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

Курс «Компьютерное зрение»

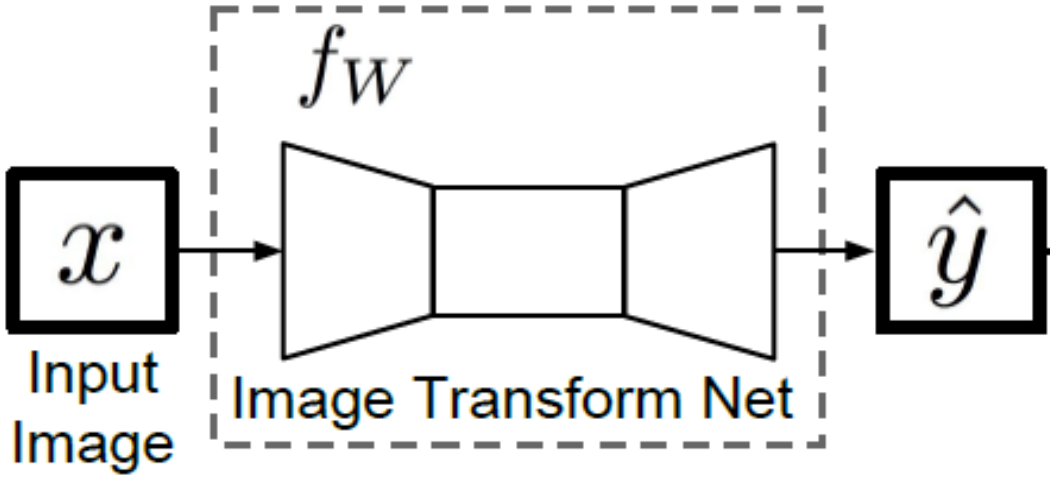
«Синтез изображений, ч.1»

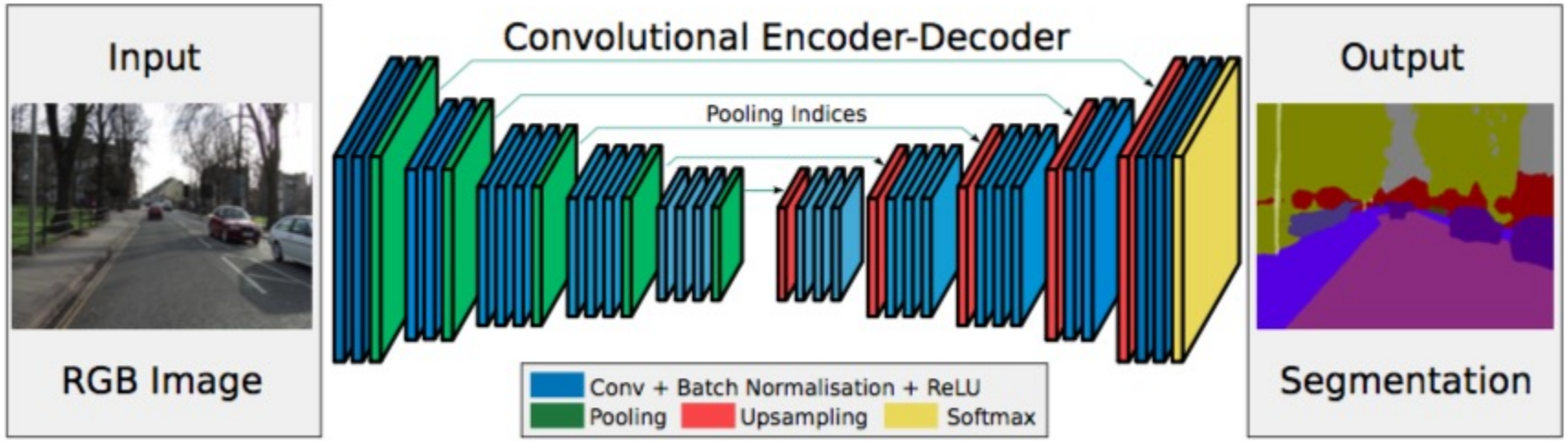
Антон Конушин и Тимур Мамедов

2025 год

Модификация изображений

Пример решения - преобразователи



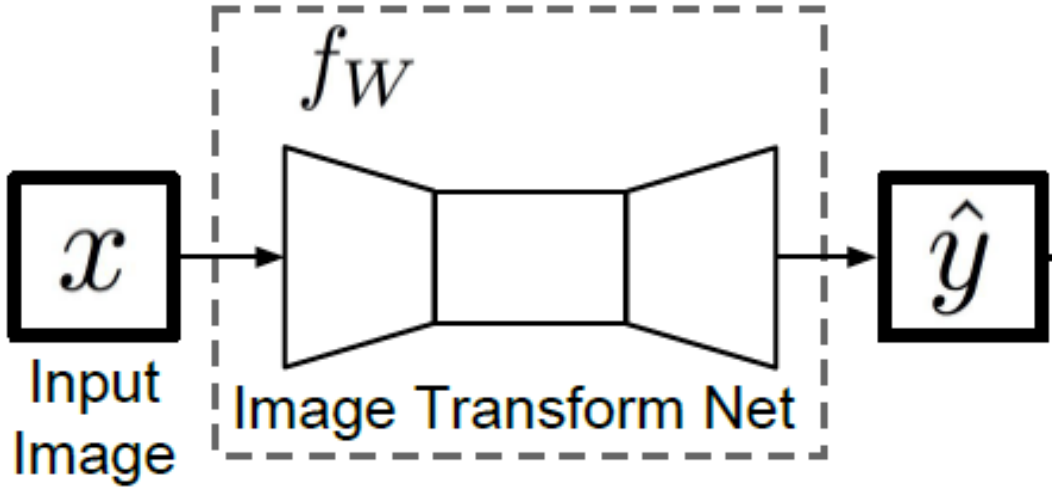


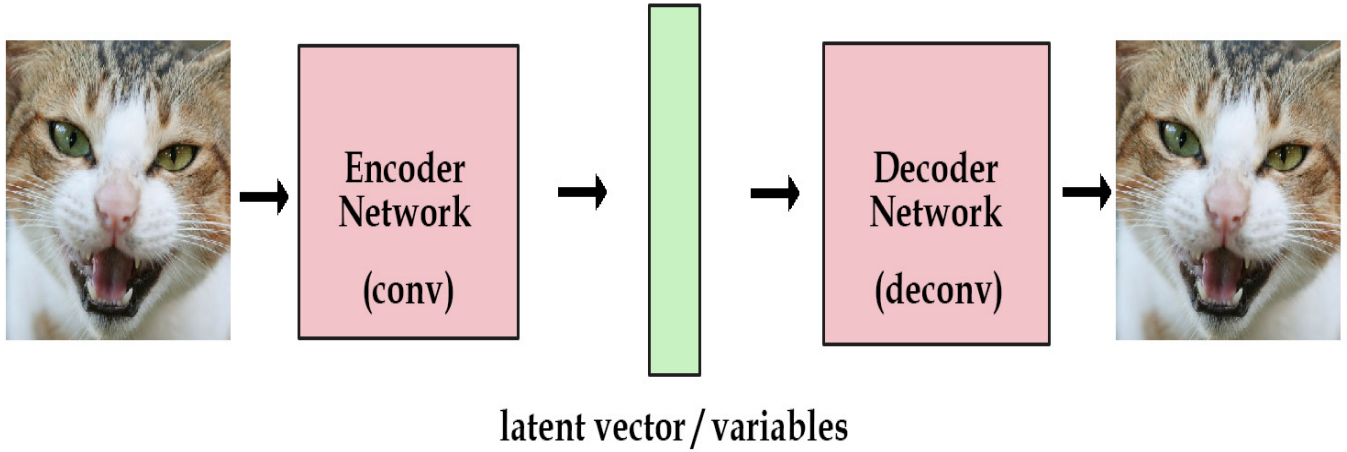
Перенос стиля изображения

+

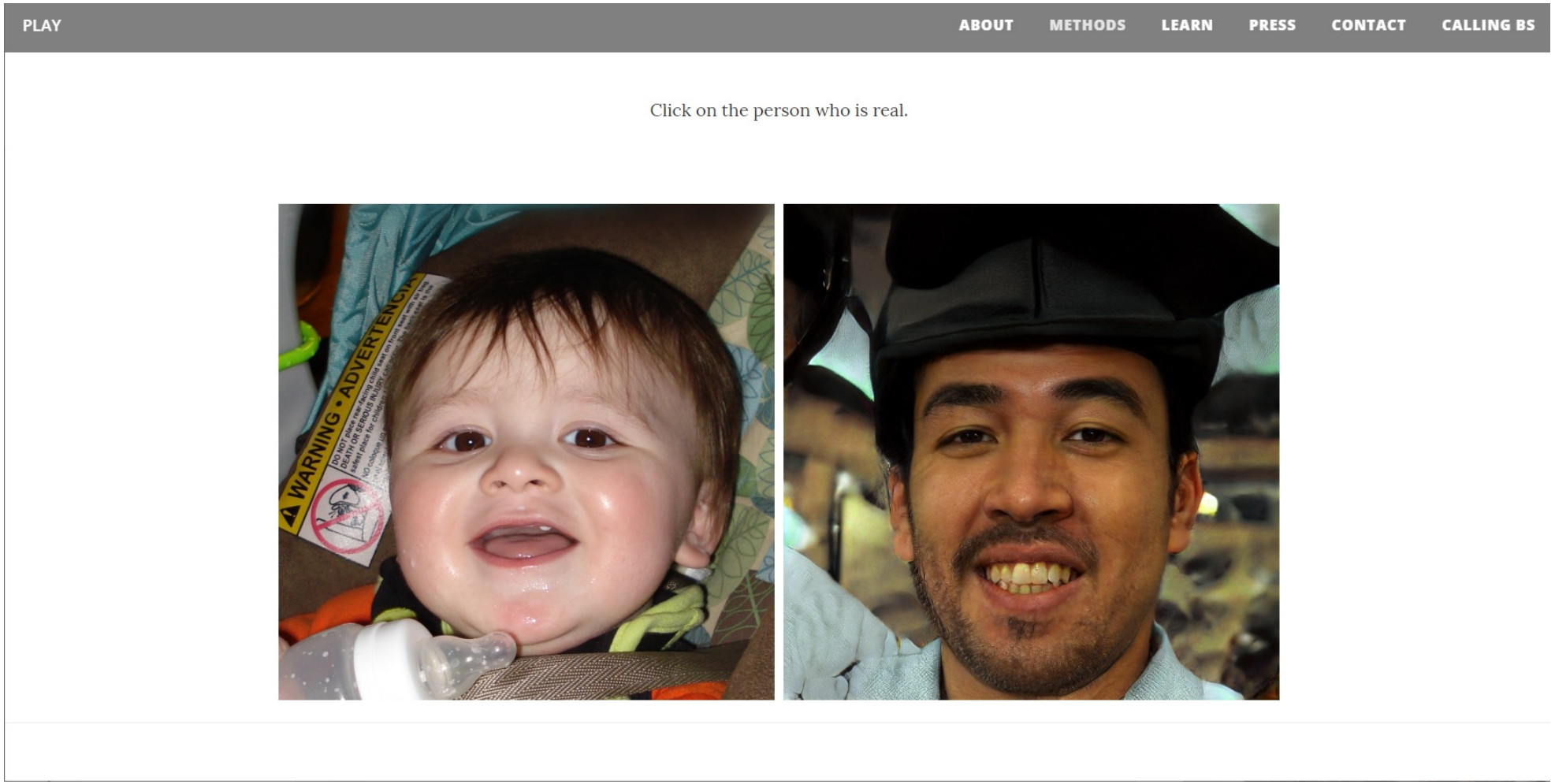
=

Подходы к решению





Синтез изображений

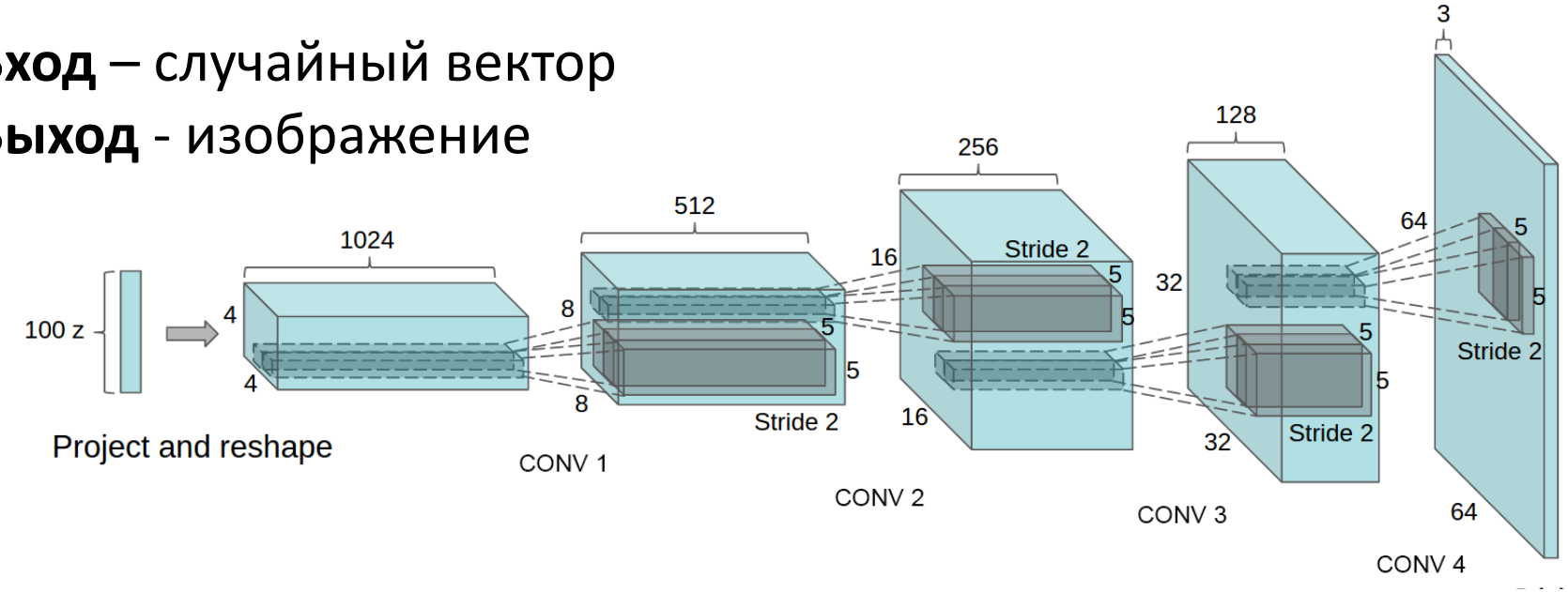


http://www.whichfaceisreal.com/

Пример решения - DCGAN

Вход – случайный вектор

Выход - изображение



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.

A.Radford, L. Metz, S. Chintala UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS. ICLR 2016

Визуализация признаков и perception loss

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 по x следующий функционал l :

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

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

$R(x)$– регуляризатор, например, Total Variation (TV):

$$R(x) = \sum_{i,j}((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2)$$

- Оптимизируем функционал градиентным спуском

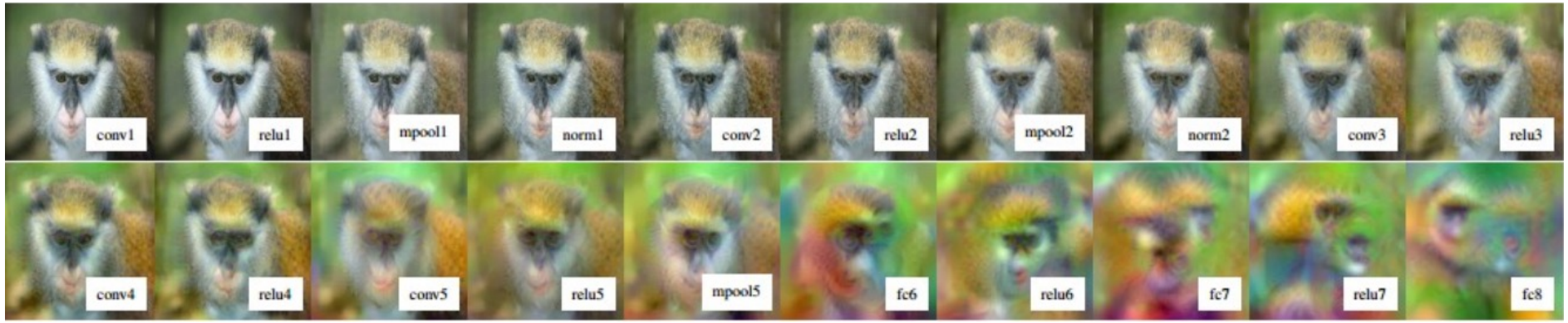
Реконструкция

С последнего свёрточного слоя

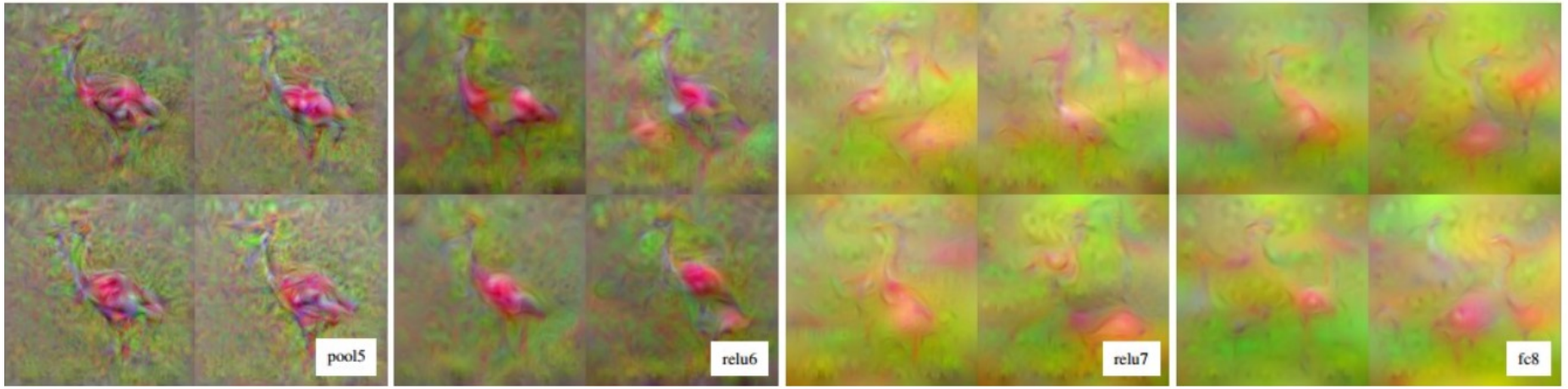
Пространственная информация во многом сохраняется

Реконструкция

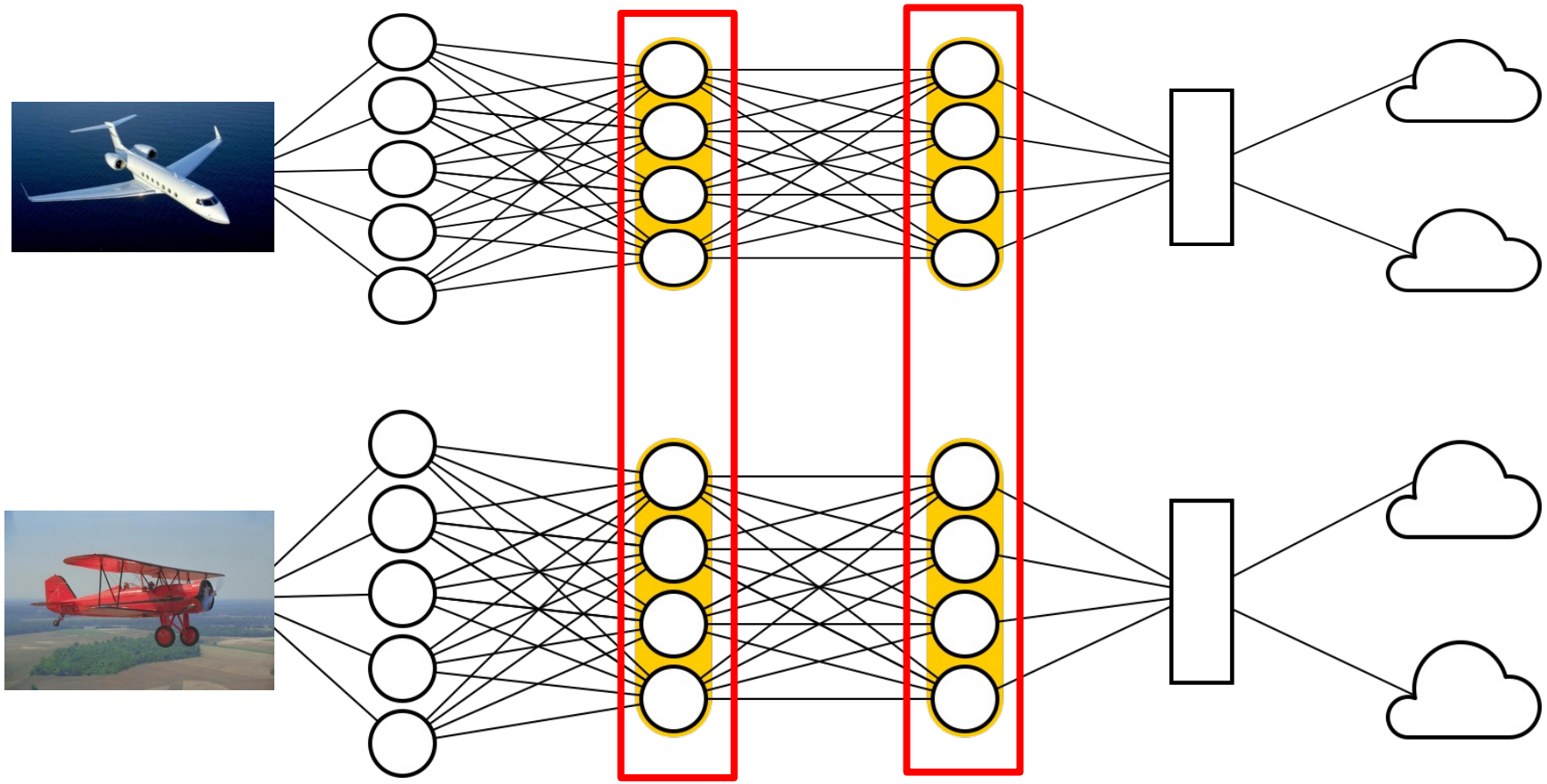
С разных уровней



Множественные реконструкции (разные изображения с одинаковыми признаками)



Можем использовать для сравнения изображений



Perception loss

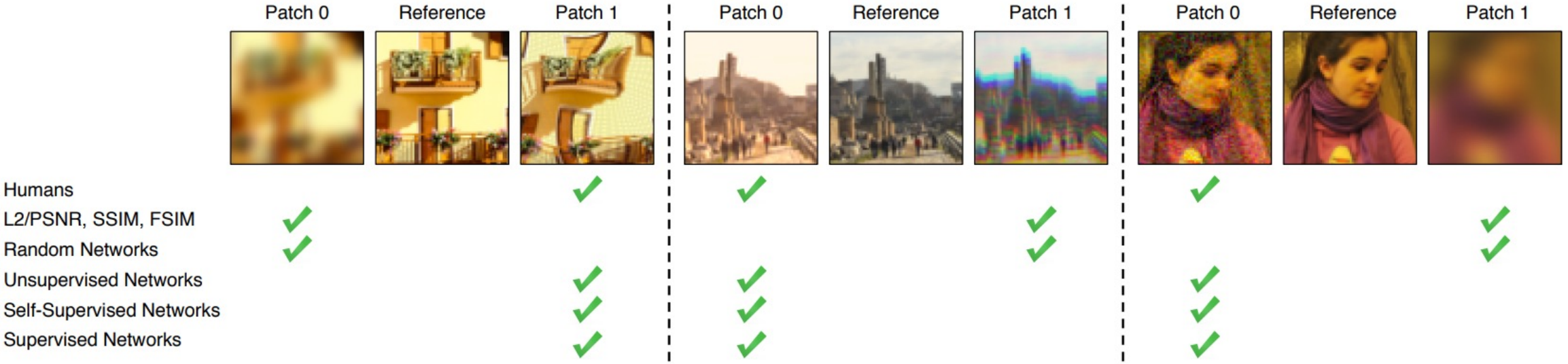
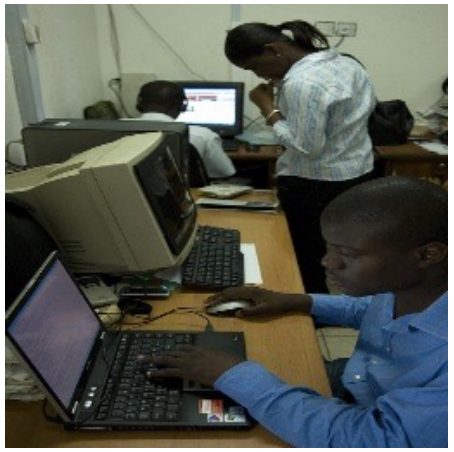


Figure 1: **Which patch (left or right) is "closer" to the middle patch in these examples?** In each case, the traditional metrics (L2/PSNR, SSIM, FSIM) disagree with human judgments. But deep networks, even across architectures (Squeezenet [20], AlexNet [27], VGG [52]) and supervision type (supervised [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`.

