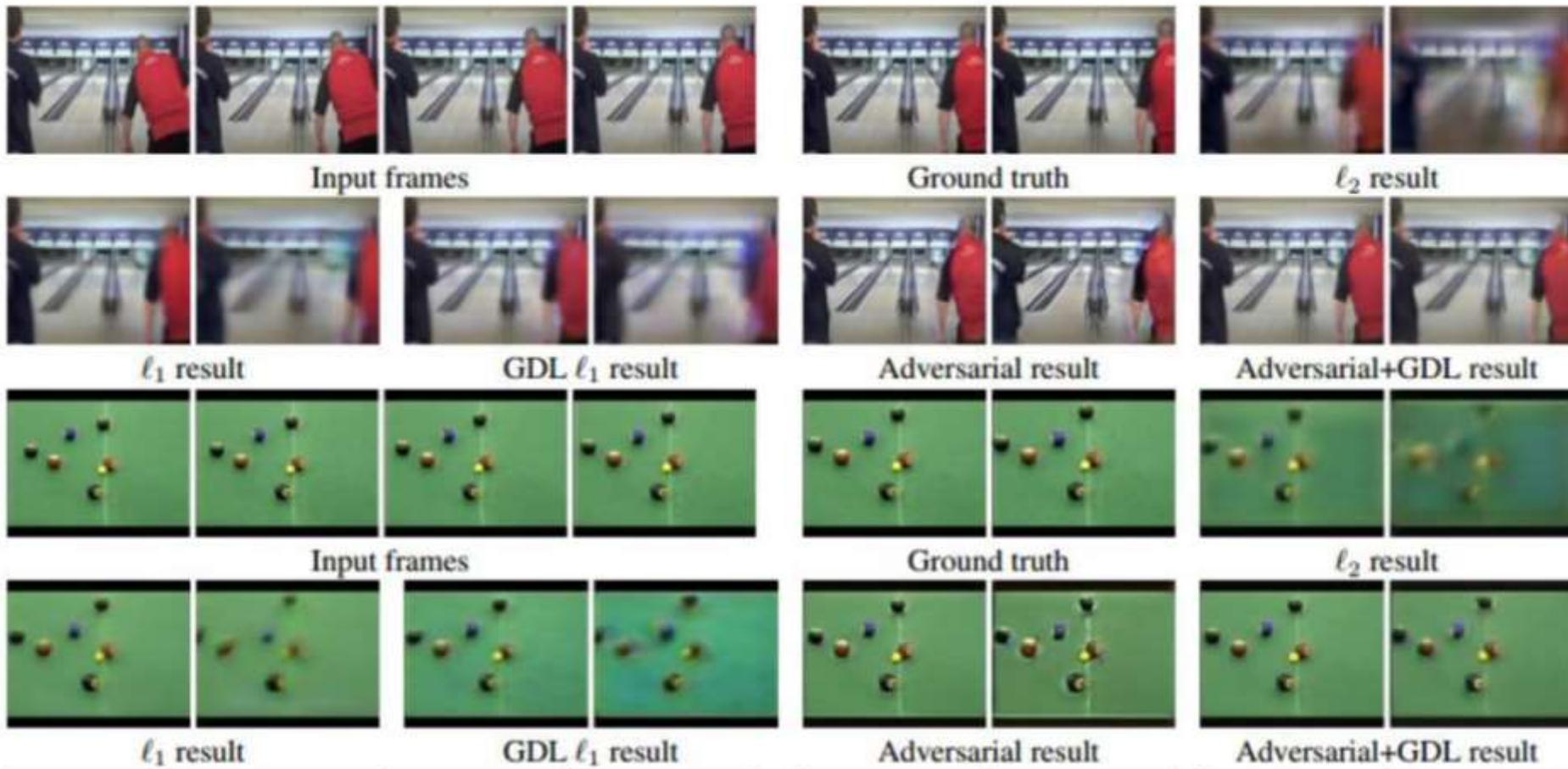
Ограничение CycleGAN:

- Преобразование из одного source домена в единственное изображение target домена