Zhang et.al. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. CVPR2018

Применим для переноса стиля

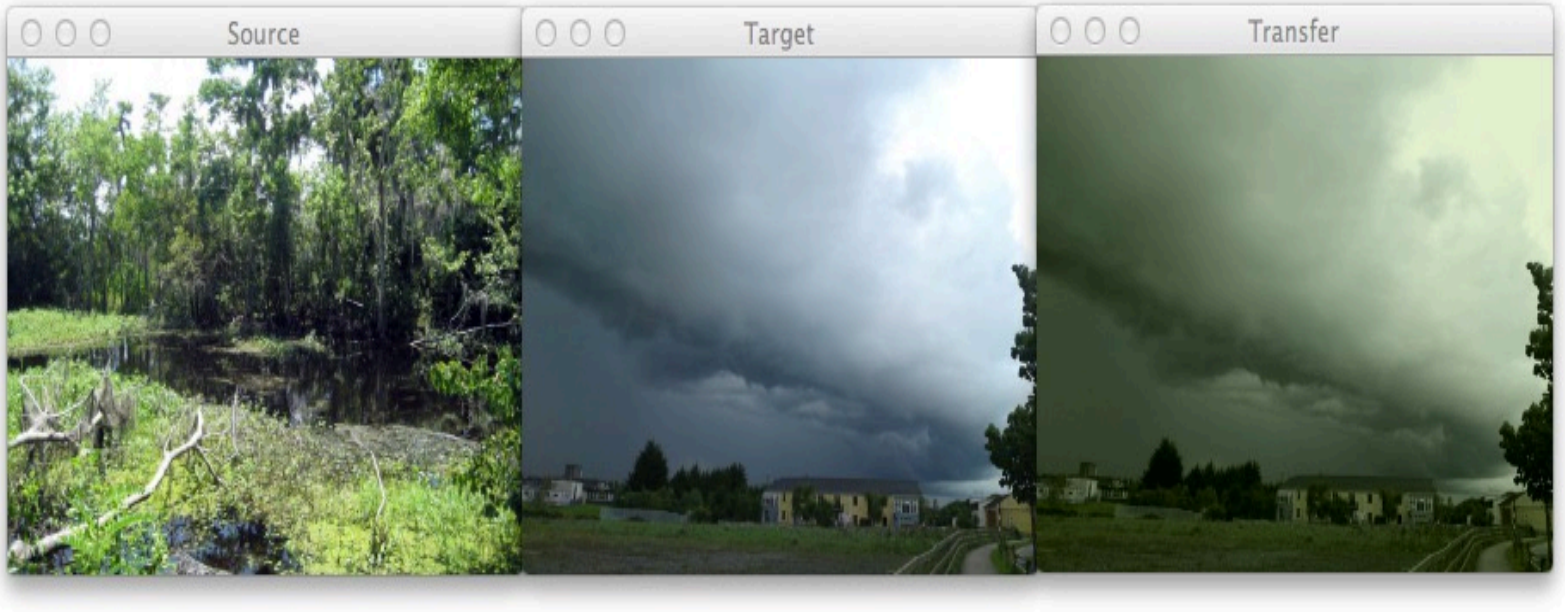
+



=

Простейшие методы переноса стиля

Простейшая стилизация – перенос цвета. Например, среднее и дисперсия по каждом каналу приравнивается к изображению «стиля».



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

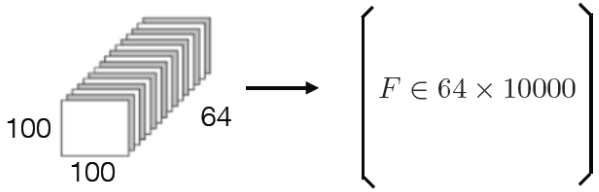
Как описать стиль?

- 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 изображений с заданным стилем

- Стиль можно описать корреляцией откликов фильтров, записав матрицу Грама
- Где G^l_{ij} вычисляется как скалярное произведение откликов i-го и j-го фильтров:

$$G^l \in \mathcal{R}^{N_l \times N_l}$$



$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$$

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

$$E_l = \frac{1}{Norm} \sum_{i,j} (G^l_{ij} - A^l_{i,j})^2$$

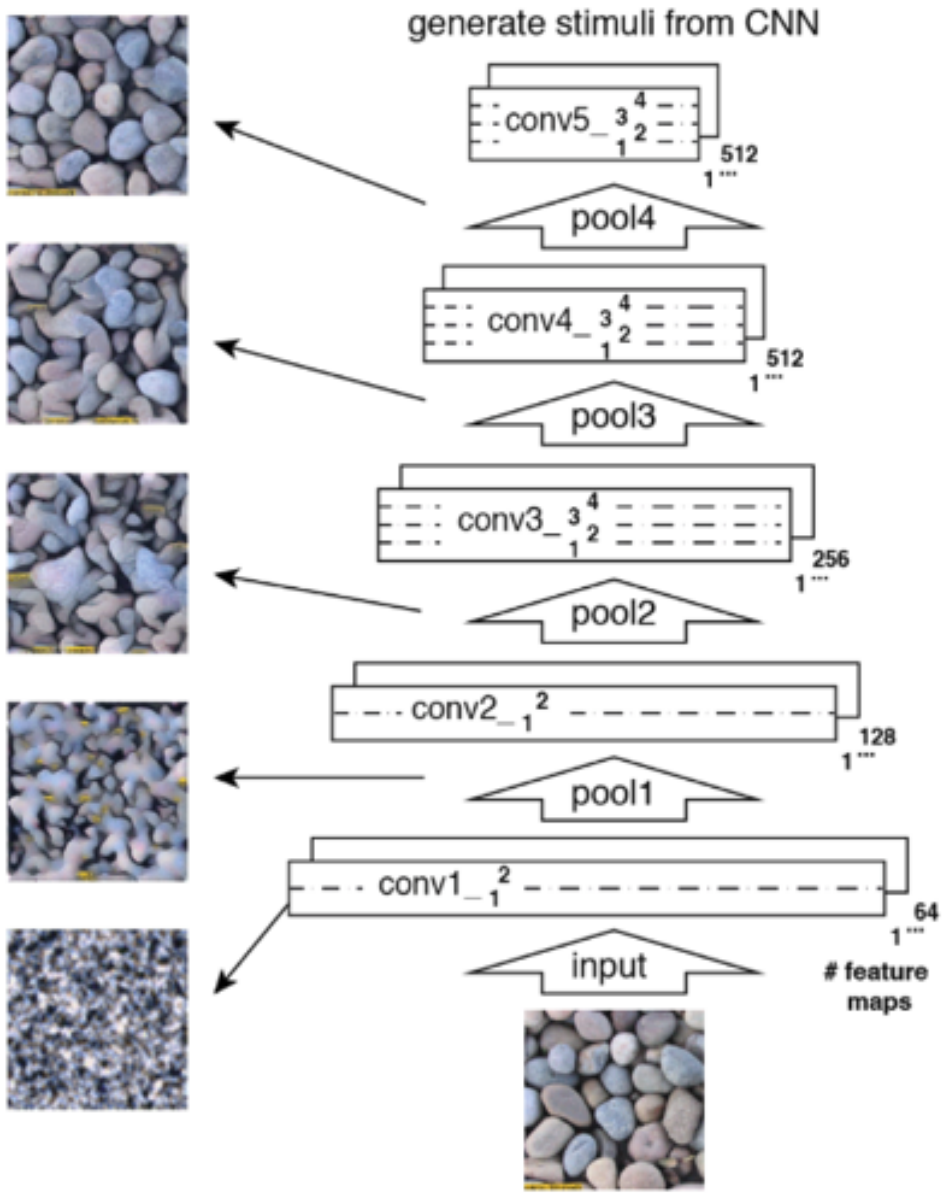
Cost for style reconstruction

$$Loss_{style} = \sum_{l=0}^{L} w_l E_l$$

Accumulate cost for lower layers

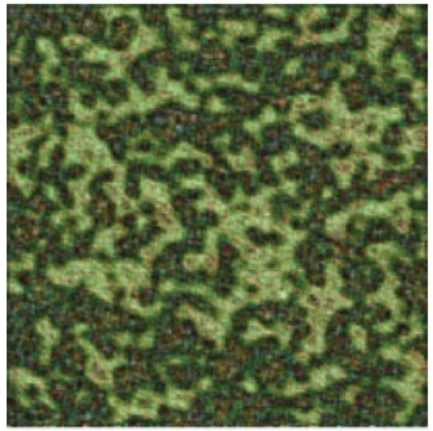
1. A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients. Portilla, J., & Simoncelli, E. P. *Int. J. Comput. Vis.* 40, 49-70 (2000)
2. Texture Synthesis and The Controlled Generation of Natural Stimuli Using Convolutional Neural Networks. *Gatys, L. A., Ecker, A. S., & Bethge, M. NIPS (2015)*

Реконструкция текстур

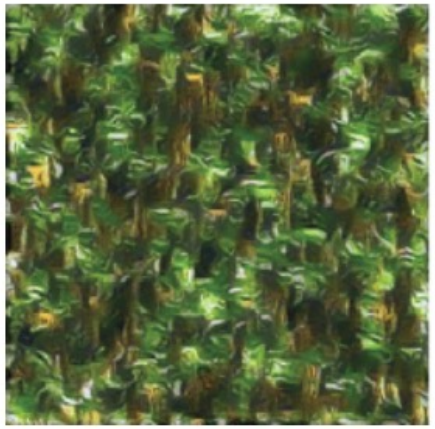


Пример реконструкции текстур

Original



Up to *Conv1_1* layer



Up to *Pool1* layer

Up to *Pool2* layer

Up to *Pool3* layer

Up to *Pool4* layer

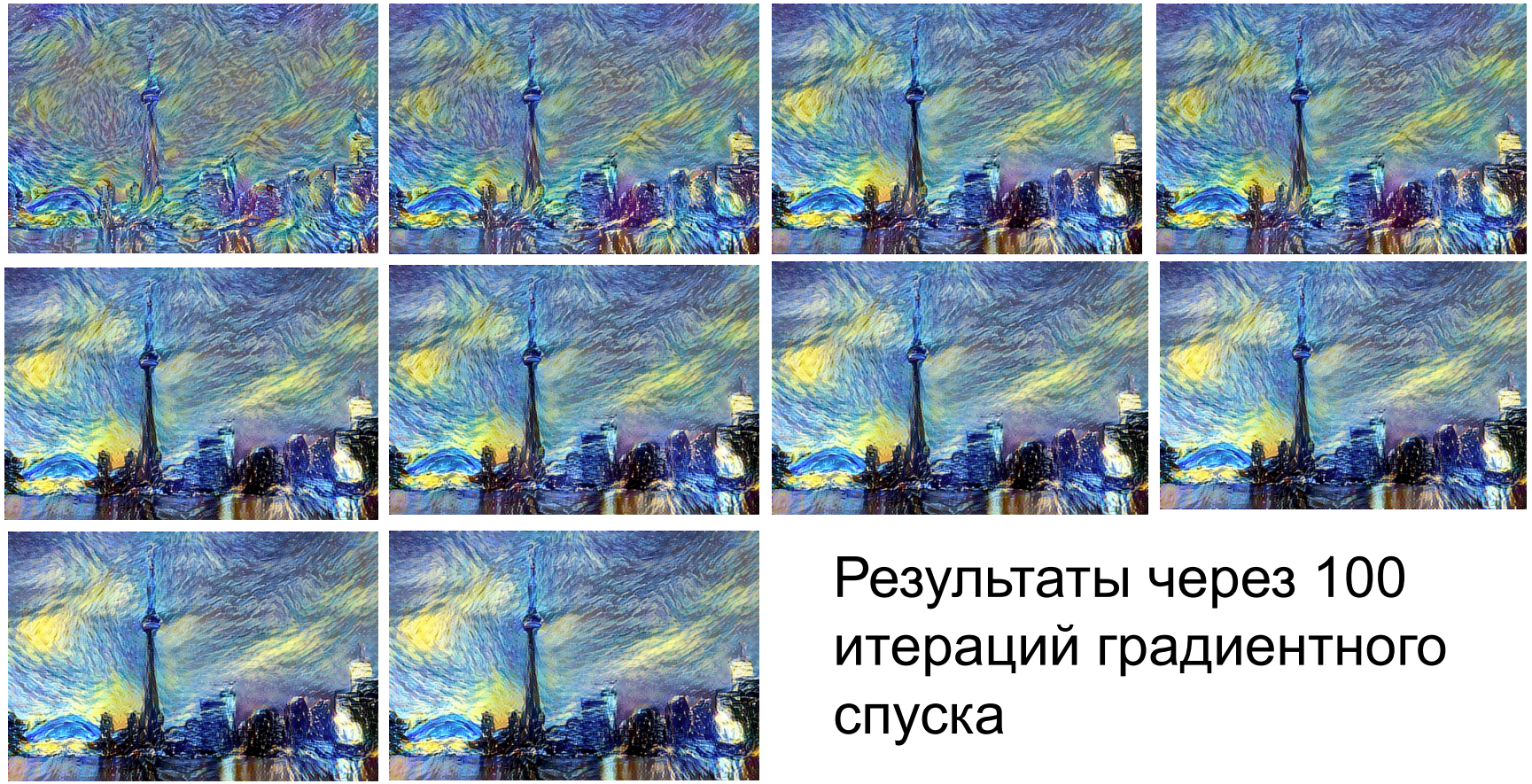
Ключевое наблюдение

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

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

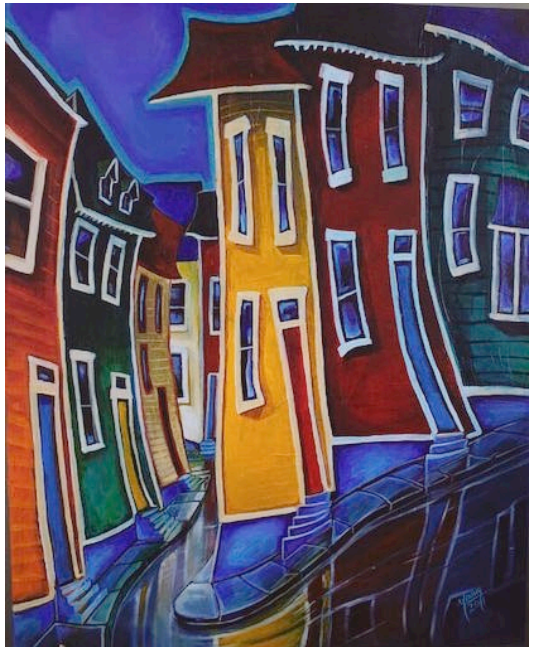
Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." *arXiv preprint arXiv:1508.06576* (2015).

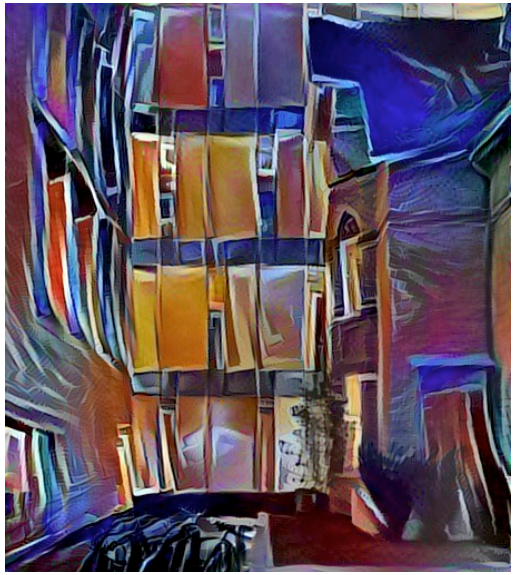
Визуализация работы



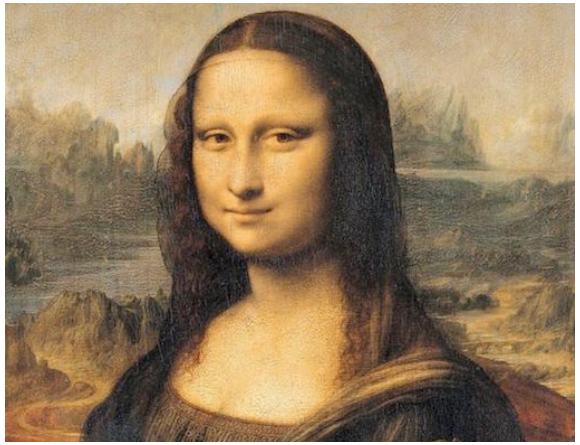
Результаты через 100 итераций градиентного спуска