Применение - суперразрешение



Применение - предсказание кадров



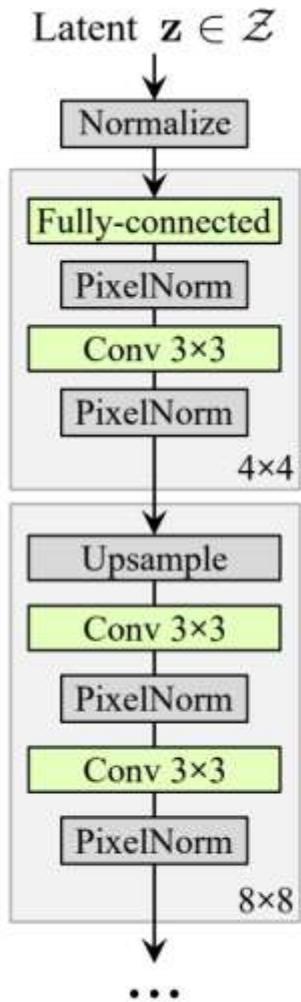
Mathieu et al. Deep multi-scale video prediction beyond mean square error, ICLR 2016 (<https://arxiv.org/abs/1511.05440>)

Применение – domain adaptation

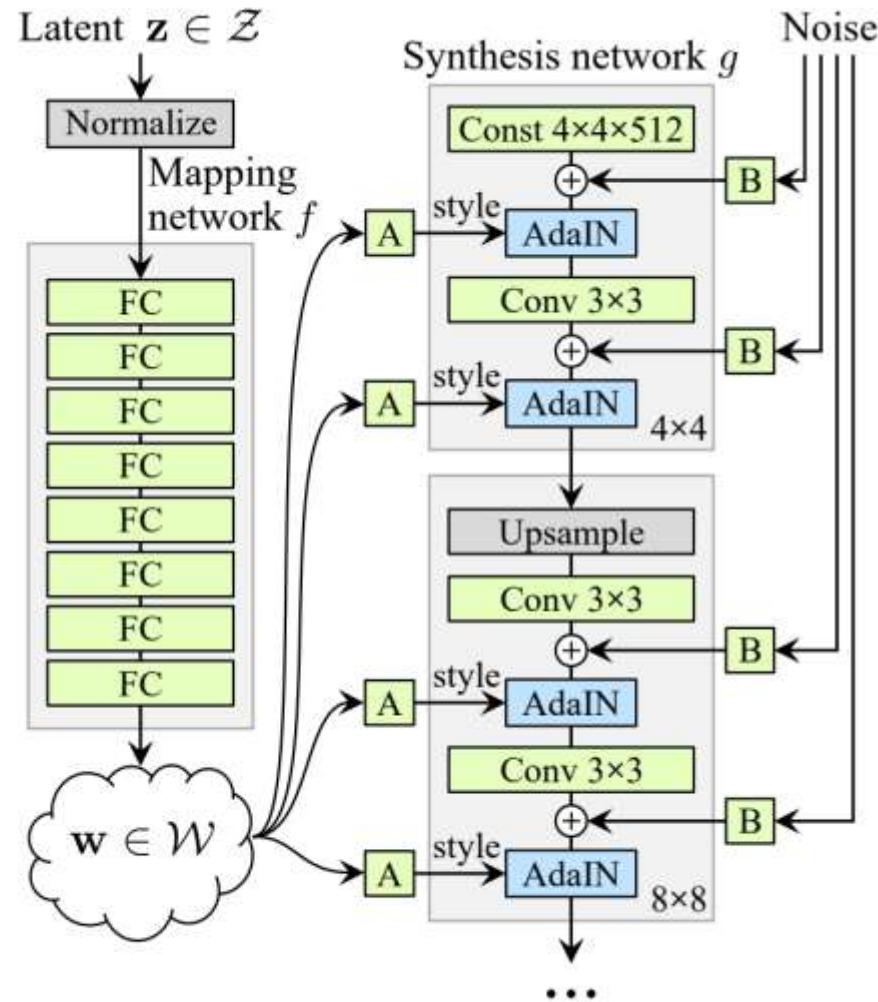


Jian Liu, Ajmal Mian. Learning Human Pose Models from Synthesized Data for Robust RGB-D Action Recognition. <https://arxiv.org/abs/1707.00823>