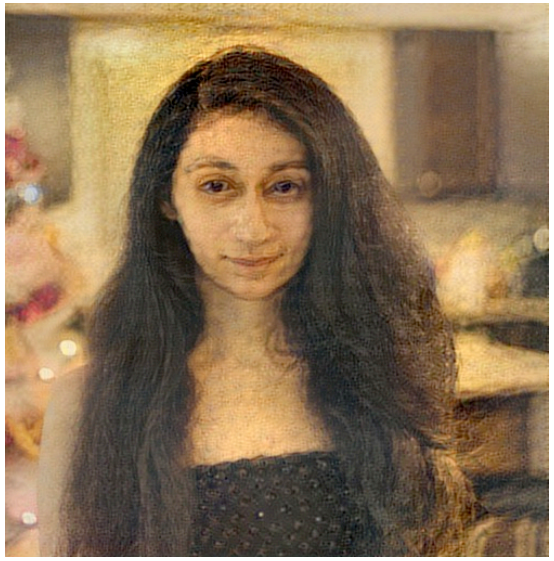
Пример работы

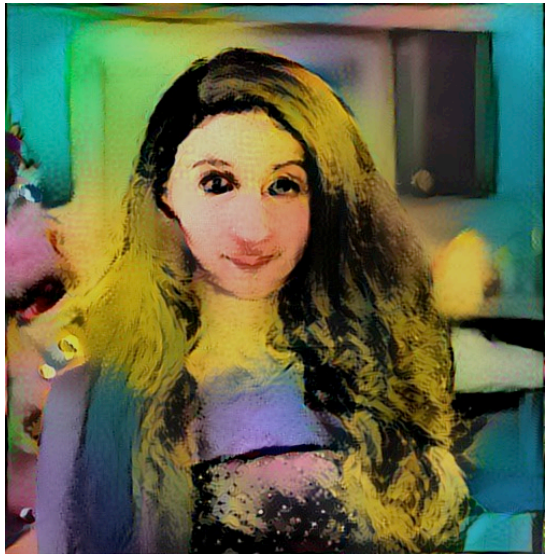




Пример работы

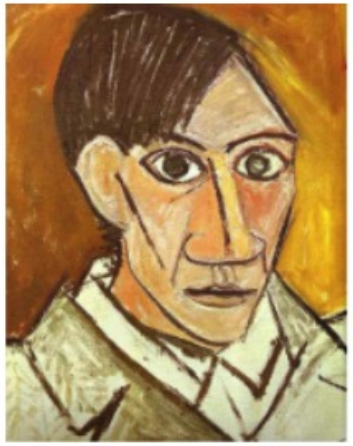






Соотношение стиля и содержания

Used for *Content*



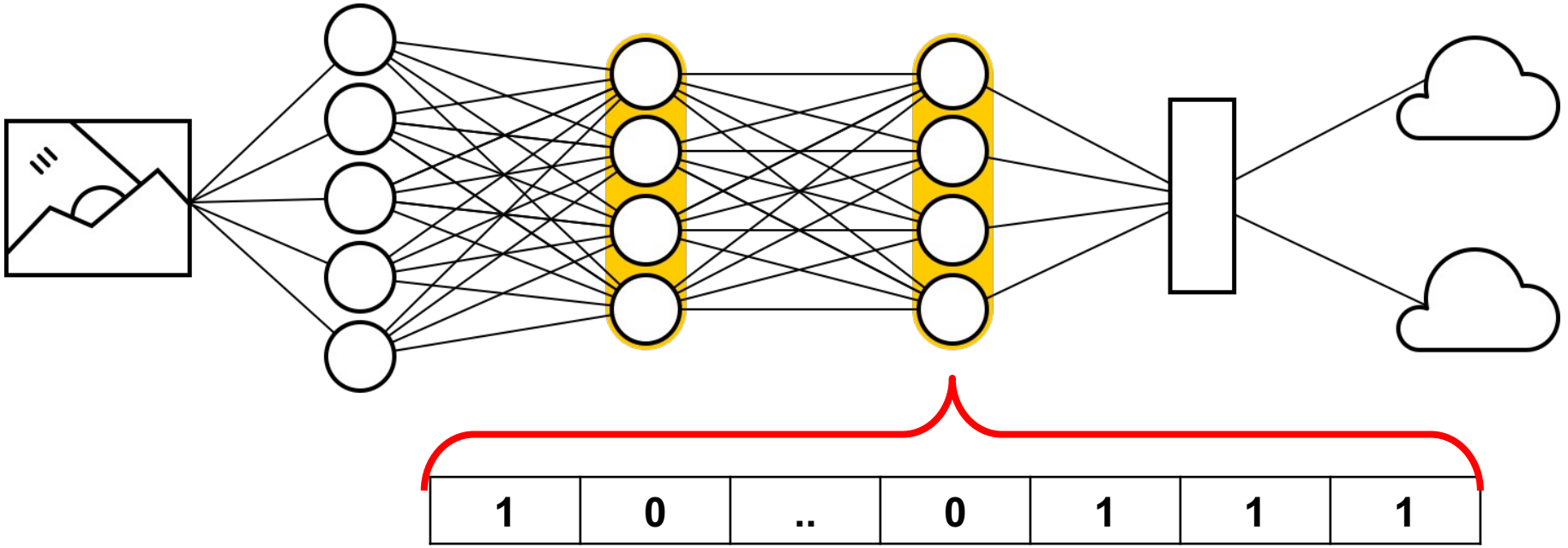
Used for *Style*

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

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

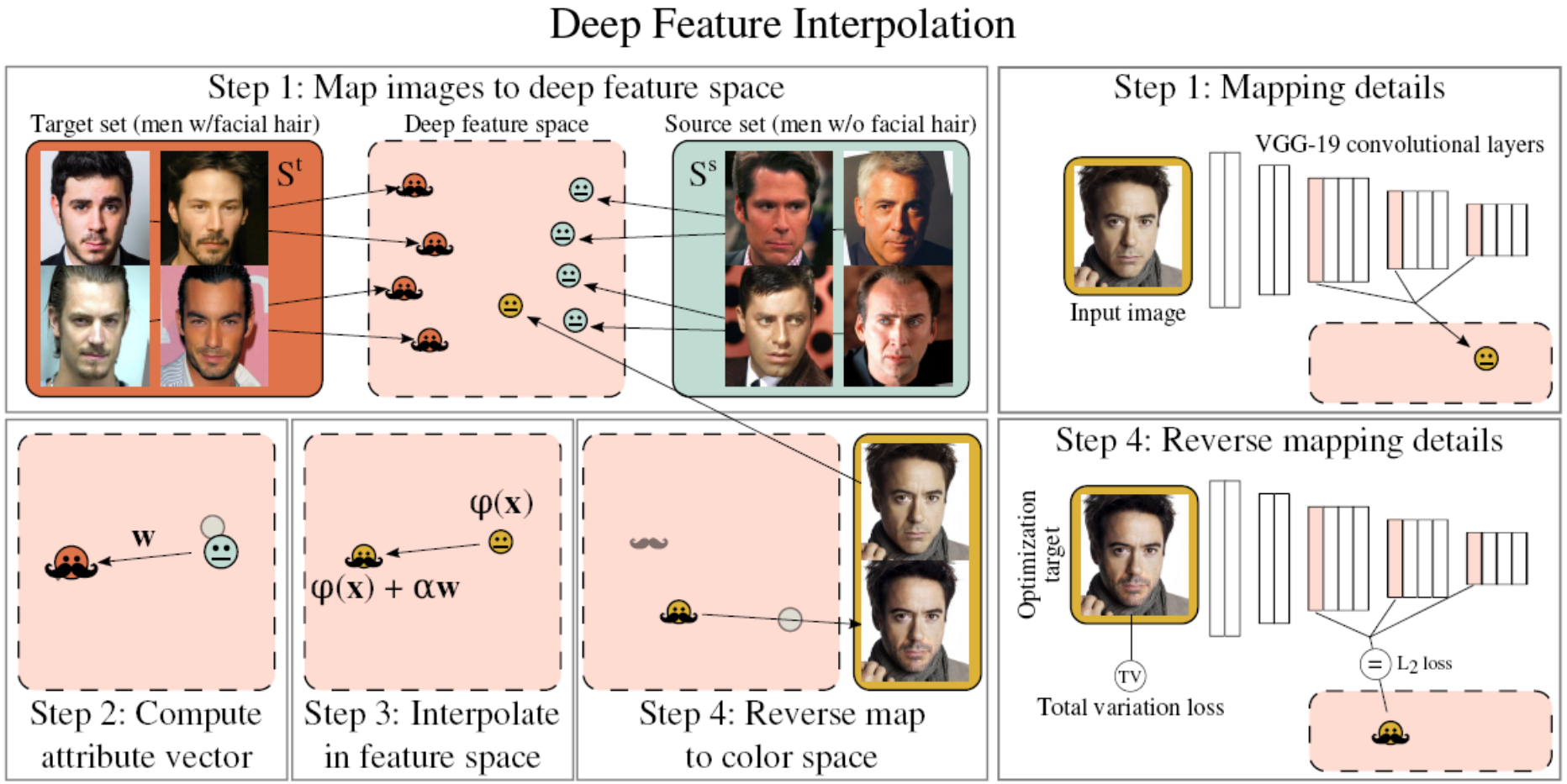


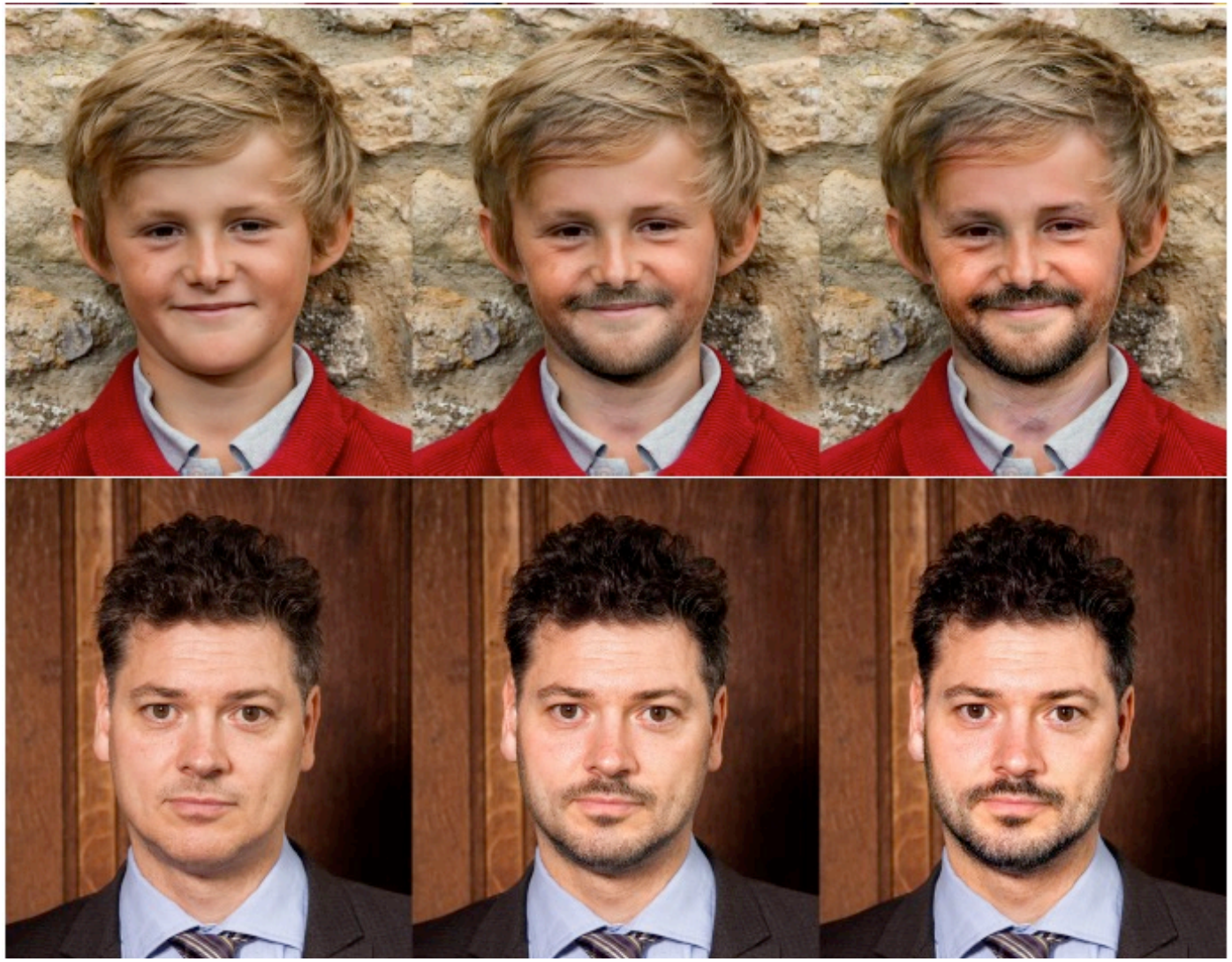
Figure 2. A schematic outline of the four high-level DFI steps.

Используем обычные VGG предобученные на ImageNet

Варьирование параметра шага

Original | $\beta = 0.1$ | $\beta = 0.2$ | $\beta = 0.4$ | $\beta = 0.6$ | $\beta = 0.8$ | $\beta = 1.0$

Пример работы на высоком разрешении



Выравнивание лиц и других атрибутов (мужчина, Caucasian)

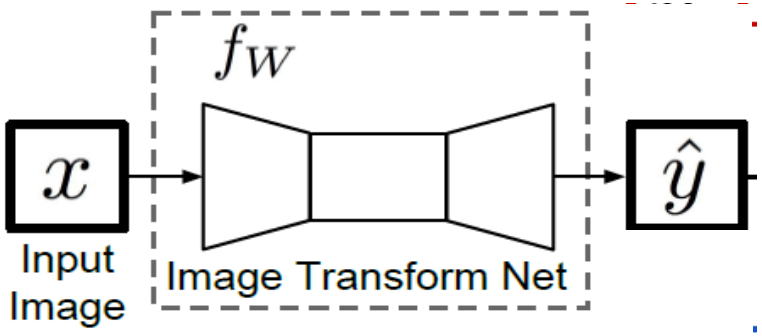
- Низкое разрешение – выравнивание глаз и рта на этапе предобработки коллекции
- Высокое разрешение – подгон лиц к тестовому перед сбором source & target набором для расчета вектора преобразования

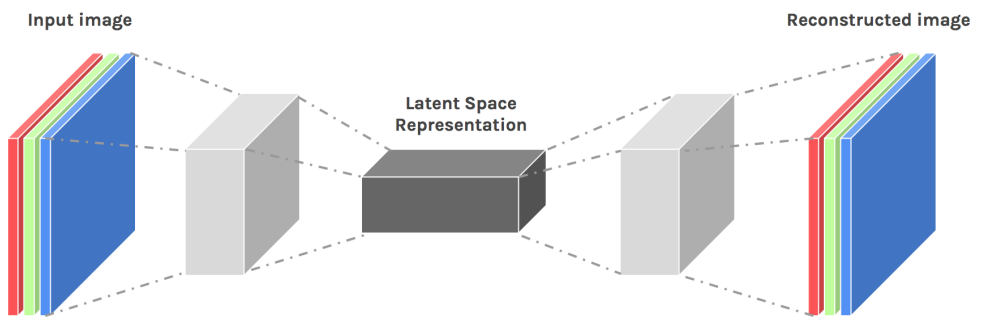
Perceptual Loss vs Identity Loss

- VGG-face, обученная для распознавания лиц, работала хуже VGG на ImageNet.
- Причина? Такая сеть учится игнорировать изменение просто «внешних» атрибутов (усов, очков), и фокусироваться на чертах лица
- Сравнение по признакам VGG-face даёт Identity Loss, т.е. оценку изменения «личности» человека
- При модификации изображений можно смешивать Perceptual Loss и Identity Loss

Стилизация с Perception Loss

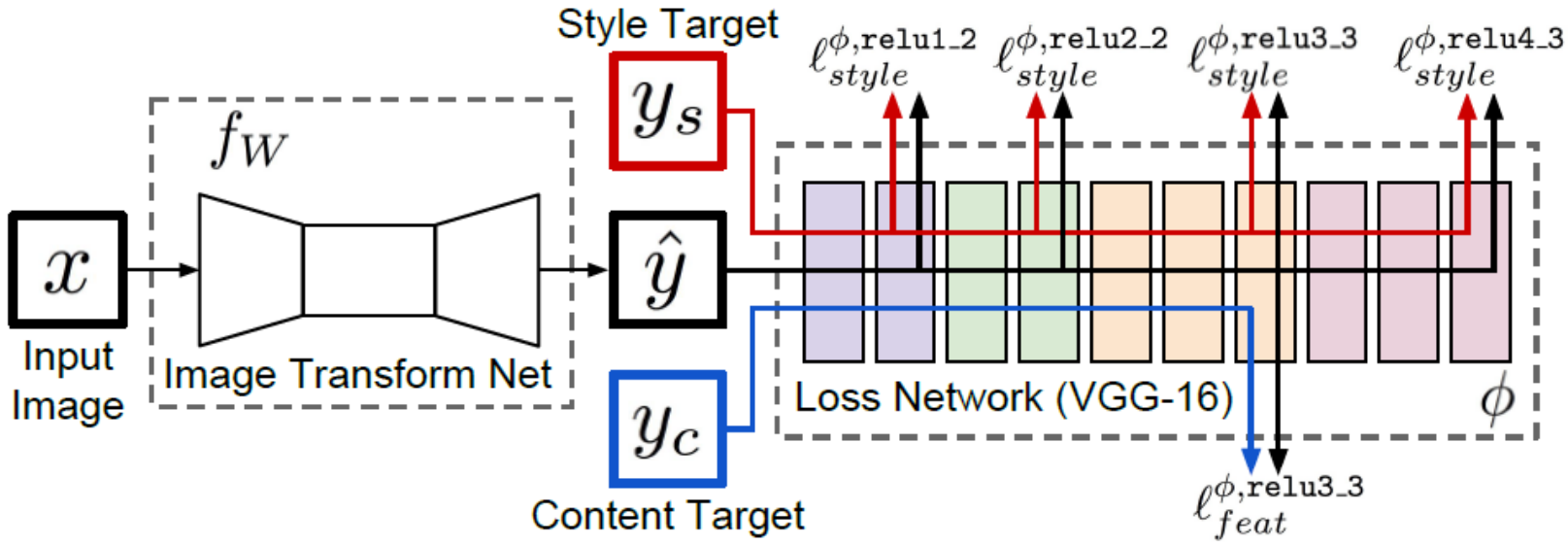
- Оптимизация с белого шума идёт медленно
- Идея: воспользоваться сетью-преобразователем
- Оценивать сходство входа и выхода будем через perception loss, стиля – через style loss
- Для каждого стиля нужно обучать свою собственную нейросеть





Johnson et. al. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. ECCV 2016

Обучение такой сети



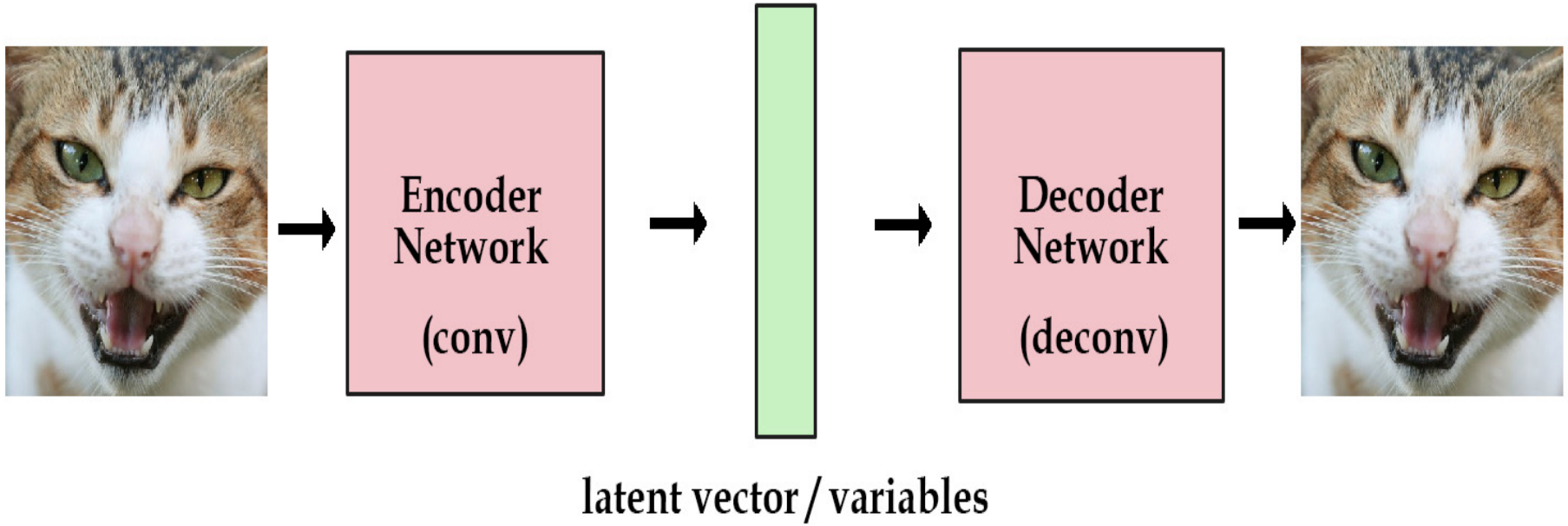
- Воспользуемся предобученной на задаче классификации на imagenet сетью VGG-16
- Будем использовать её для извлечения признаков, и она будет зафиксирована при обучении трансформационной сети
- Через неё прогоняем изображения y_s (стиля), и y_c=x (содержание)

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
- Обучим «автокодировщик» - сеть типа encoder-decoder, которая должна «сжимать» изображение в скрытое (latent) векторное представление F и декодировать его в исходное «без потерь»
- Будем манипулировать скрытым векторным представлением изображения F
- 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 $F_S F_S^T$
- Можно показать, что добиться цели можно через линейное преобразование вектор-признака

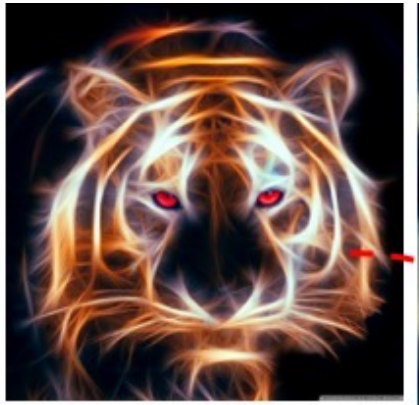
$FF^T = I$ $FF^T = F_S F_S^T$

- Делаем в 2 этапа:
  - сначала уберем корреляцию исходных признаков
  - затем приравняем нужные матрицы

Смешение стилей

Content

Mask



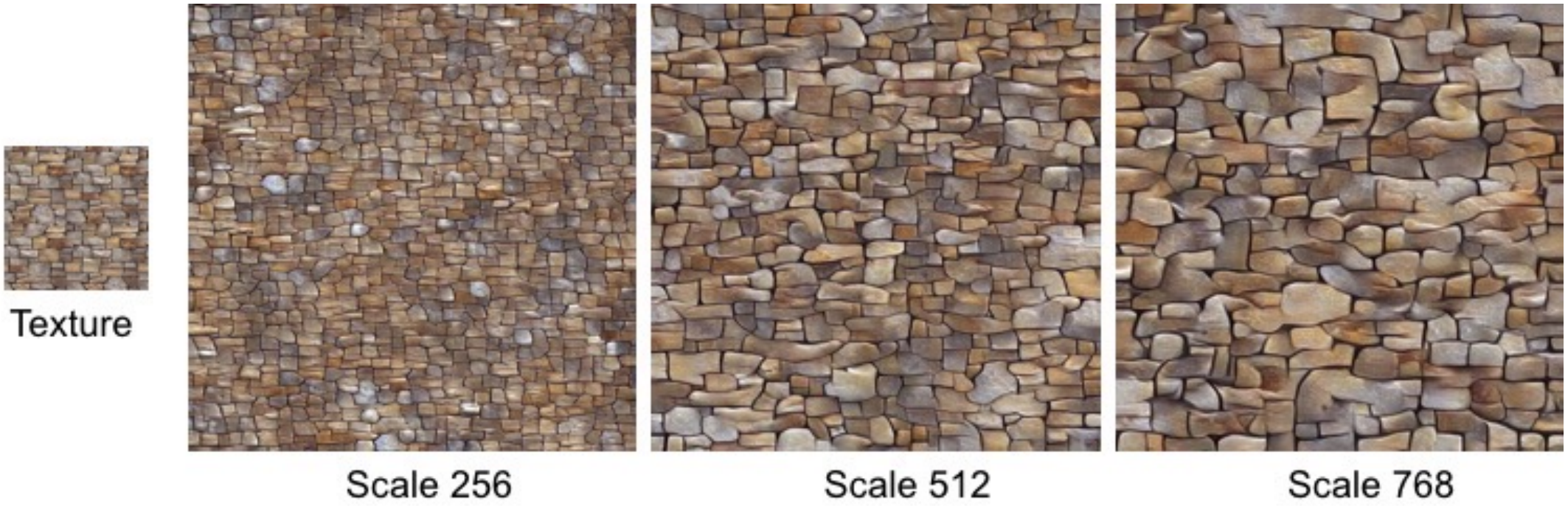
Style I

Style II

Style transfer result

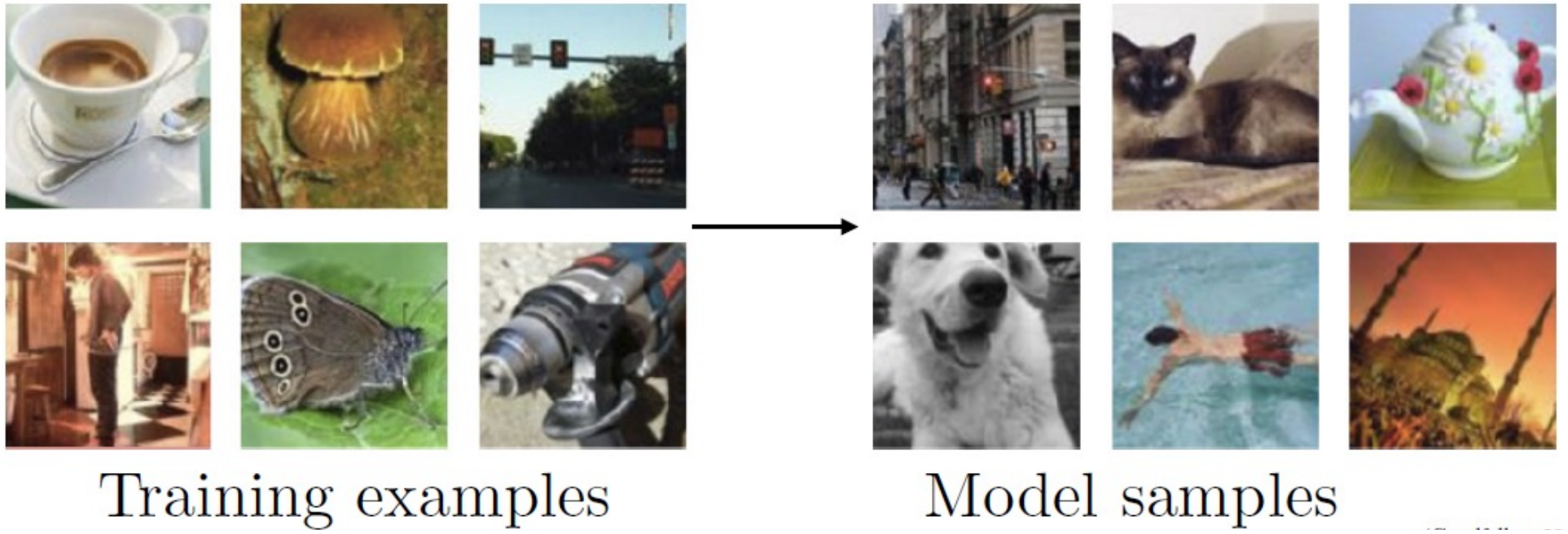
Генерация текстур

Замена изображения – содержания на шум



Синтез изображений и соперничающие сети

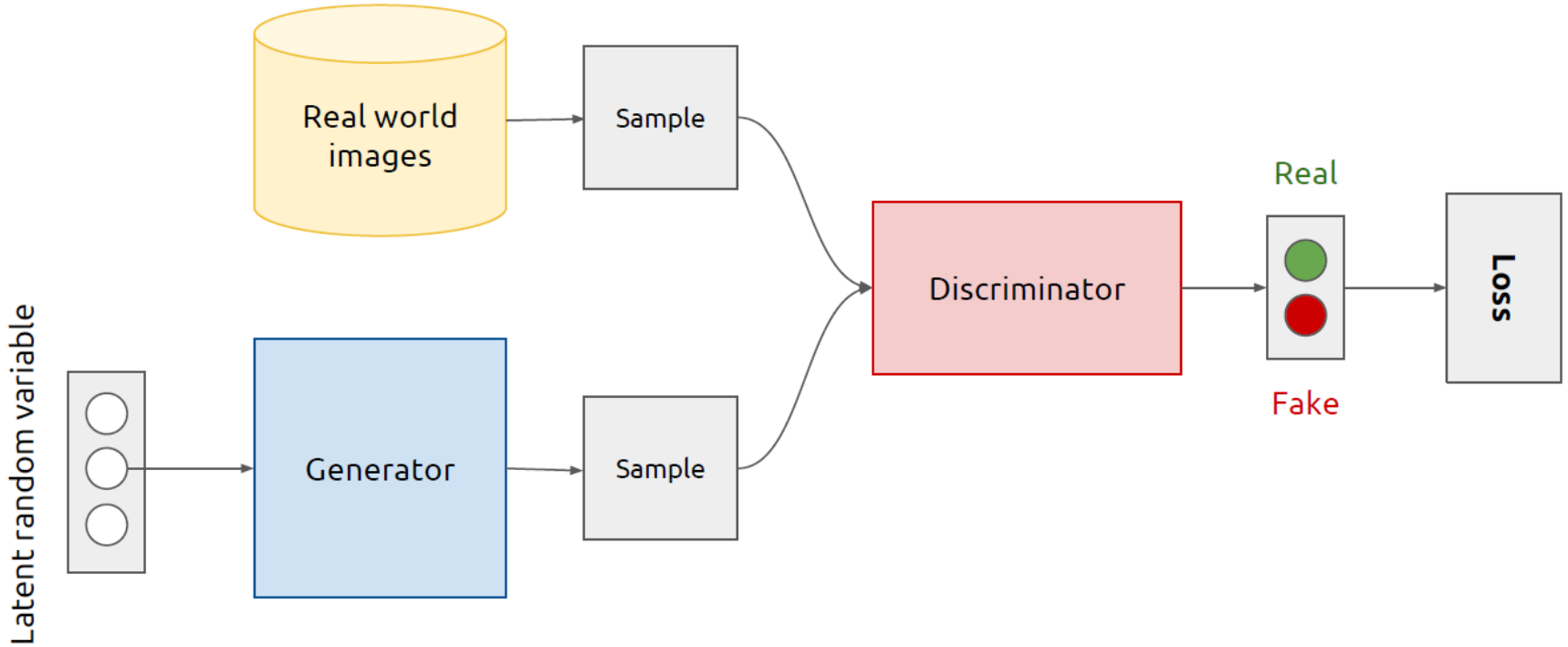
Хотим научиться генерировать изображения, похожие на обучающую выборку



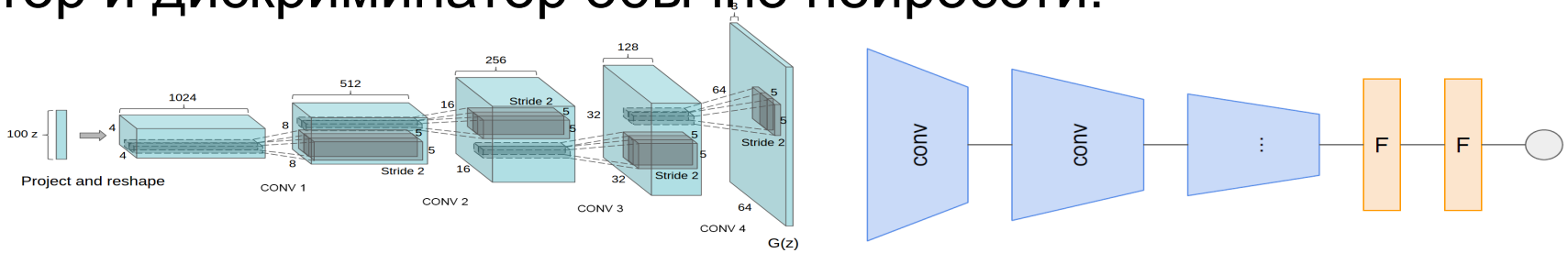
Как проверить, что у нас получилось хорошее, реалистичное изображение?

Общая схема Generative Adversarial Networks (GAN)

Сталкиваем генератор изображений из шума и дискриминатор:



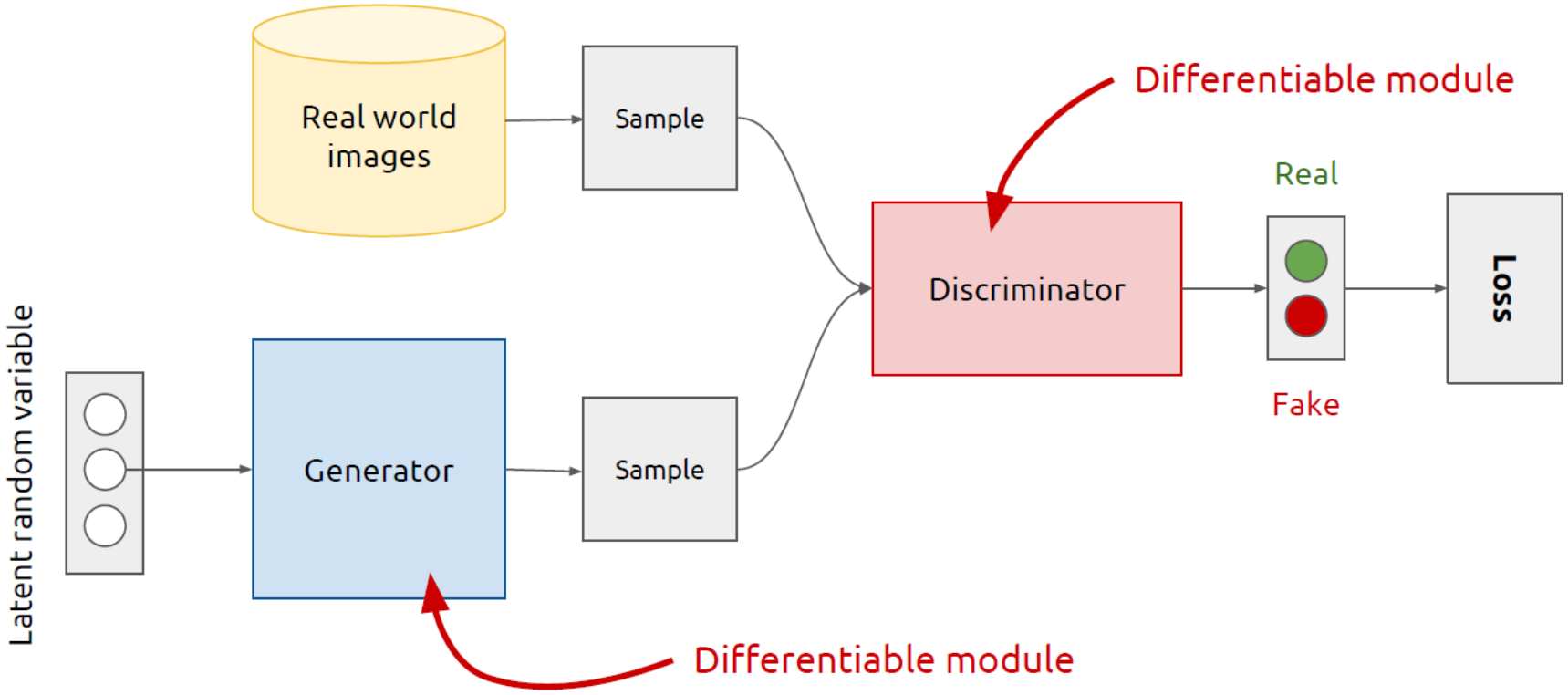
Генератор и дискриминатор обычно нейросети:



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

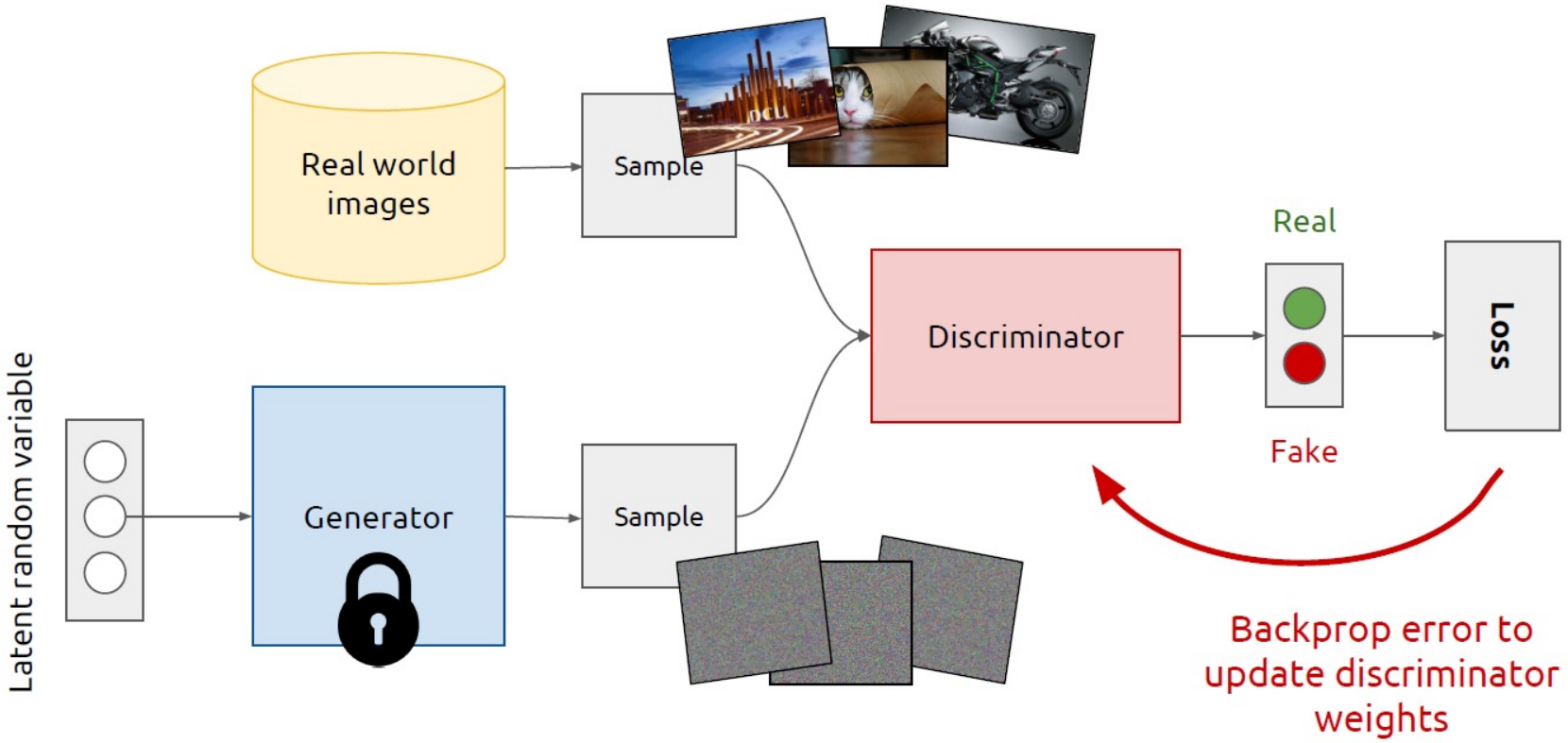
Обучение GAN

Будем обучать совместно, чередуя обучение дискриминатора и генератора



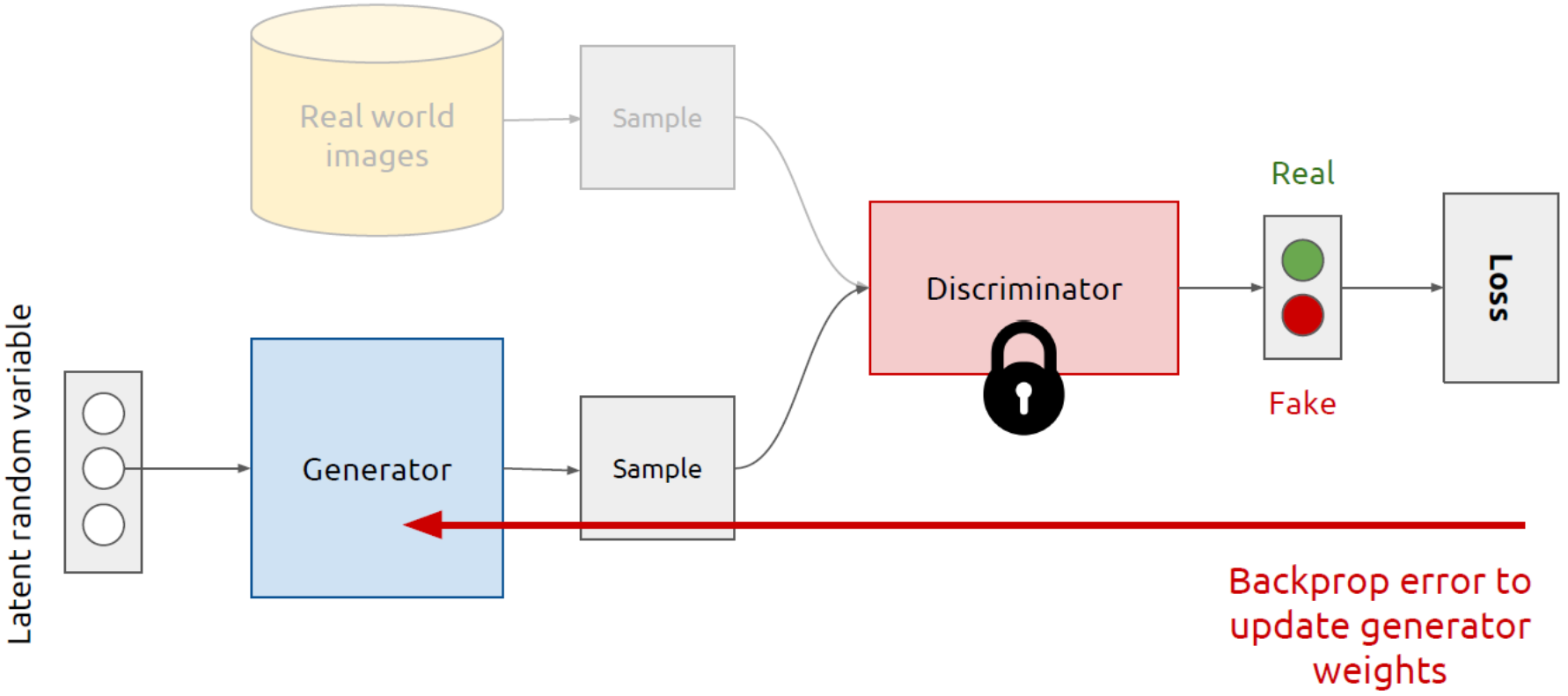
Почему нельзя взять готовый дискриминатор?

Обучение GAN



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

Обучение GAN

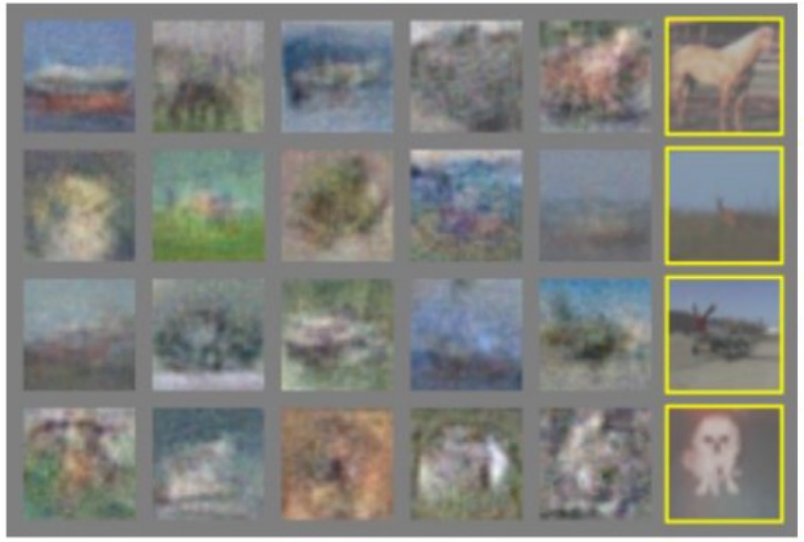


- Фиксируем веса дискриминатора
- Генерируем примеры генератором
- Распространяем ошибку от дискриминатора

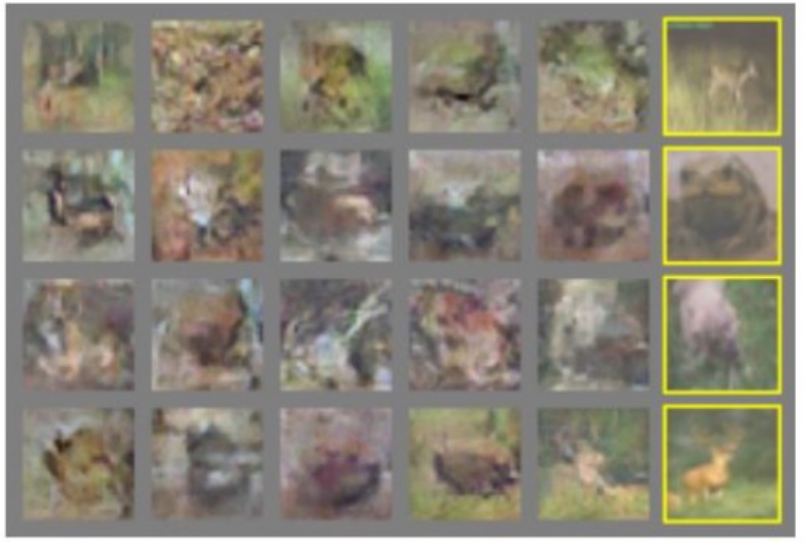
Результаты

a)

b)

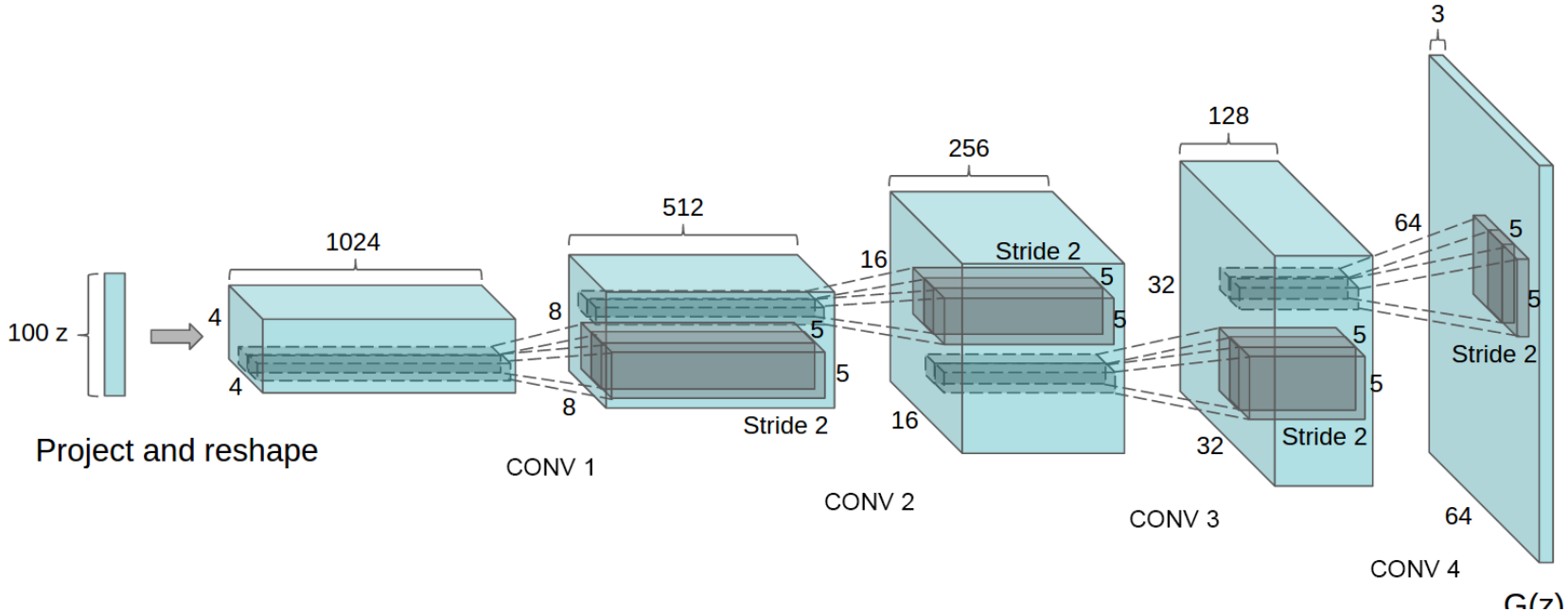


c)



d)

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.

A.Radford, L. Metz, S. Chintala UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS. ICLR 2016

Примеры - лица

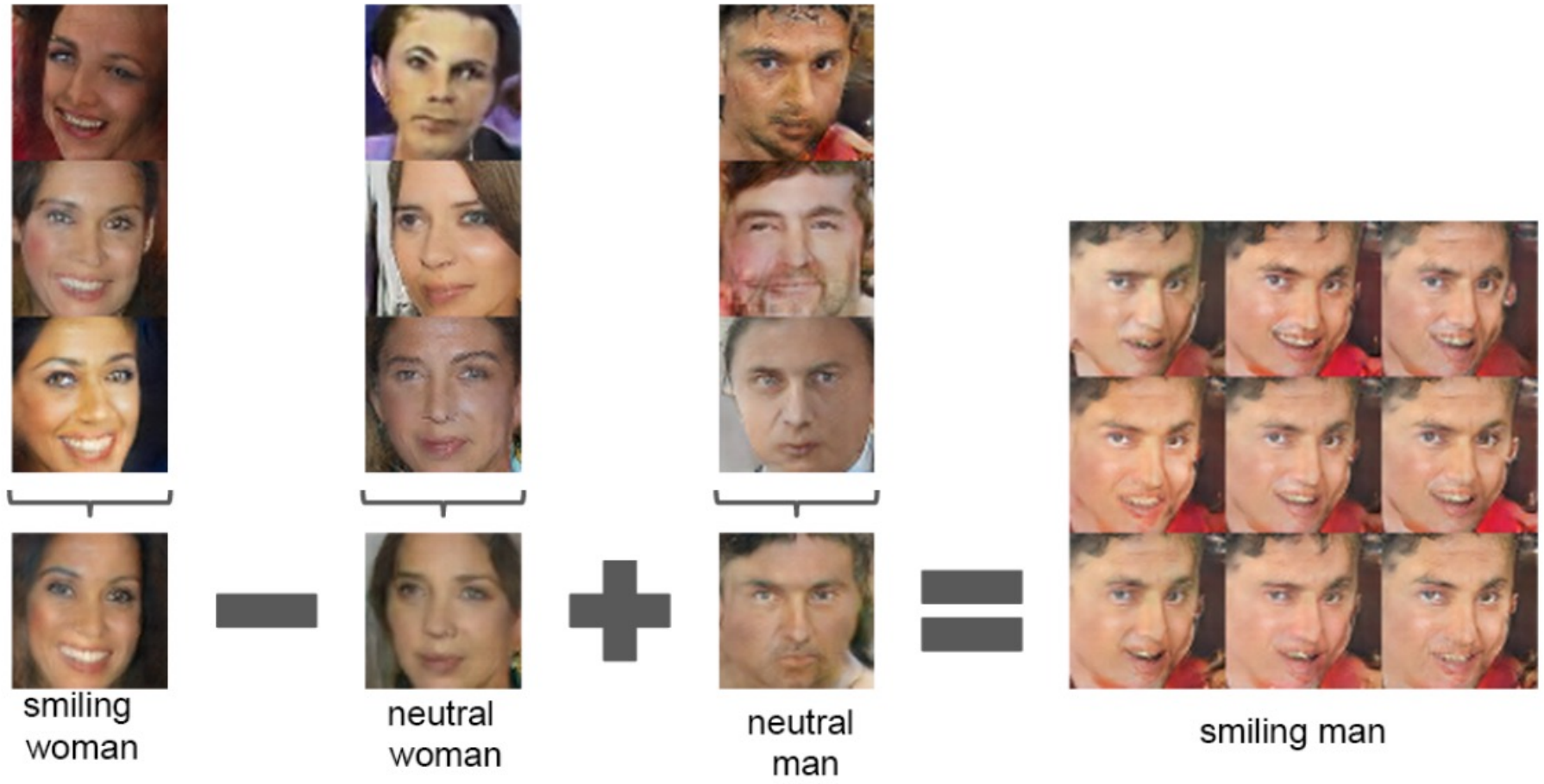
Примеры – обложки альбомов

Примеры - спальни

Изучение представления

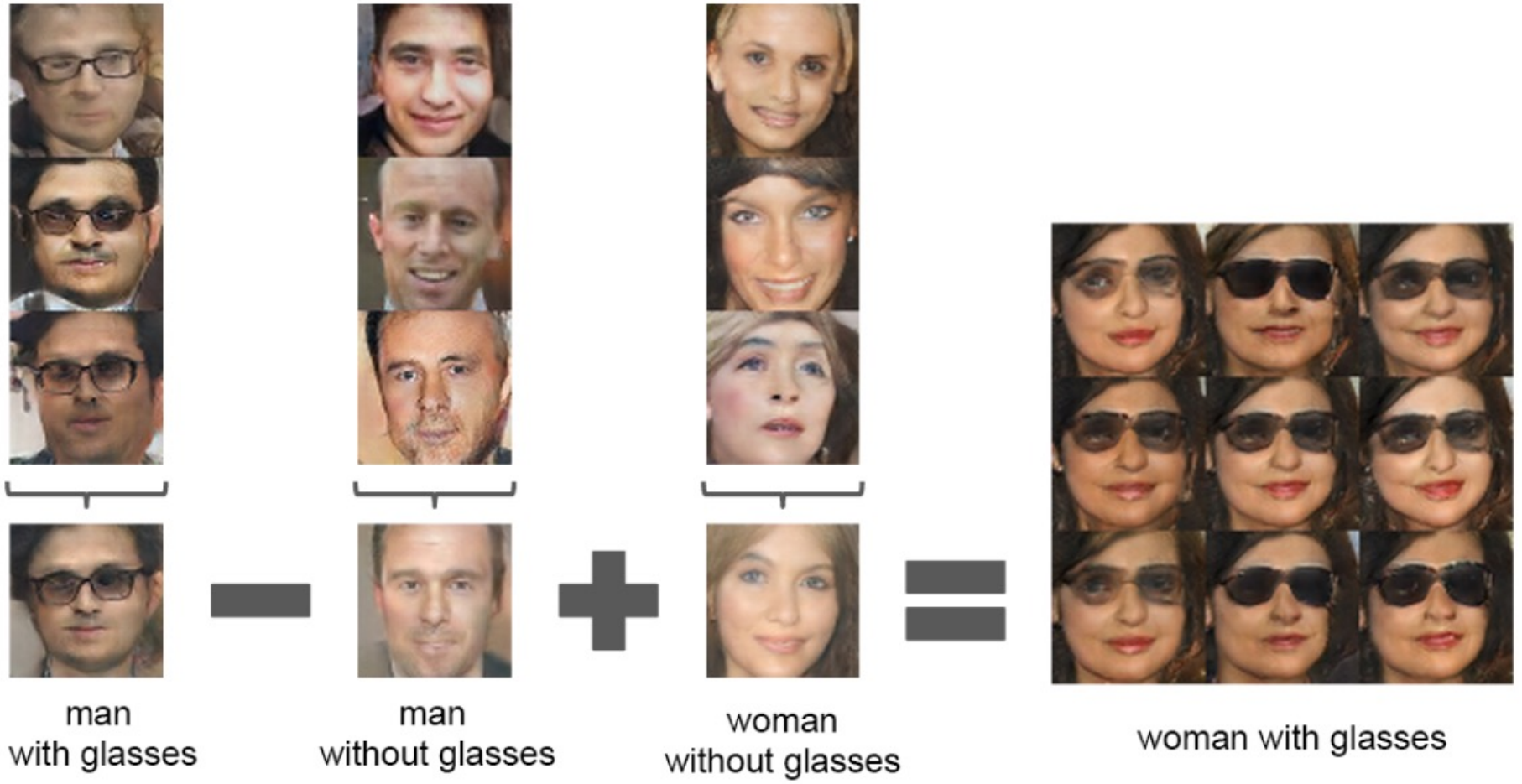
- Поиск фильтров, которые соответствуют определённым объектам
  - Логистическая регрессия по активациям фильтров внутри нарисованных областей
- Обнуление фильтров, которые были выбраны как соответствующие объектам
- На изображениях «исчезли» окна при некотором ухудшении качества
- Мораль – получаем представление,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