Style-Based GAN



(a) Traditional



(b) Style-based generator

Instance Normalization

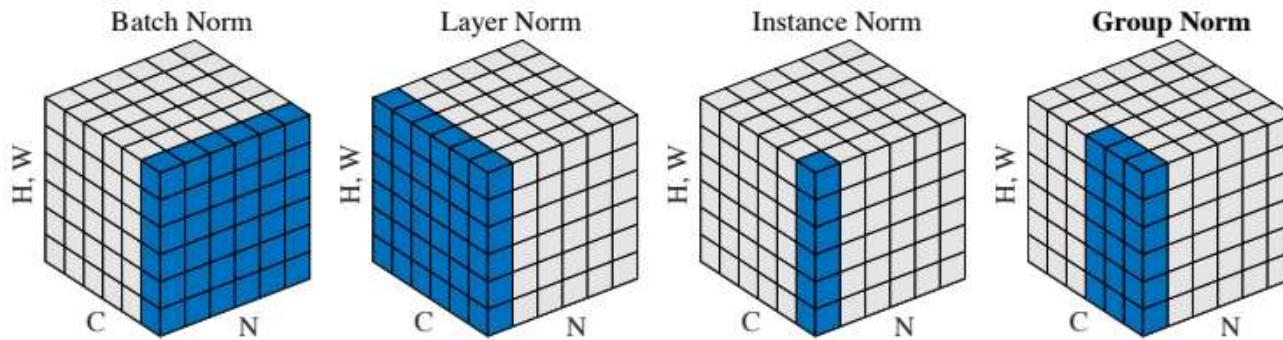


Figure 2. **Normalization methods.** Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

Adaptive Instance Normalization: control normalization by specifying y parameters

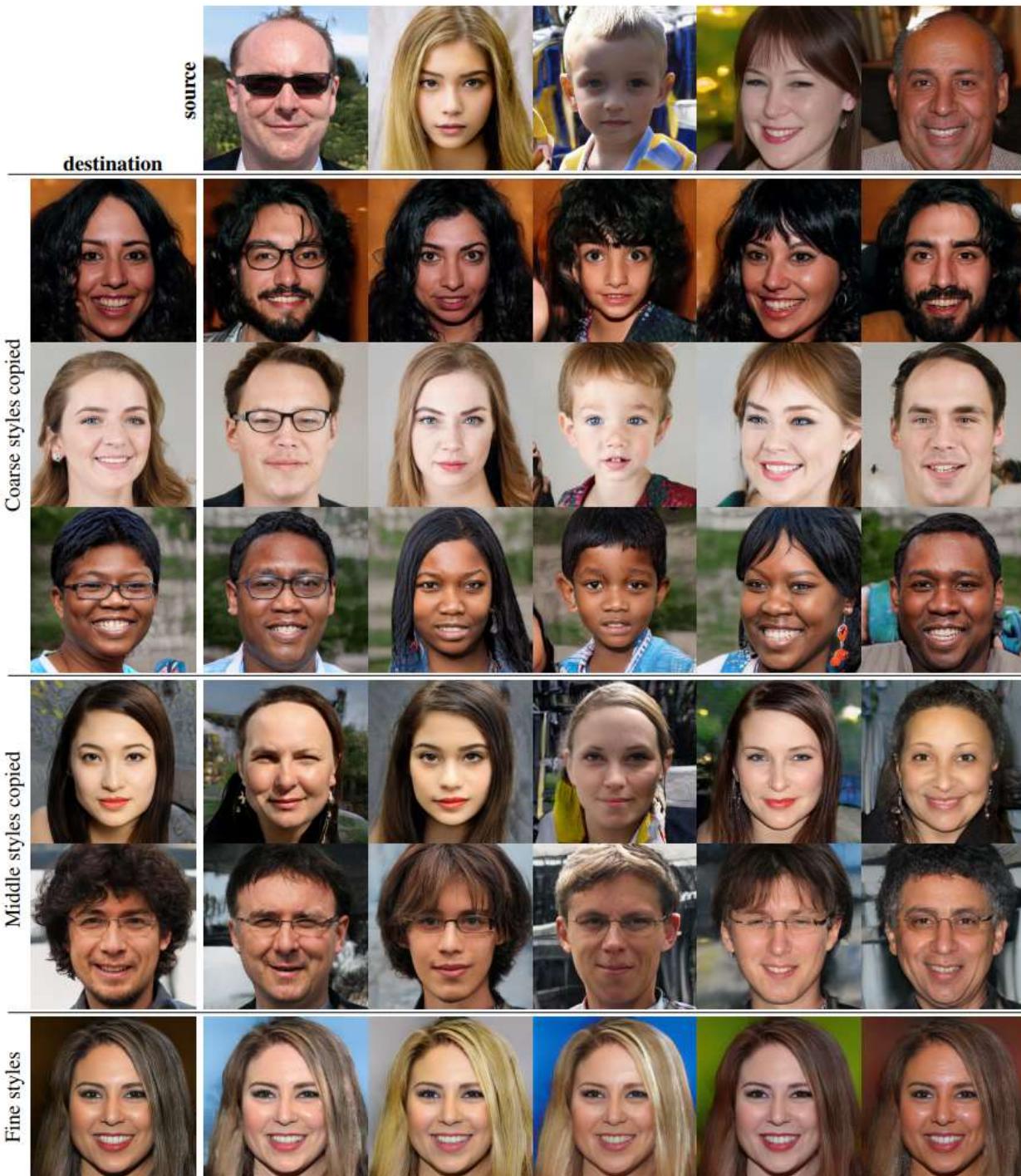
$$\text{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$