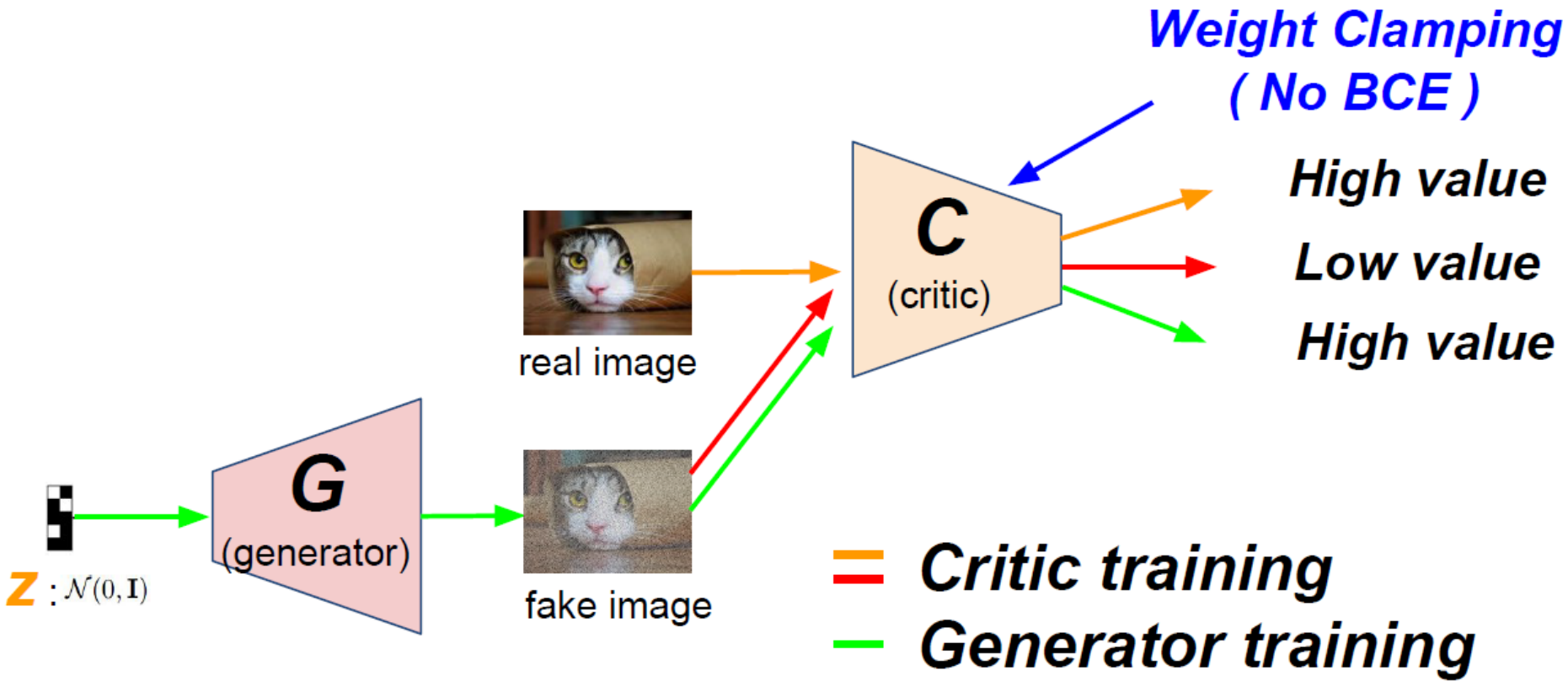


Векторная арифметика

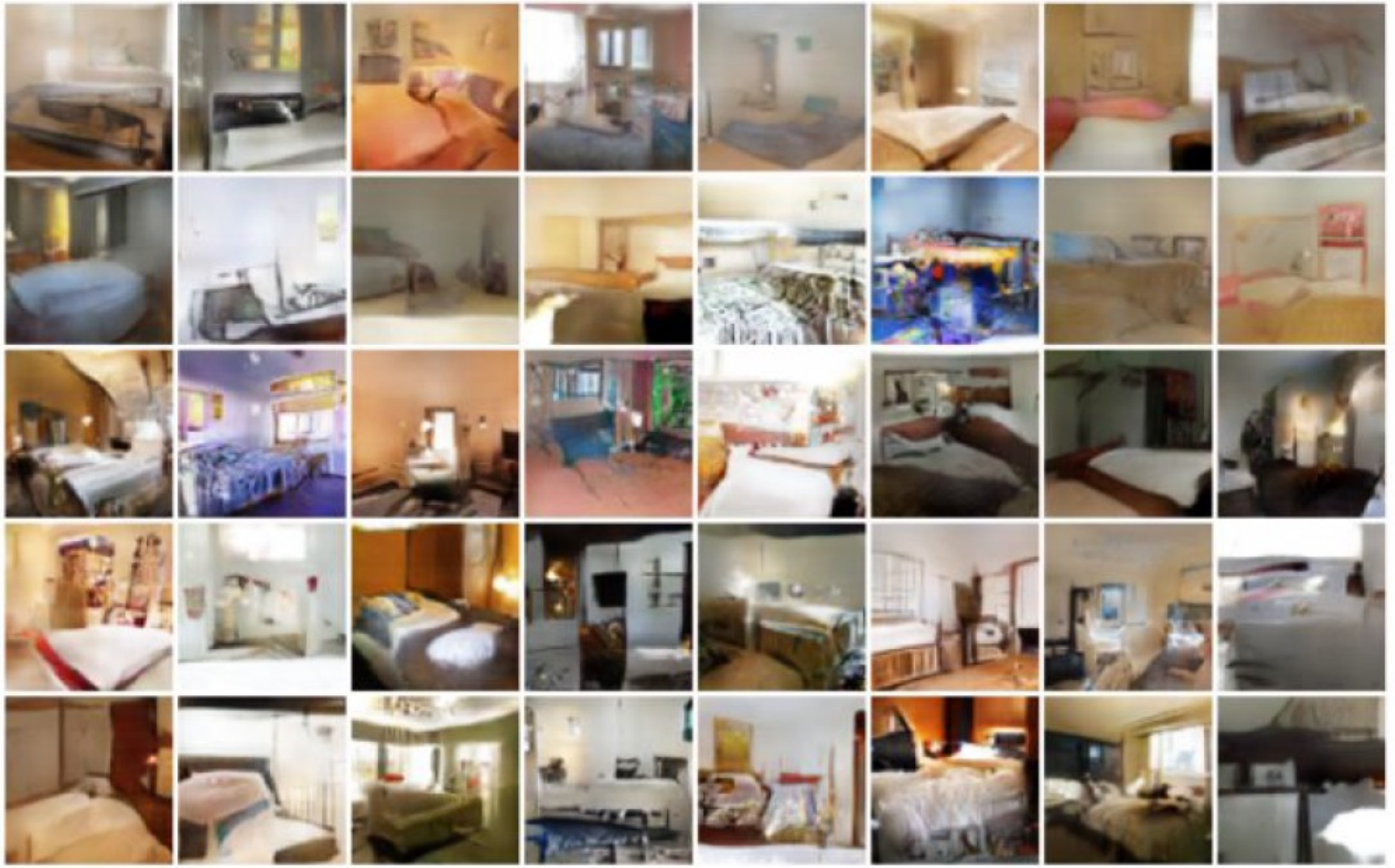
- Вектор «поворота» как разница между усреднёнными векторами 4х изображений с лицами повернутыми налево и 4мя направо

Wasserstein GAN

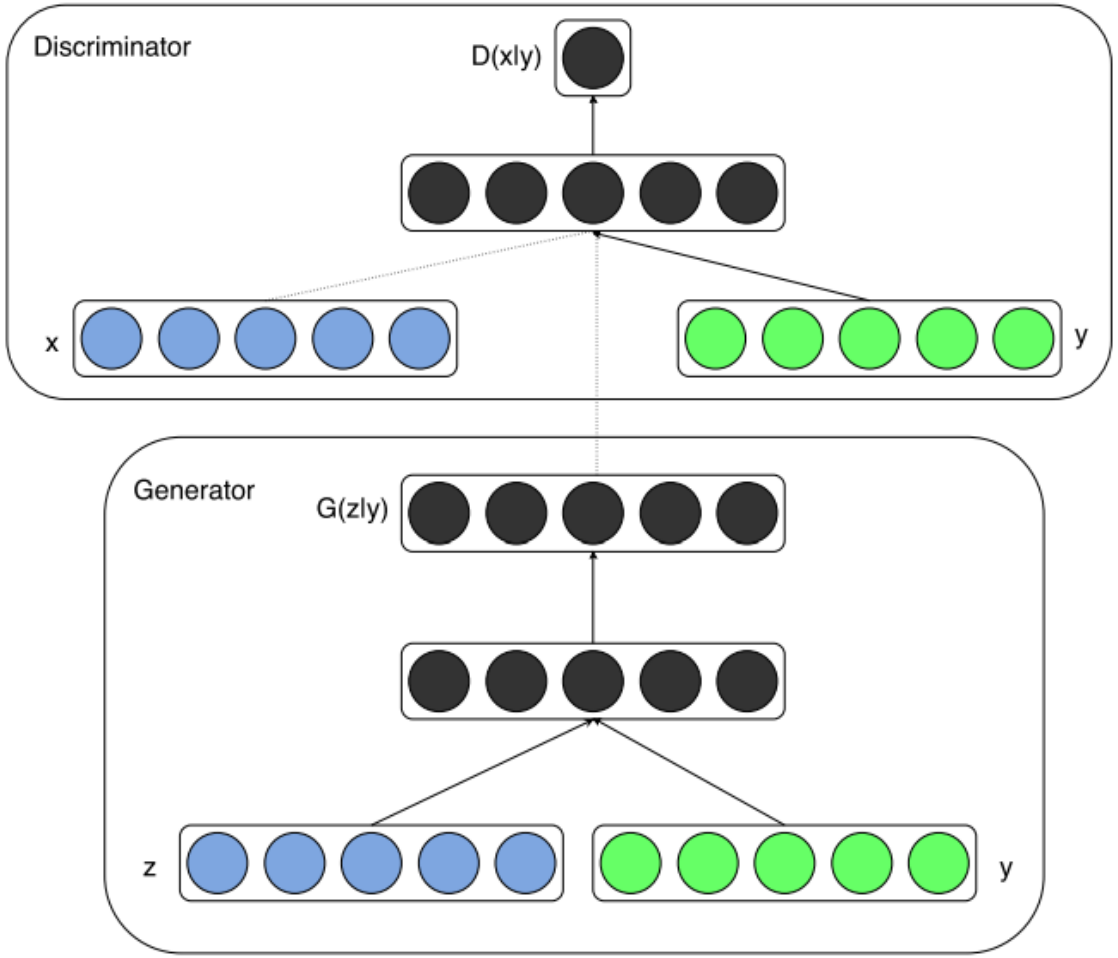
- Martin et al, Wasserstein GAN, 2017



Примеры работы

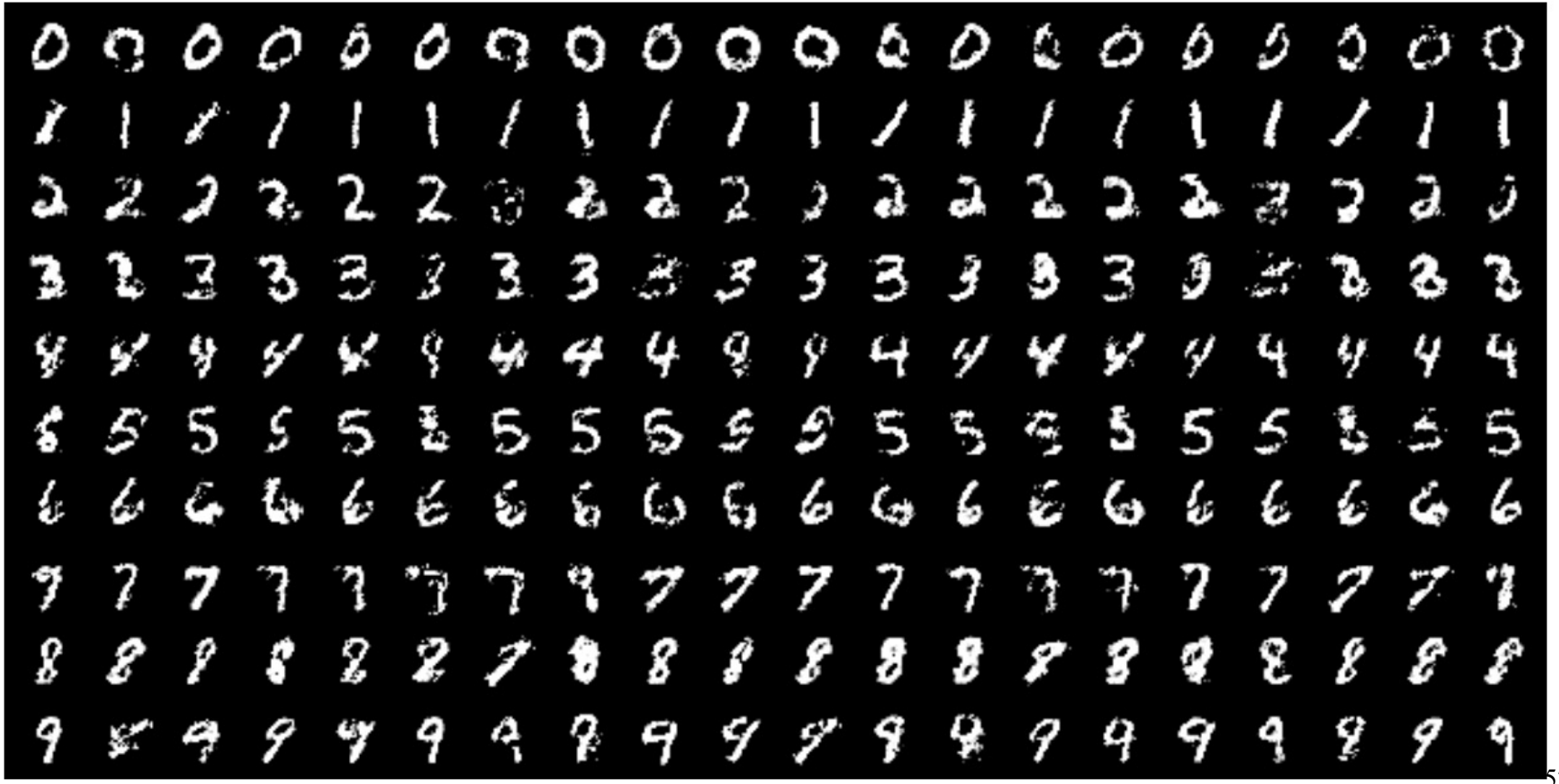


Conditional GAN

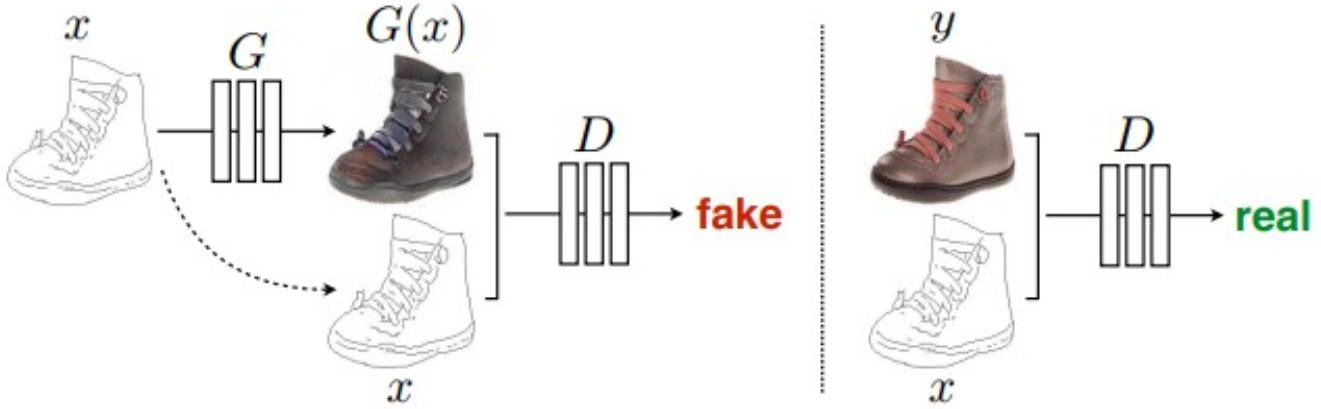


Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." *arXiv preprint arXiv:1411.1784* (2014).

Conditional GAN на MNIST



Pix2Pix для преобразования изображений



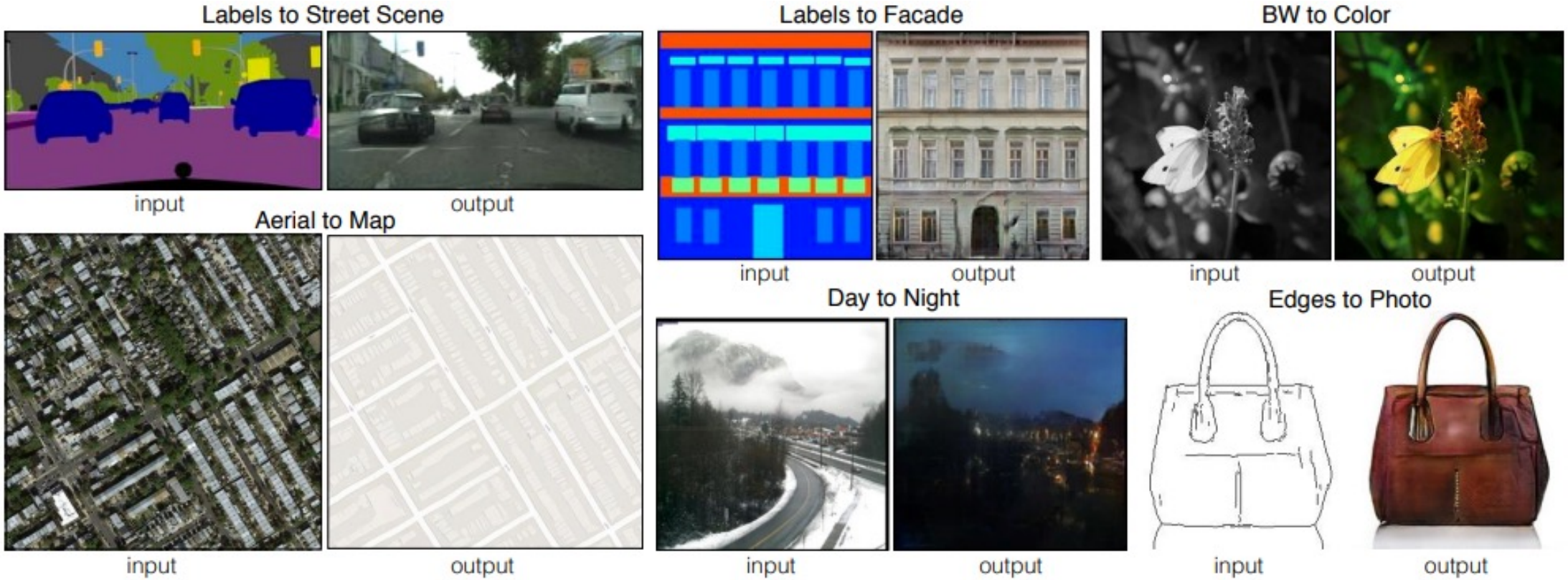
$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]$$

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

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

Pix2Pix

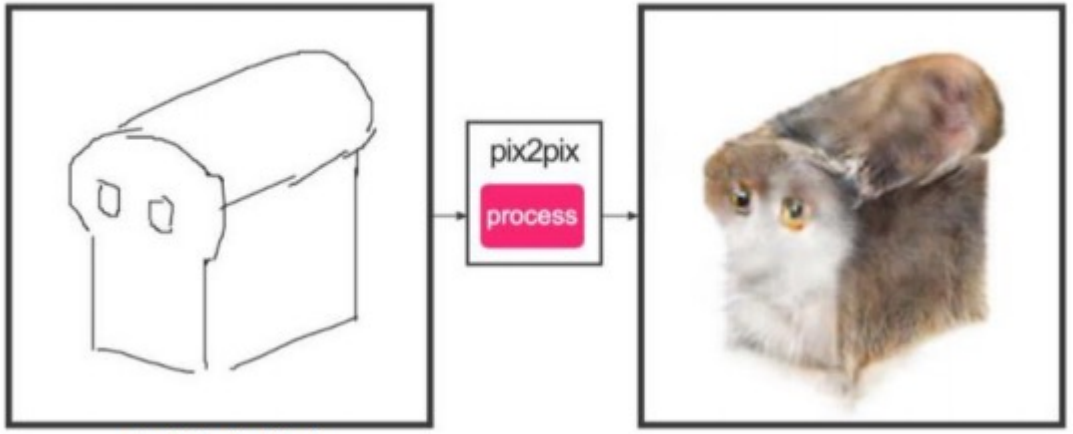

Labels to Street Scene
input
output
Labels to Facade
input
output
BW to Color
input
output
Aerial to Map
input
output
Day to Night
input
output
Edges to Photo
input
output

Pix2Pix

Pix2Pix

Pix2Pix

#edges2cats *by Christopher Hesse*



sketch by Ivy Tsai

Background removal

by Kaihu Chen

Palette generation

by Jack Qiao

Sketch → Pokemon

by Bertrand Gondouin

"Do as I do"

by Brannon Dorsey

Pix2Pix

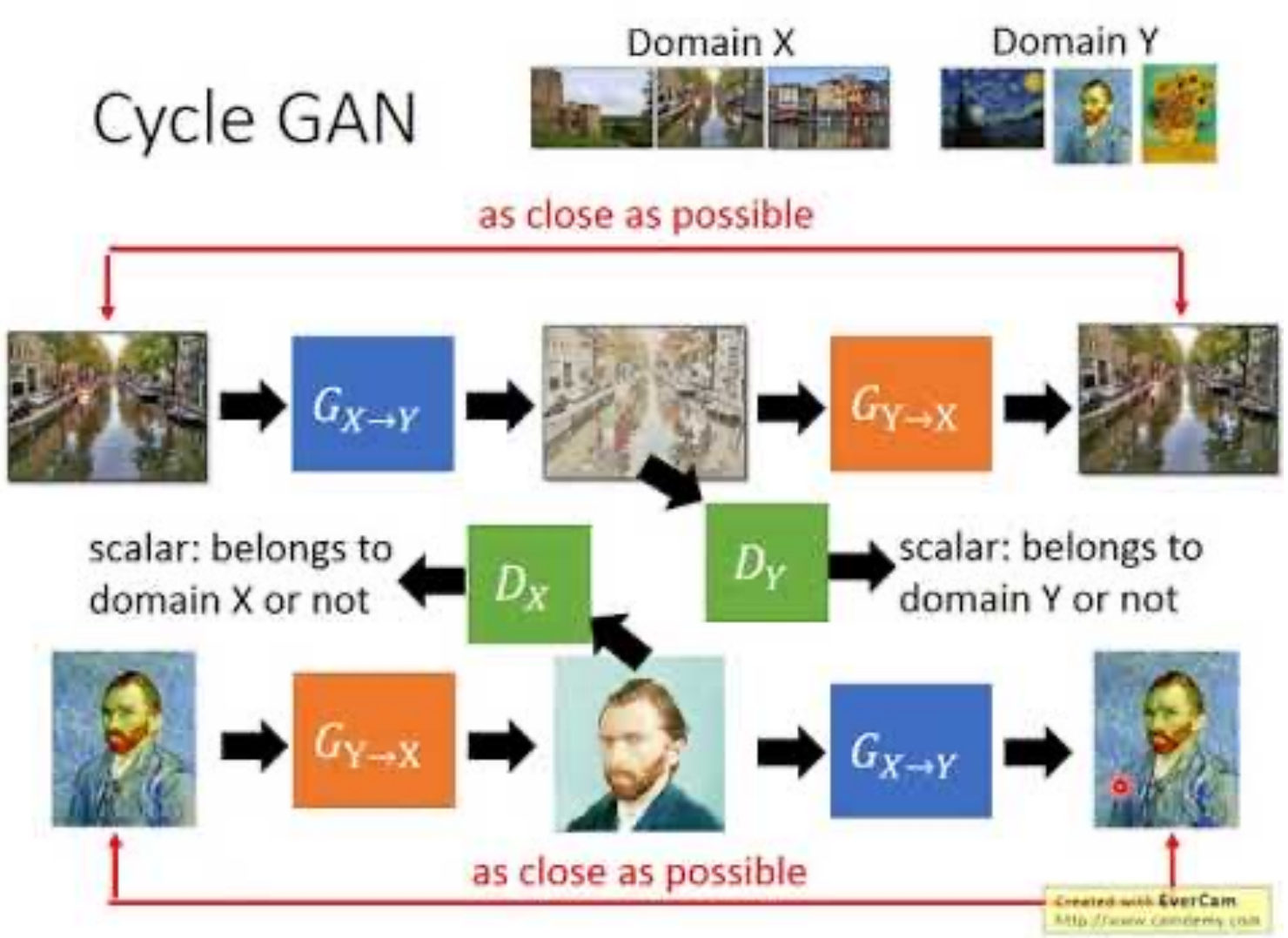
Архитектура:

- Генератор: U-net
- Дискриминатор: PatchGAN

Ограничение:

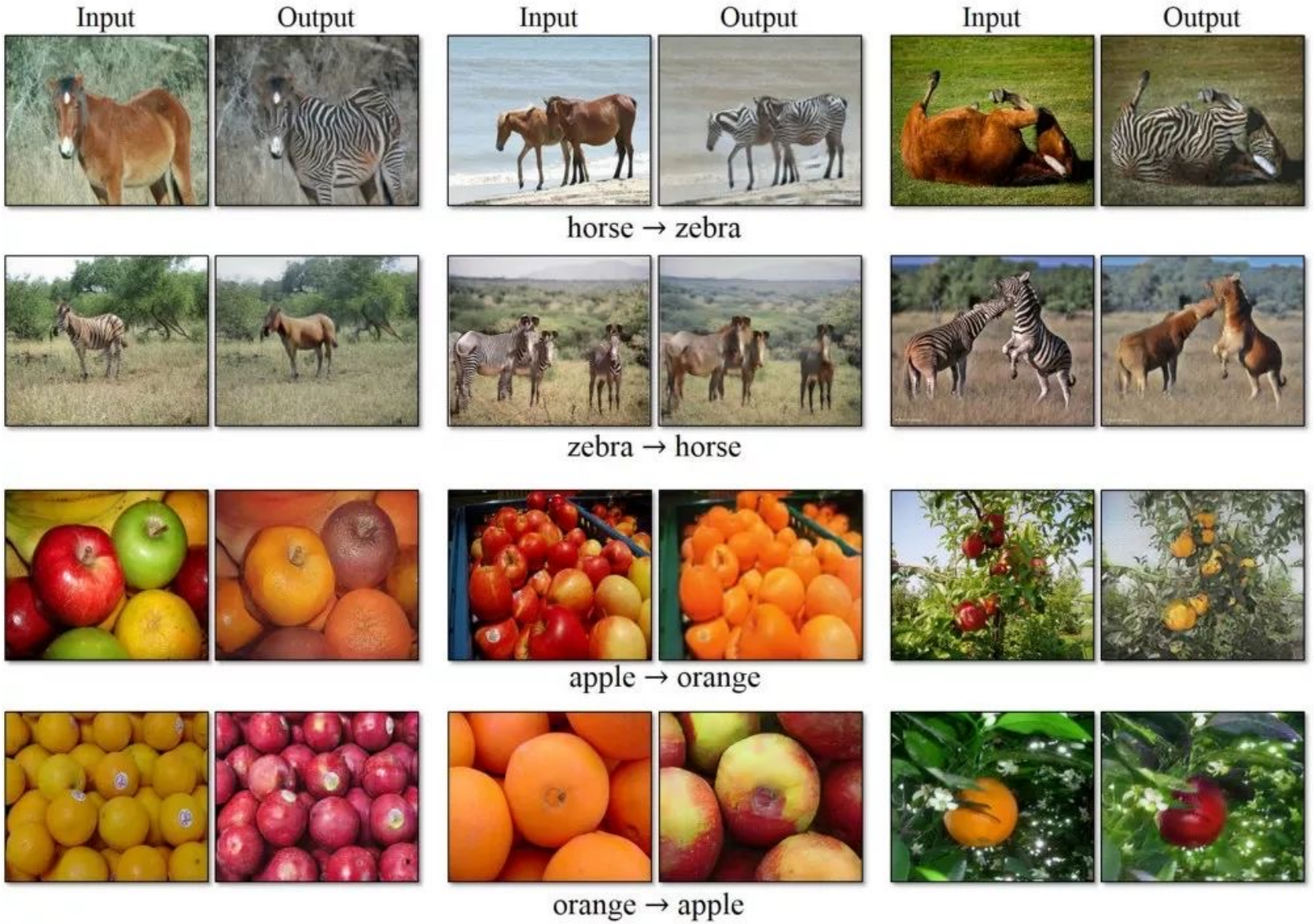
- Требует размеченные попарно данные

CycleGAN



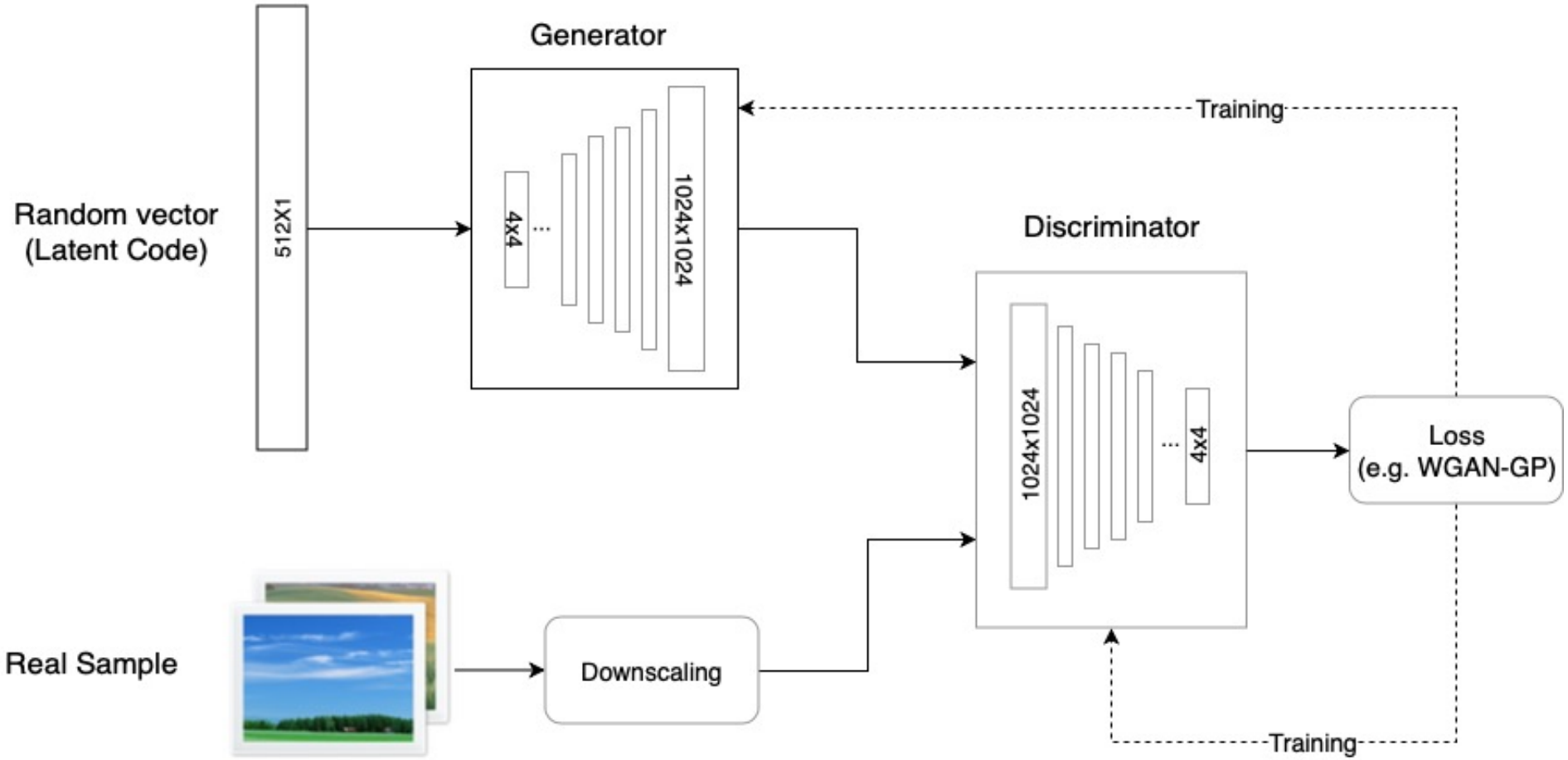
Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." *arXiv preprint*(2017).

CycleGAN


Input
Output
Input
Output
Input
Output
horse → zebra
zebra → horse
apple → orange
orange → apple

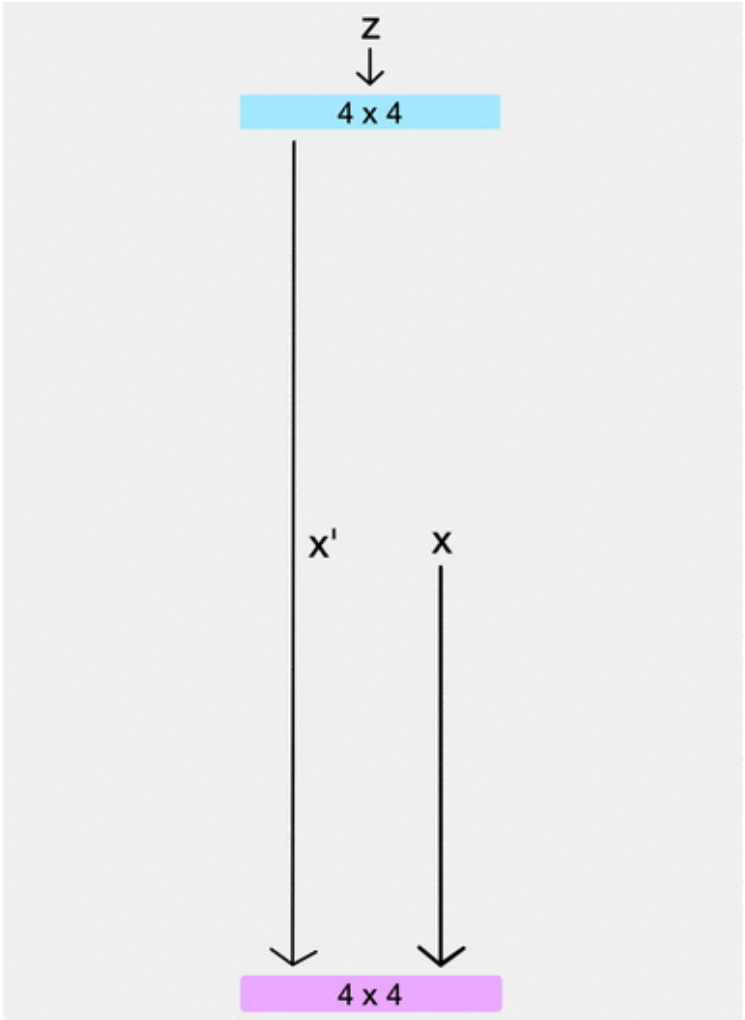
CycleGAN

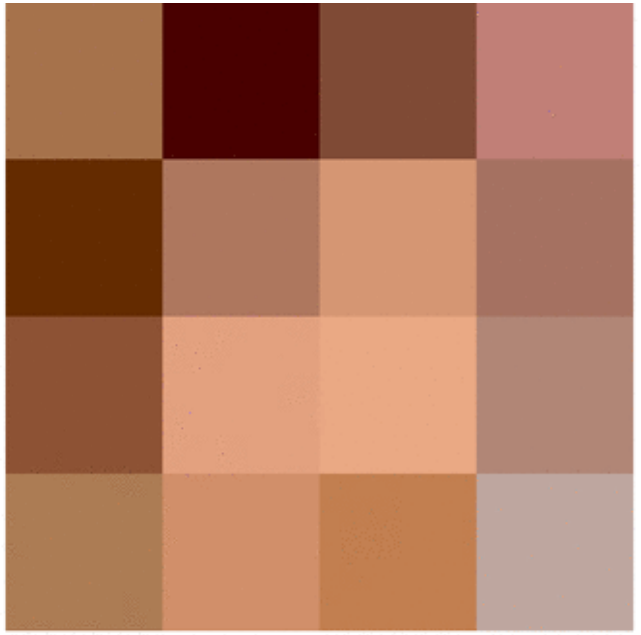
Progressive GAN



https://arxiv.org/abs/1710.10196

Демонстрация ProGAN



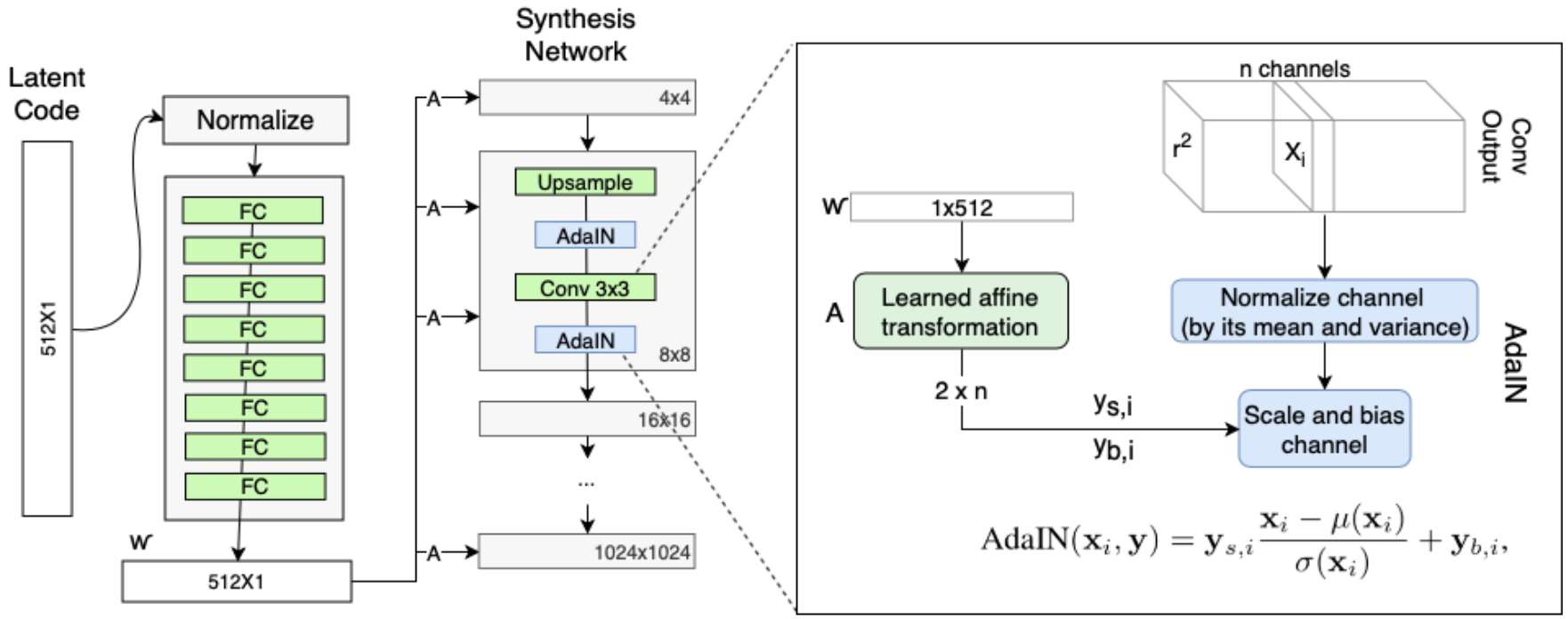


Training time: 0 days
4x4 resolution

Generator
Discriminator

z = random code
x = real image
x' = generated image

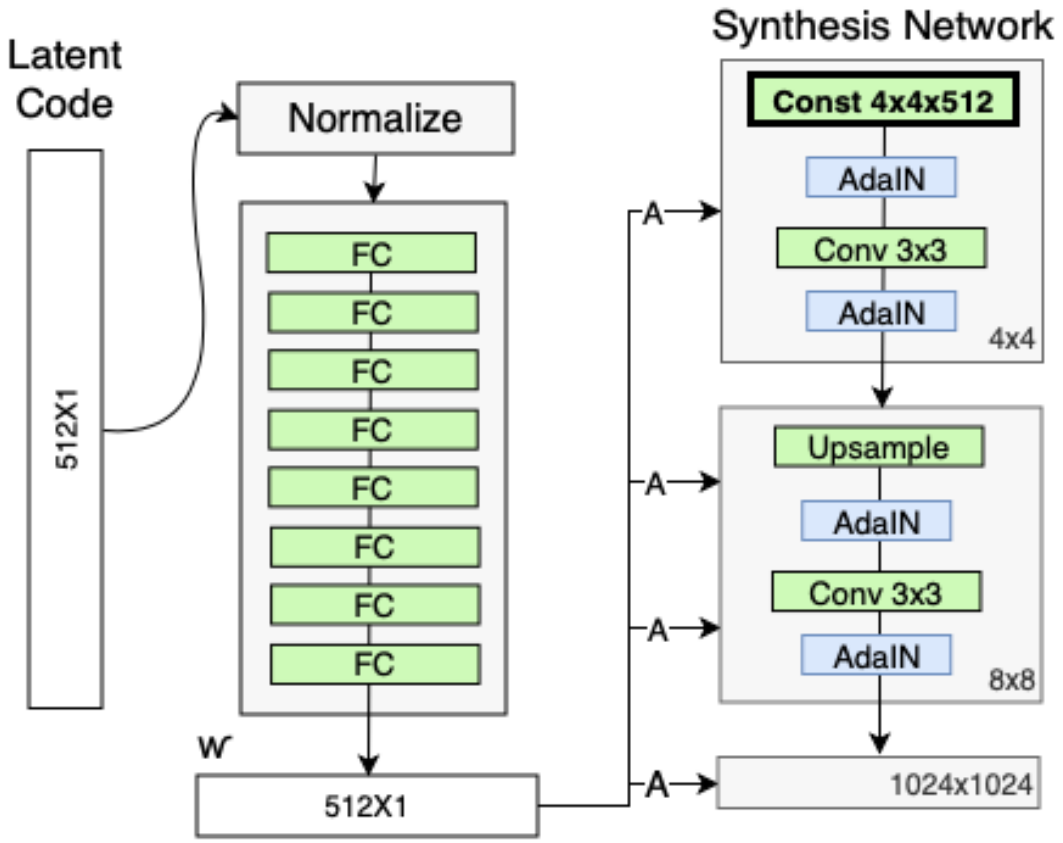
StyleGAN



Пропустим код через специальную сеть, которая сгенерирует «управляющий» сигнал. Сигнал подается на AdaIN слой