Mixing styles при обучении:

- До определённой точки код w_1
- После этой точки код w_2

Это учит сеть, что
 w
нескореллированы
на разных уровнях

Можем смешивать
после этого коды
разных
изображений



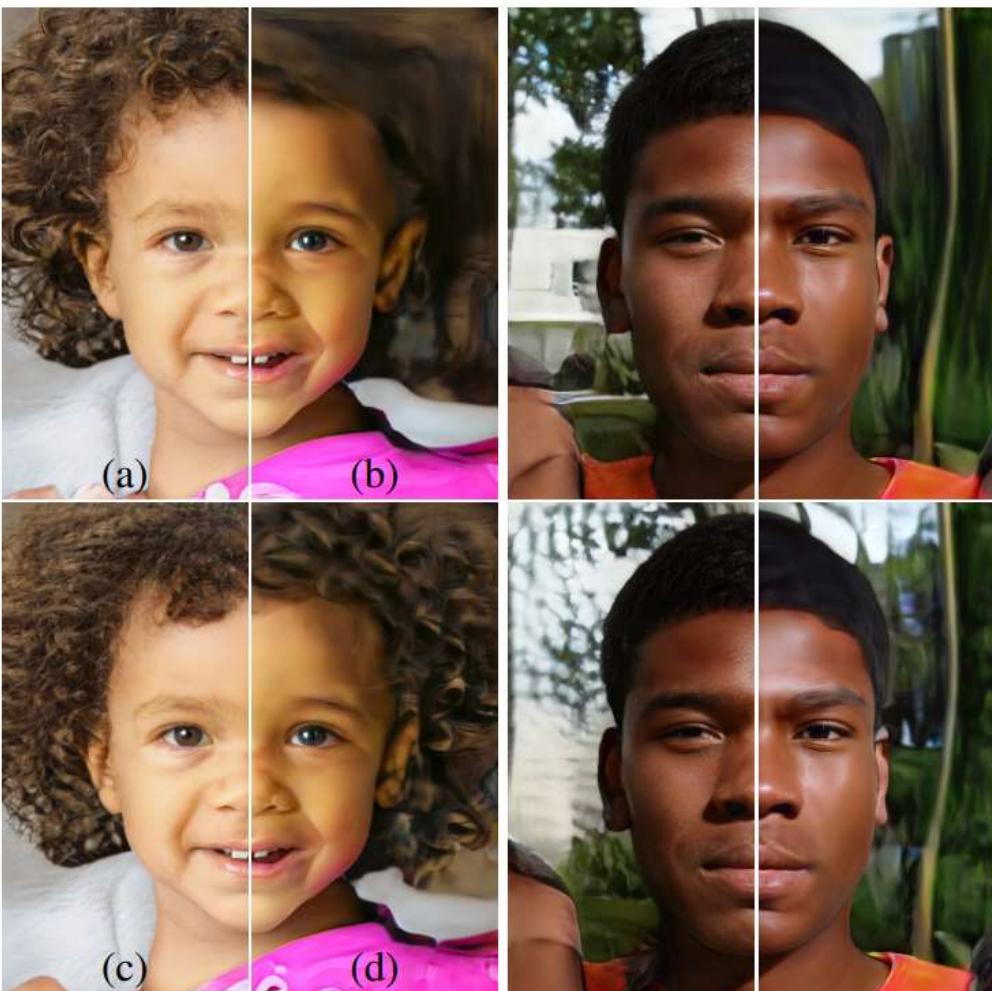


Figure 5. Effect of noise inputs at different layers of our generator. (a) Noise is applied to all layers. (b) No noise. (c) Noise in fine layers only ($64^2 - 1024^2$). (d) Noise in coarse layers only ($4^2 - 32^2$). We can see that the artificial omission of noise leads to featureless “painterly” look. Coarse noise causes large-scale curling of hair and appearance of larger background features, while the fine noise brings out the finer curls of hair, finer background detail, and skin pores.

Примеры работы



Figure 10. Uncurated set of images produced by our style-based generator (config F) with the LSUN BEDROOM dataset at 256^2 .