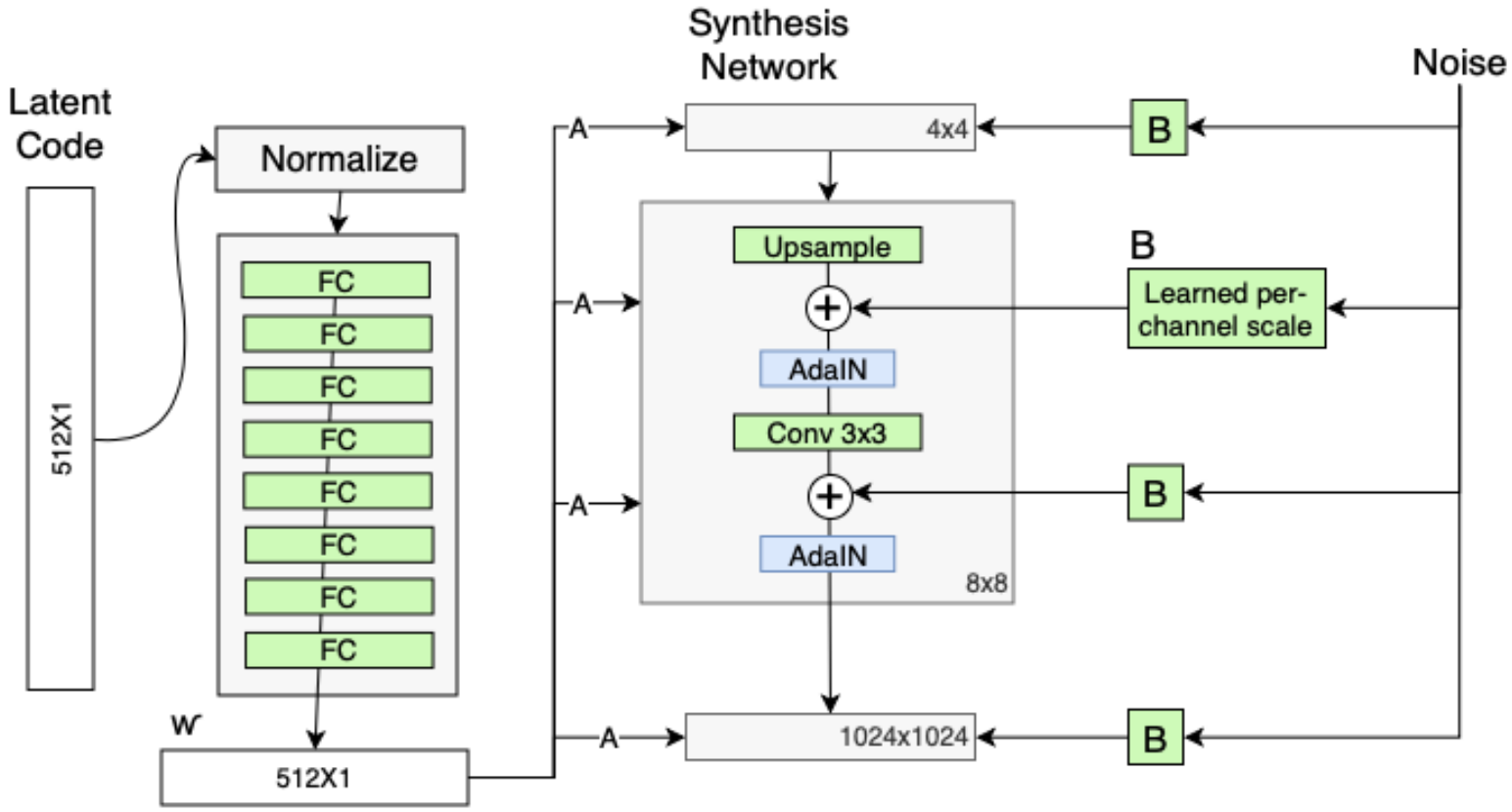
https://arxiv.org/pdf/1812.04948.pdf

StyleGAN



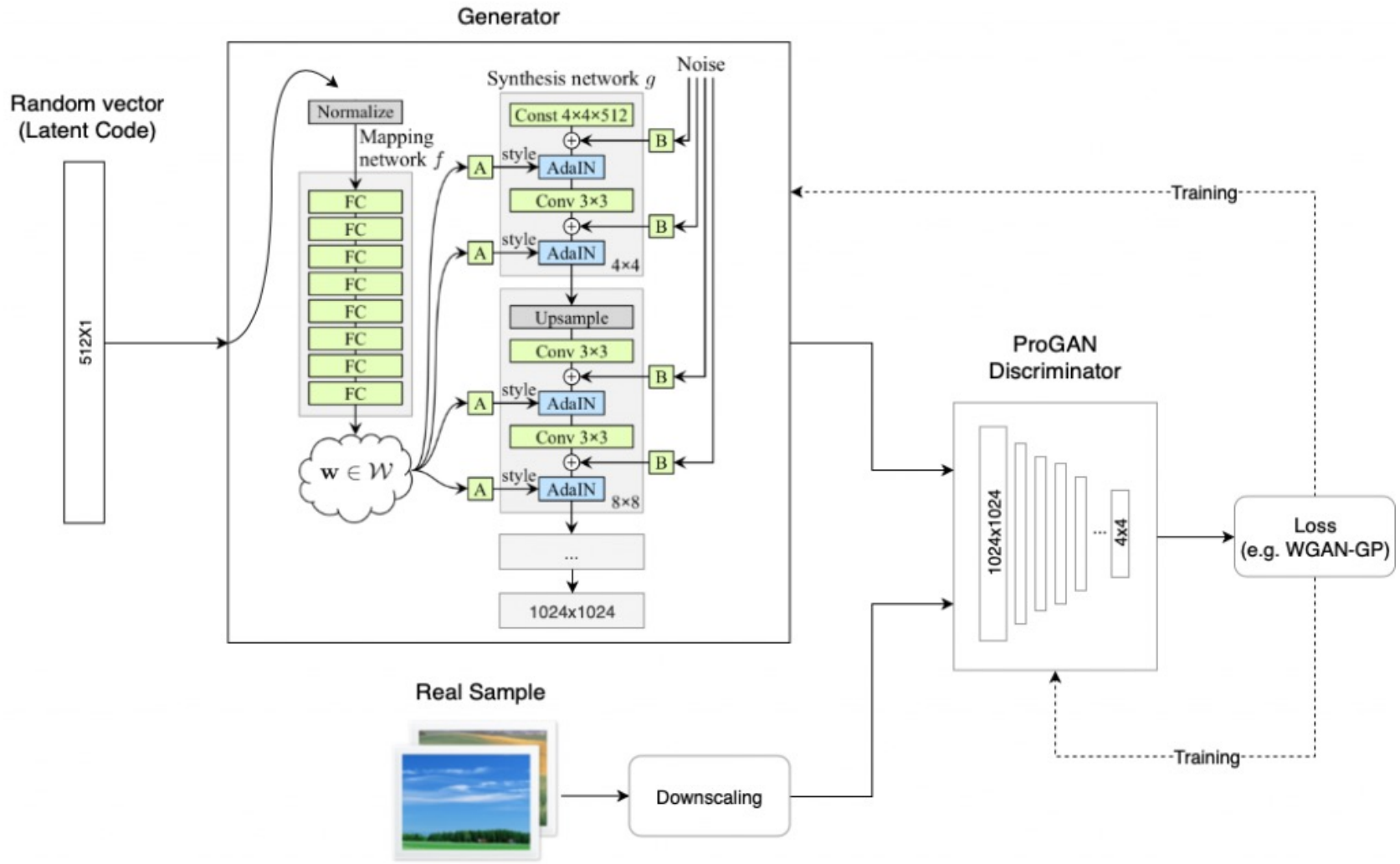
Оказывается, что на вход генератору можно подавать константное мини-изображение. И дальше оно будет преобразовываться через управляющие сигналы в целевое

StyleGAN



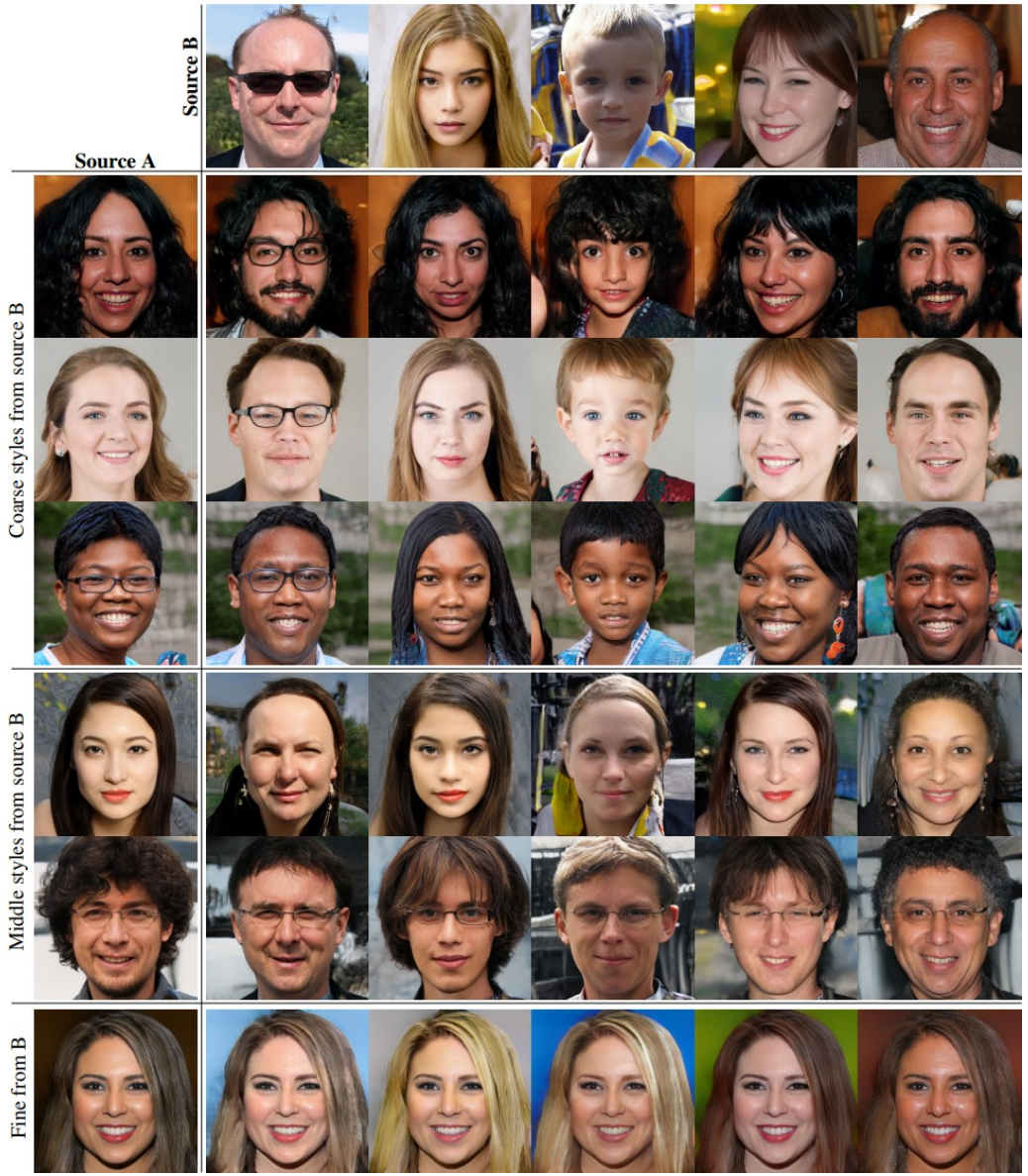
Для генерации случайных мелких деталей и дополнительной рандомизации будем подмешивать на каждом этапе шум

StyleGAN



Обучаем в прогрессивном режиме

StyleGAN



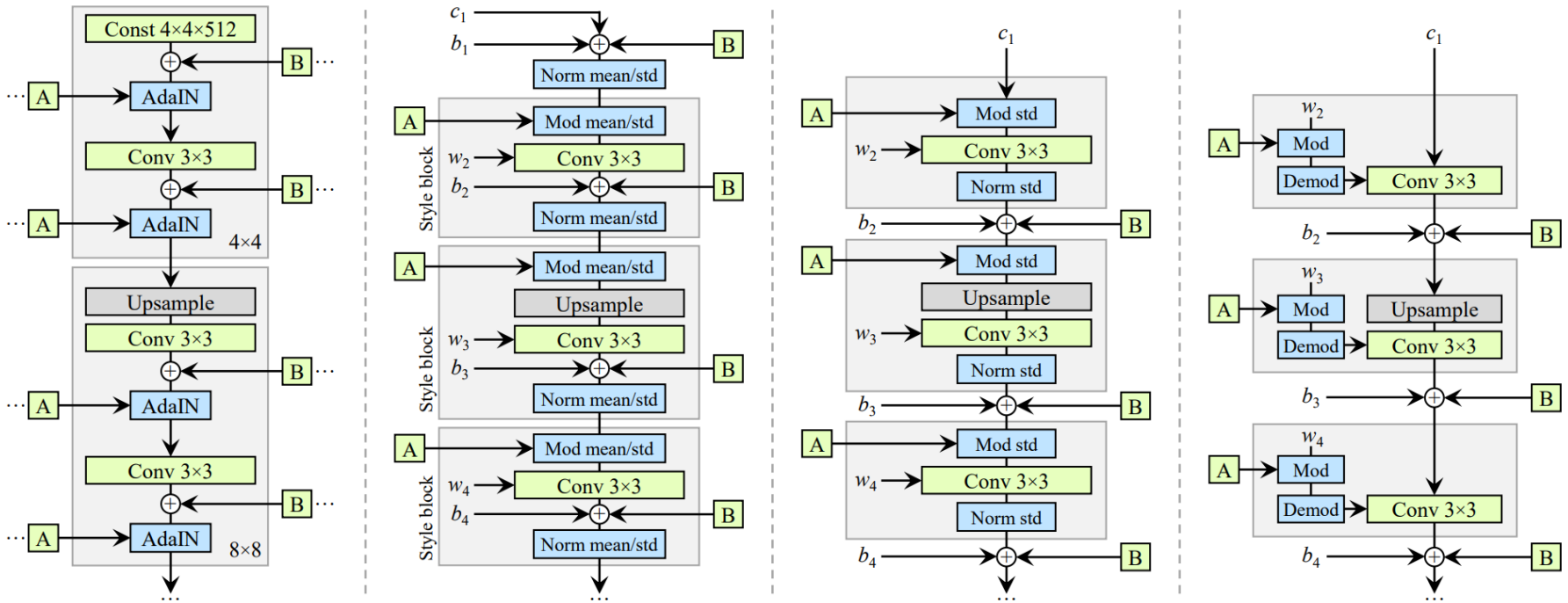
- Разделим «стиль» на компоненты
- Для этого будем учить сеть так, что на разные уровни подаётся управляющий сигнал, сгенерированный от разных кодов
- Можем управлять генерацией, выбирая интересные нам изображения и подмешивая их коды
- В примерах – стиль от B на соответствующем уровне, и всё остальное от A

Артефакты StyleGAN

Figure 1. Instance normalization causes water droplet -like artifacts in StyleGAN images. These are not always obvious in the generated images, but if we look at the activations inside the generator network, the problem is always there, in all feature maps starting from the 64x64 resolution. It is a systemic problem that plagues all StyleGAN images.

https://arxiv.org/pdf/1912.04958.pdf

StyleGAN v2



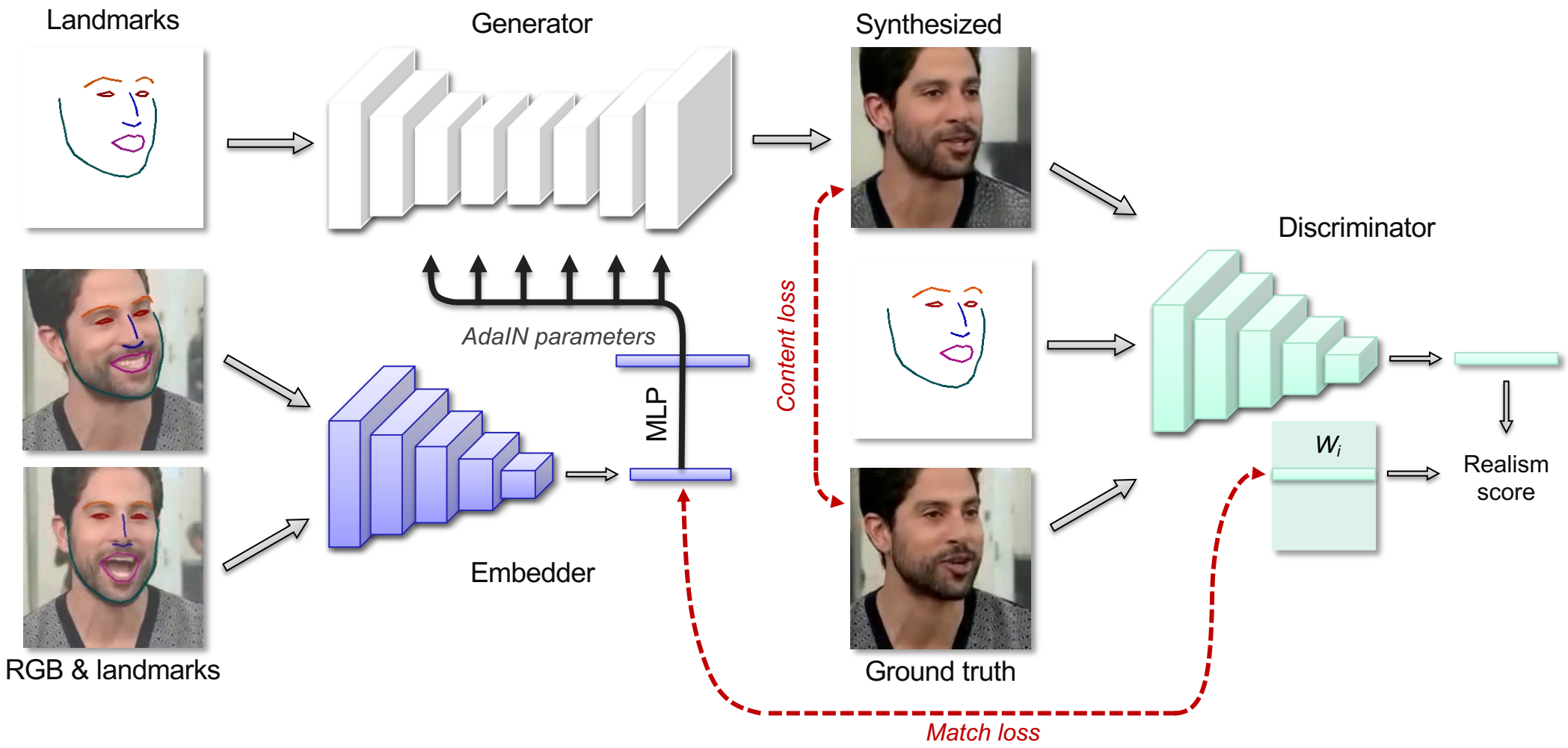
(a) StyleGAN (b) StyleGAN (detailed) (c) Revised architecture (d) Weight demodulation

Figure 2. We redesign the architecture of the StyleGAN synthesis network. (a) The original StyleGAN, where A denotes a learned affine transform from $\mathcal{W}$ that produces a style and B is a noise broadcast operation. (b) The same diagram with full detail. Here we have broken the AdaIN to explicit normalization followed by modulation, both operating on the mean and standard deviation per feature map. We have also annotated the learned weights (w), biases (b), and constant input (c), and redrawn the gray boxes so that one style is active per box. The activation function (leaky ReLU) is always applied right after adding the bias. (c) We make several changes to the original architecture that are justified in the main text. We remove some redundant operations at the beginning, move the addition of b and B to be outside active area of a style, and adjust only the standard deviation per feature map. (d) The revised architecture enables us to replace instance normalization with a "demodulation" operation, which we apply to the weights associated with each convolution layer.

https://arxiv.org/pdf/1912.04958.pdf

StyleGAN v2

AdaIN human avatars

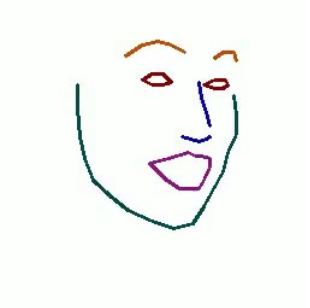


[Zakharov et al. ICCV19]

AdaIN human avatars

Training data:

…

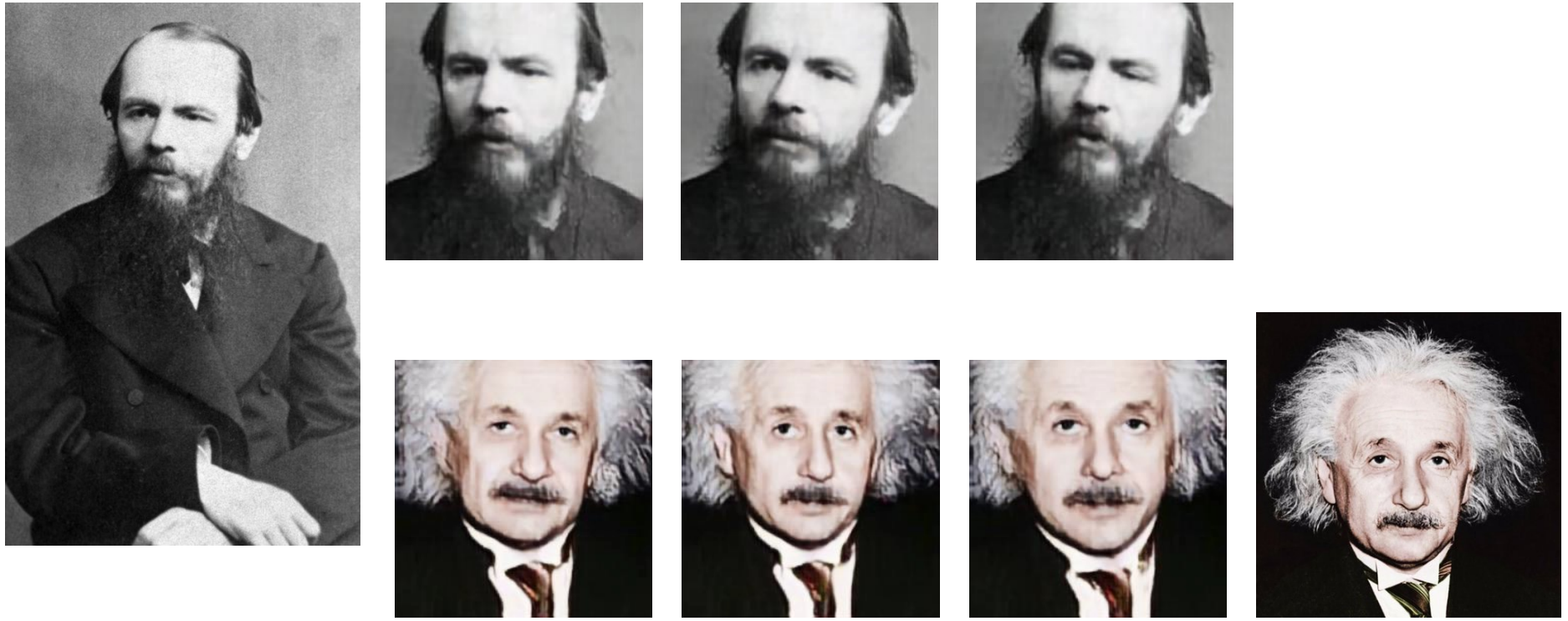


Landmarks

One-shot result Eight-shot result 32-shot result

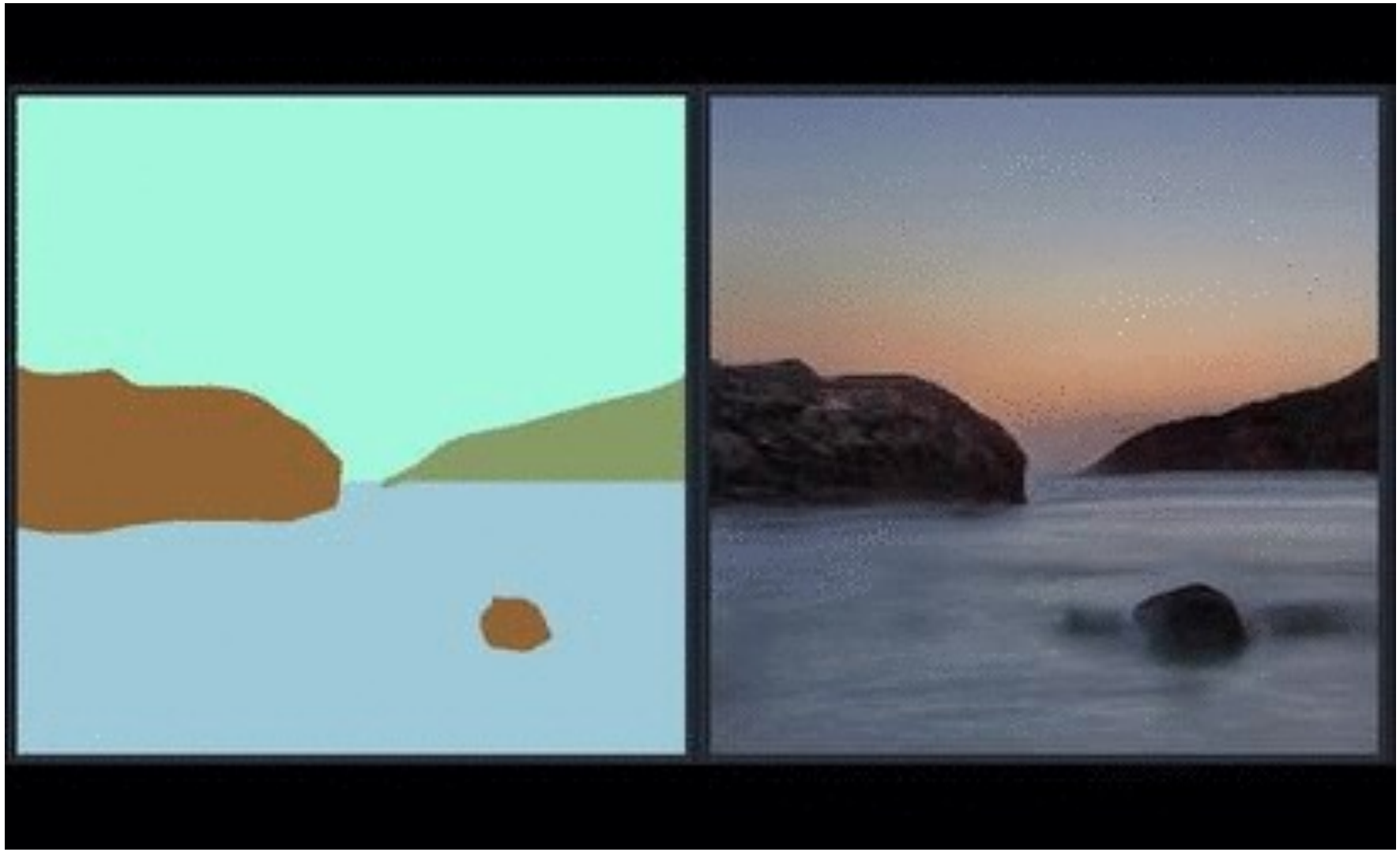
[Zakharov et al. ICCV19]