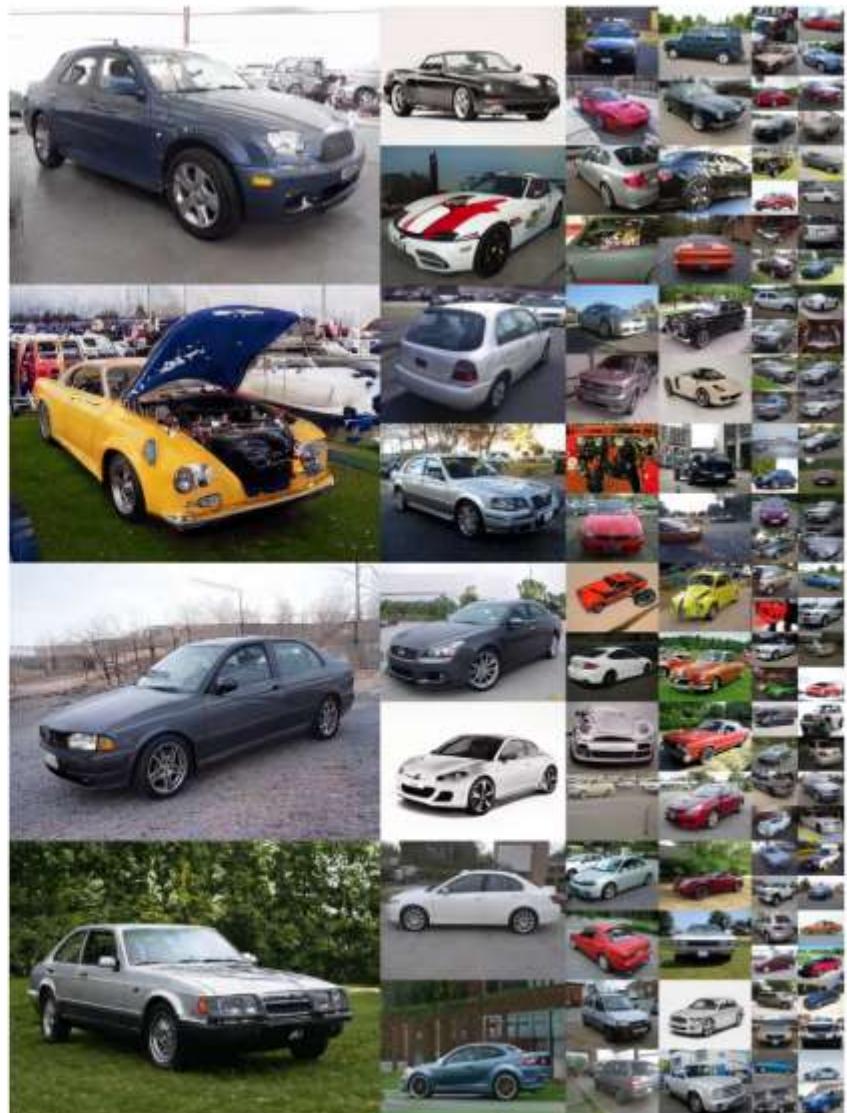
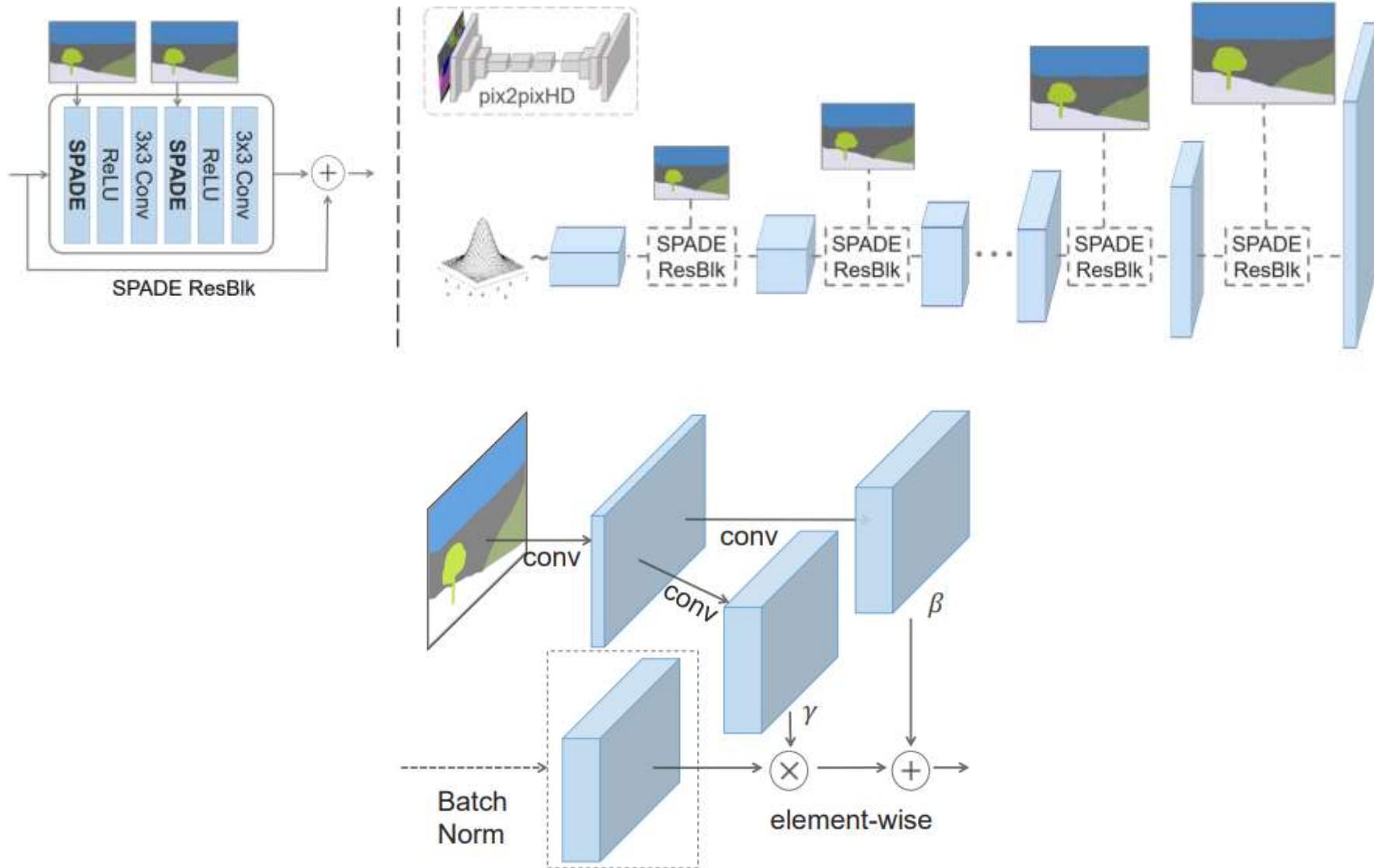
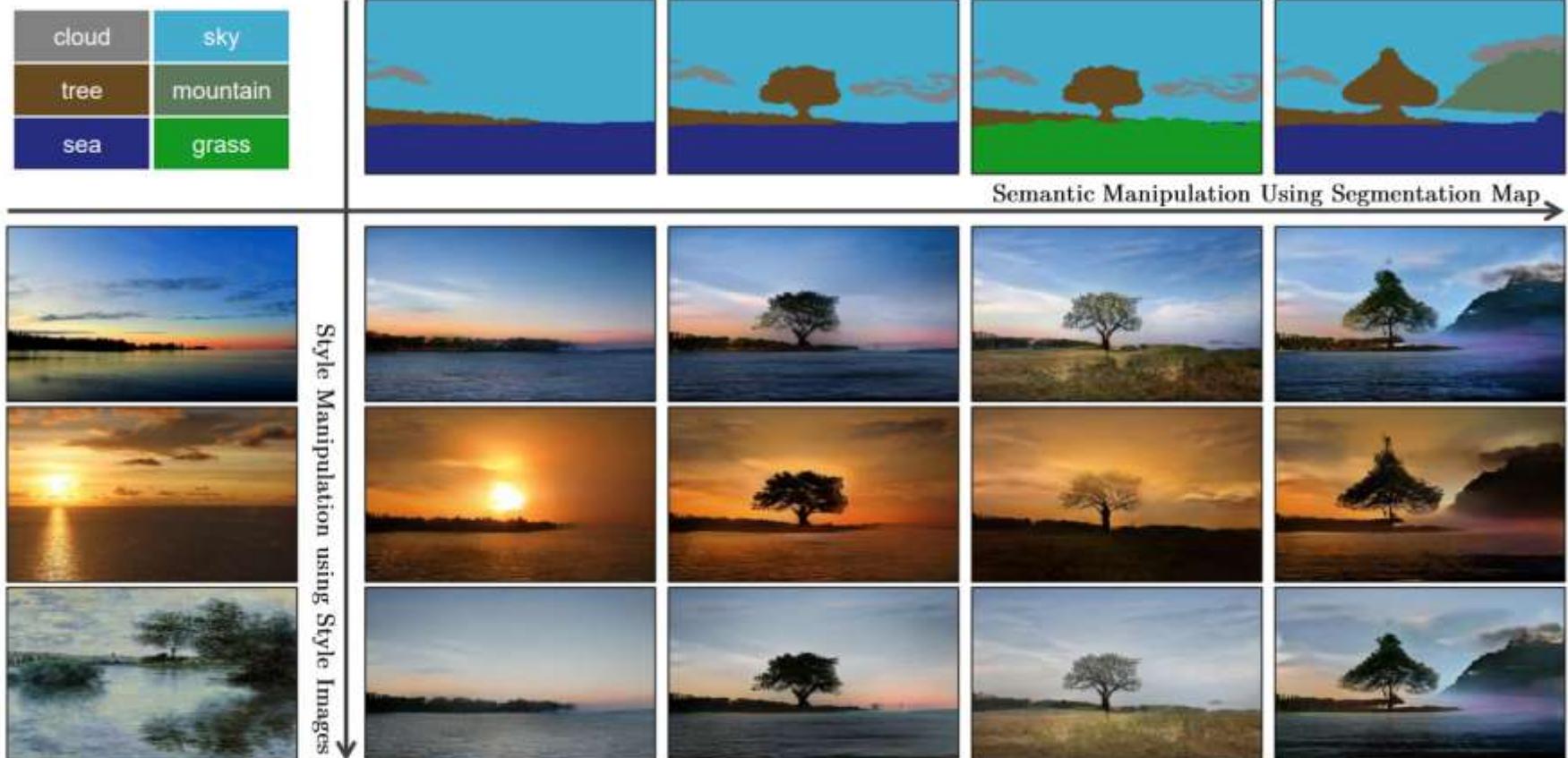


Figure 11. Uncurated set of images produced by our style-based generator (config F) with the LSUN CAR dataset at 512×384 .

SPADE = Spatially-Adaptive Normalization



Примеры работы



Резюме лекции

- Реалистичность изображения можно оценить с помощью сети-дескриминатора
- Похожесть изображений хорошо оценивается с помощью Perception Loss – сравнения значений нейросетевых признаков
 - М.б. Identity Loss – если признак кодируют «личность»
- Style Loss – функция от признаков, описывающие статистики распределения признаков
- Можем проводить линейные преобразования в пространстве признаков изображения
- Cycle Loss позволяет работать с произвольными выборками, а не с парами соответствующих изображений
 - Альтернатива – вариант PerceptionLoss для входа/выхода