AdaIn human avatars



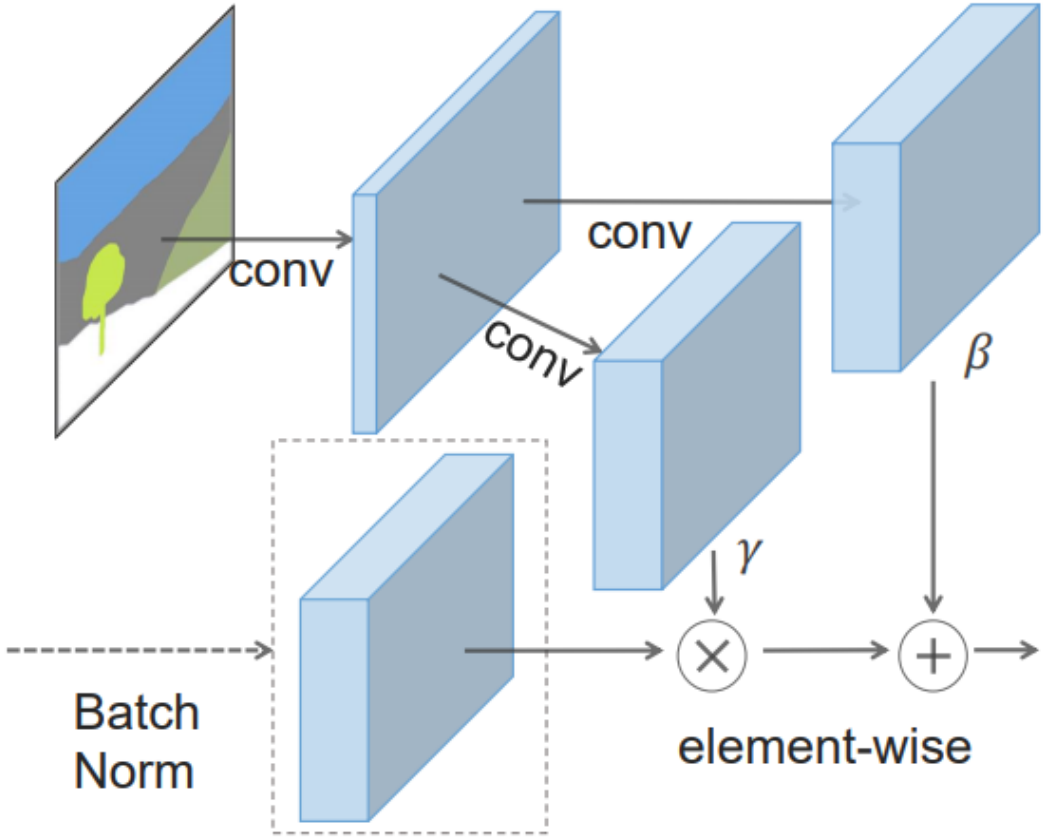
[Zakharov et al. ICCV19]

SPADE (or GauGAN)



https://arxiv.org/pdf/1903.07291.pdf

Spatially-adaptive denormalization



- Похоже на AdaIN, но генерируем параметры для каждого пикселя
- Параметры генерируются управляющей сеткой

Генератор в SPADE

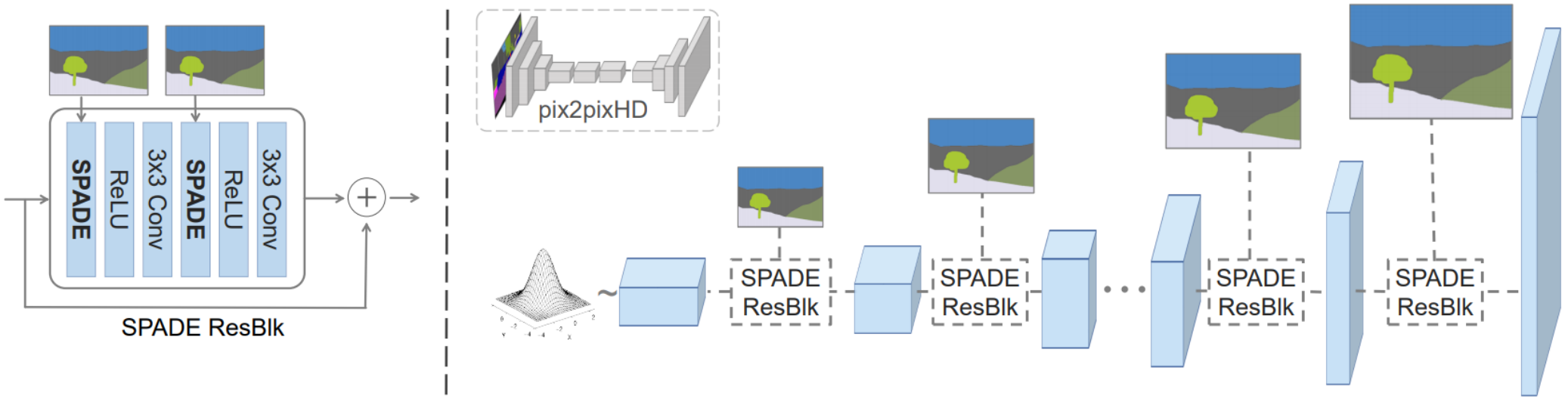
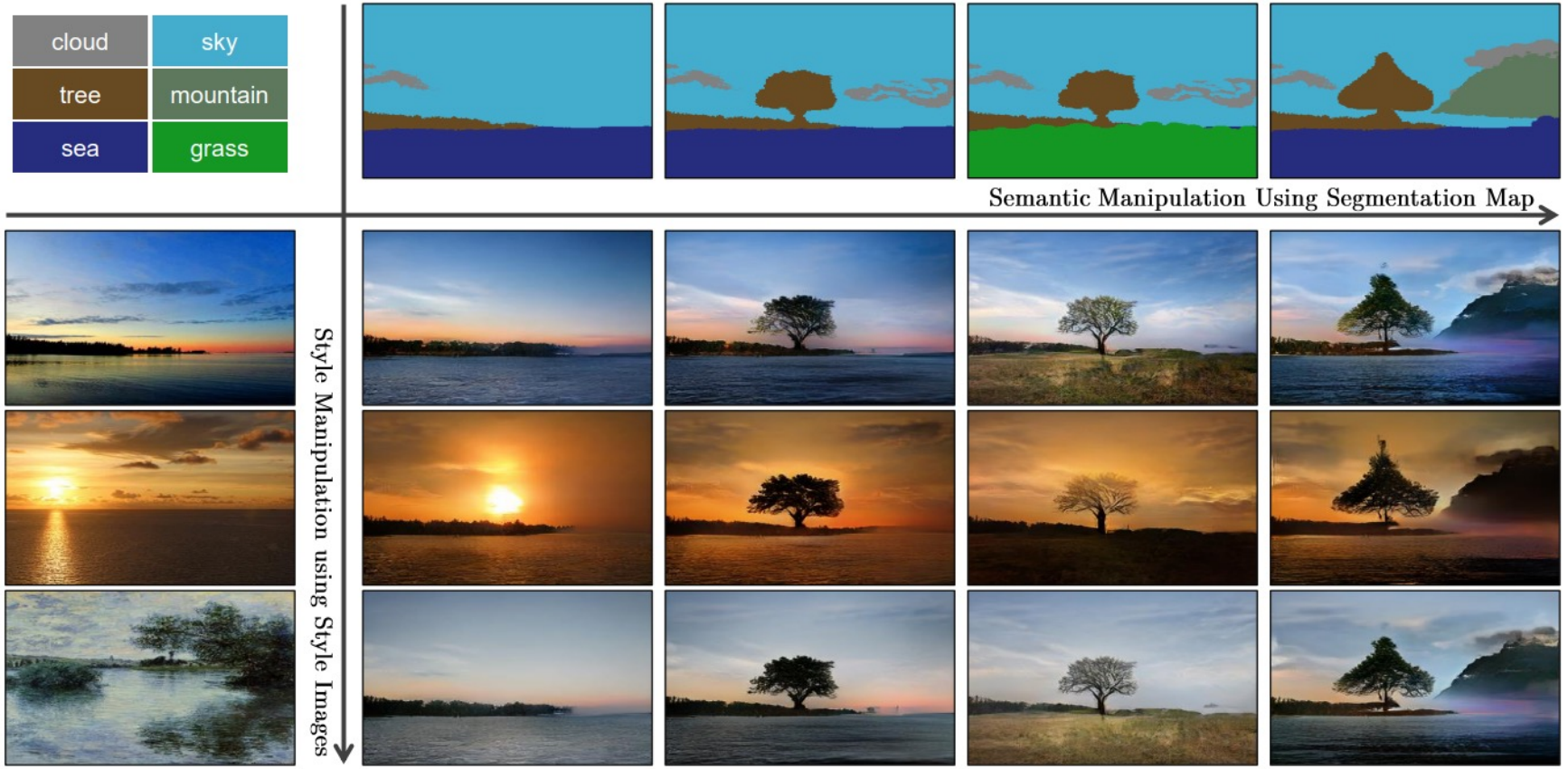


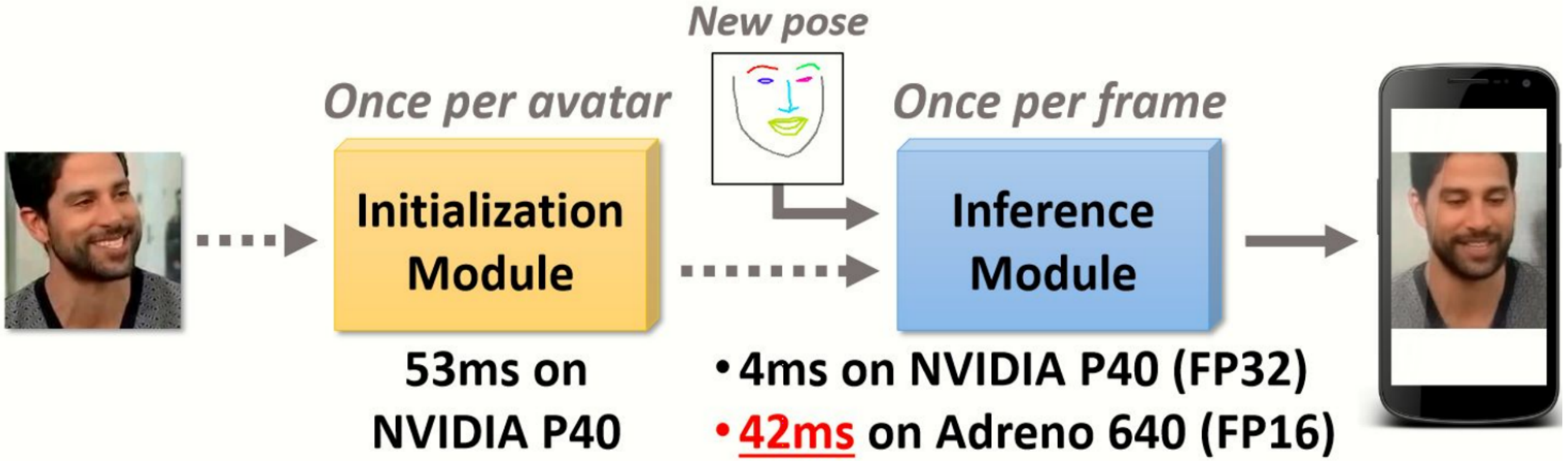
Figure 4: In the SPADE generator, each normalization layer uses the segmentation mask to modulate the layer activations. *(left)* Structure of one residual block with the SPADE. *(right)* The generator contains a series of the SPADE residual blocks with upsampling layers. Our architecture achieves better performance with a smaller number of parameters by removing the downsampling layers of leading image-to-image translation networks such as the pix2pixHD model [48].

Стилизация изображения



- На вход подаётся случайный вектор
- Вместо случайного вектора можно подавать «сжатое» представление стиля, полученное из Encoder-Decoder сети

Bi-layer model



- One-shot initialization
- Mobile inference
- State-of-the-art quality

Идея алгоритма

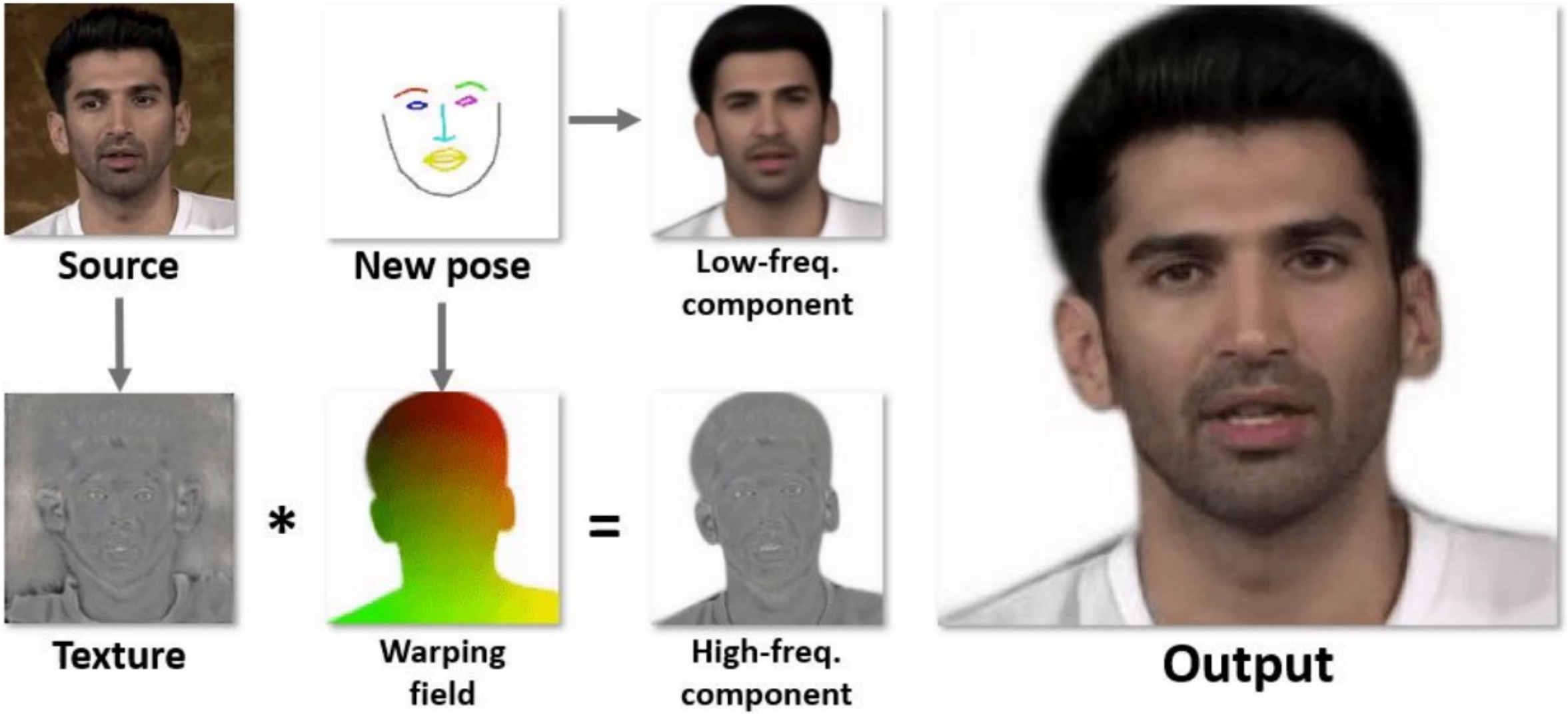
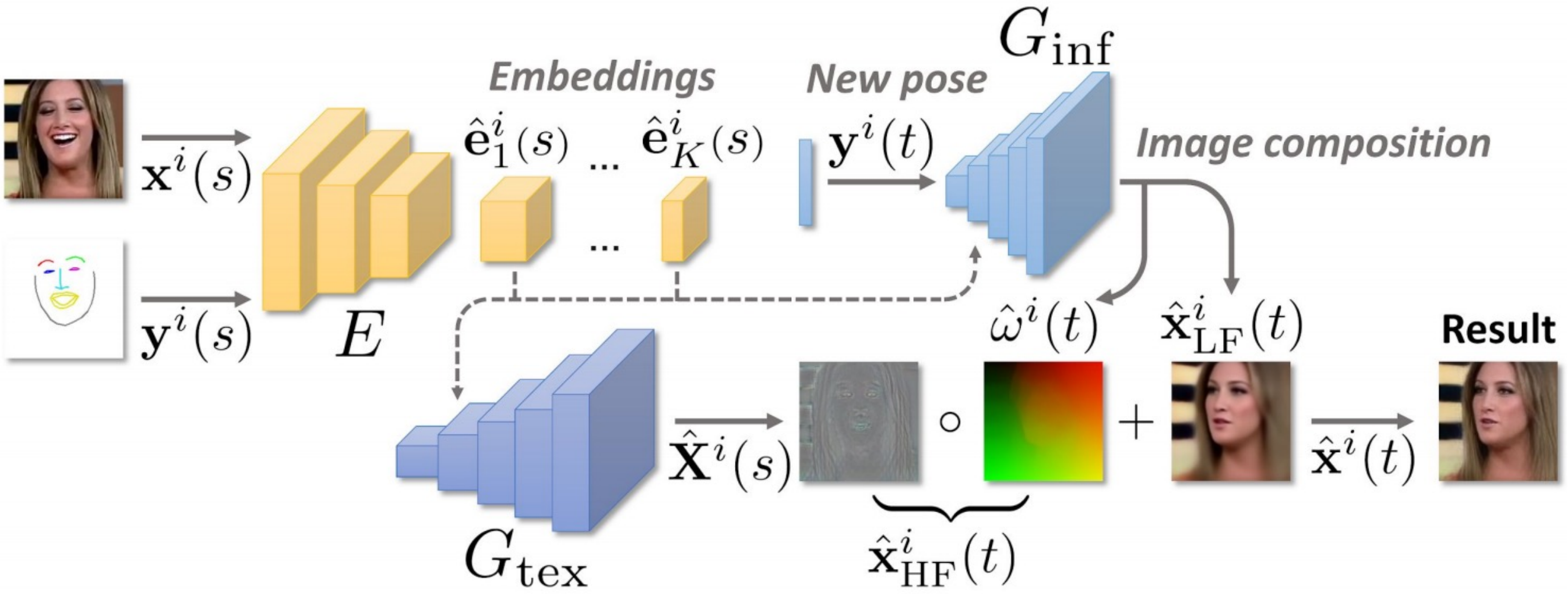
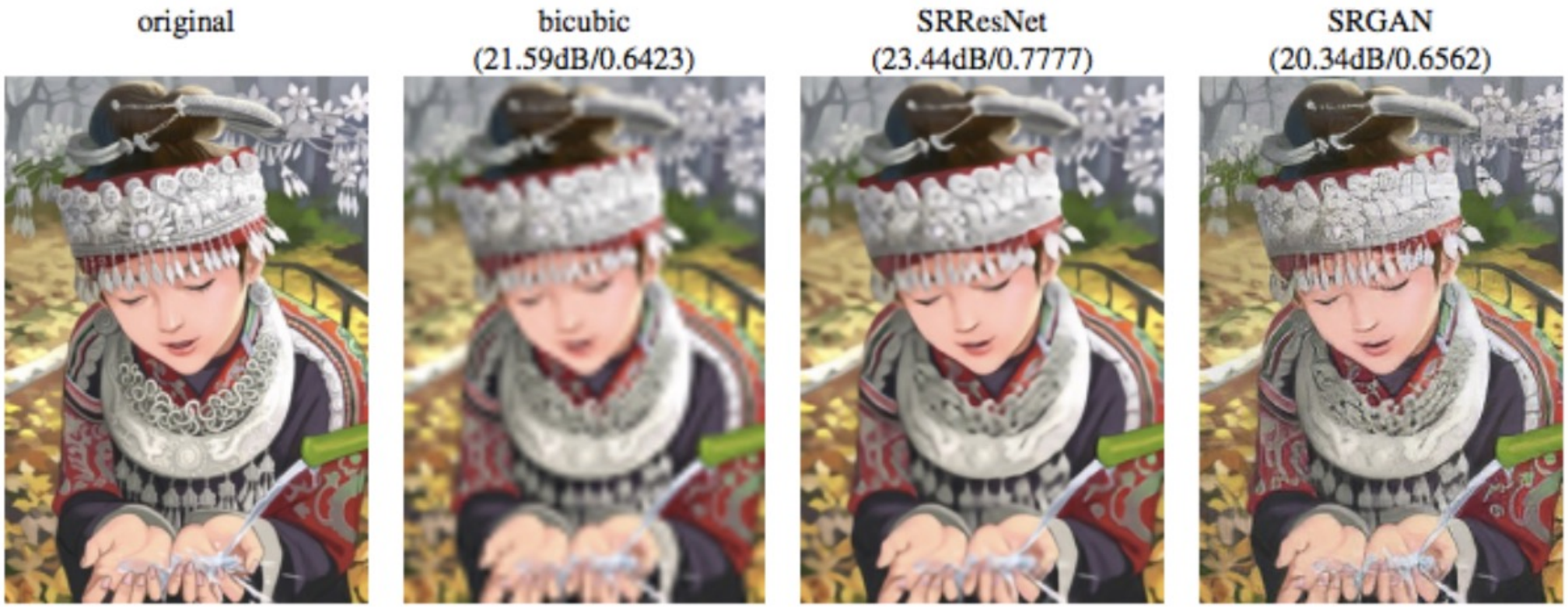


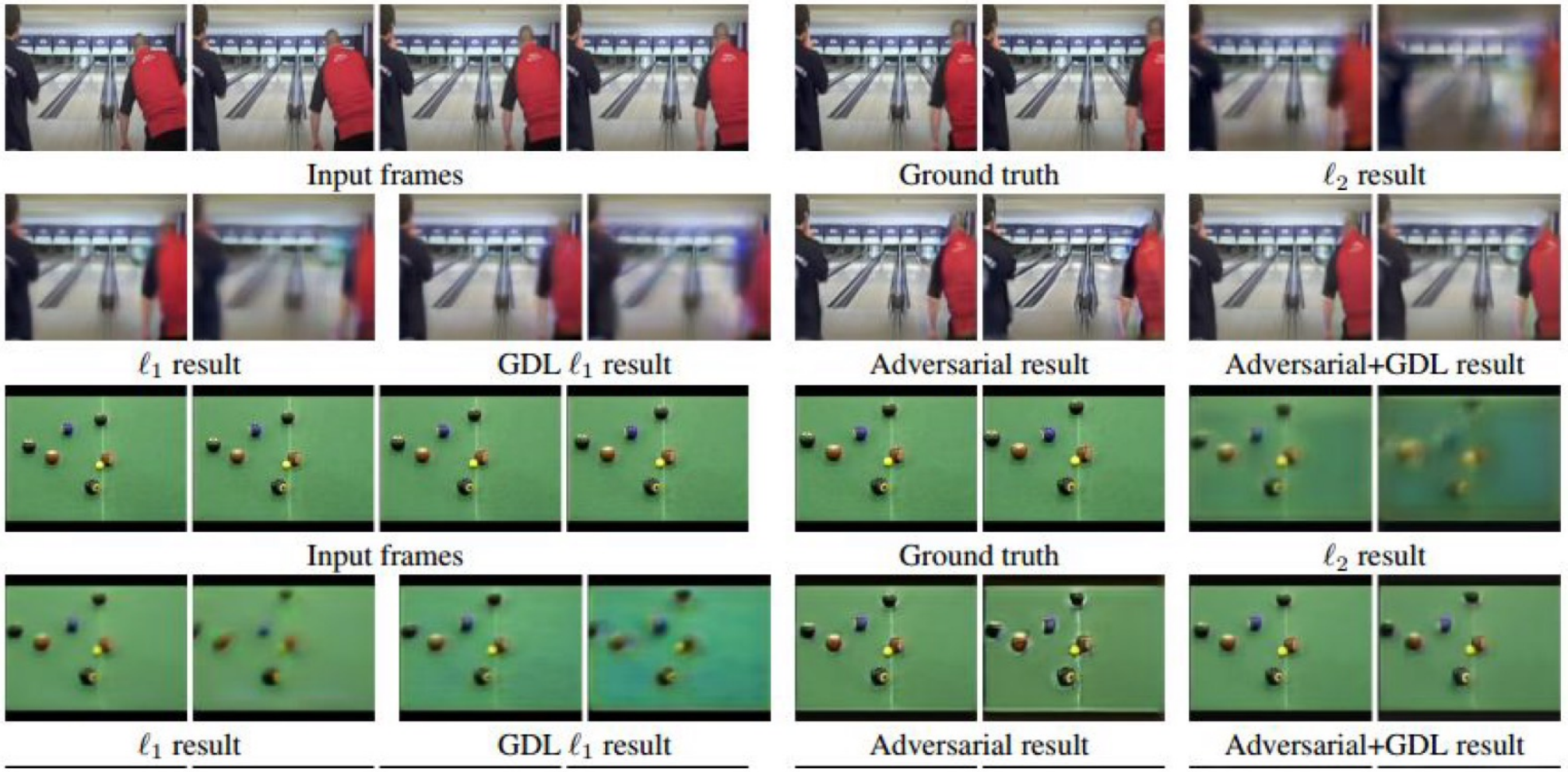
Схема модели



Применение - суперразрешение

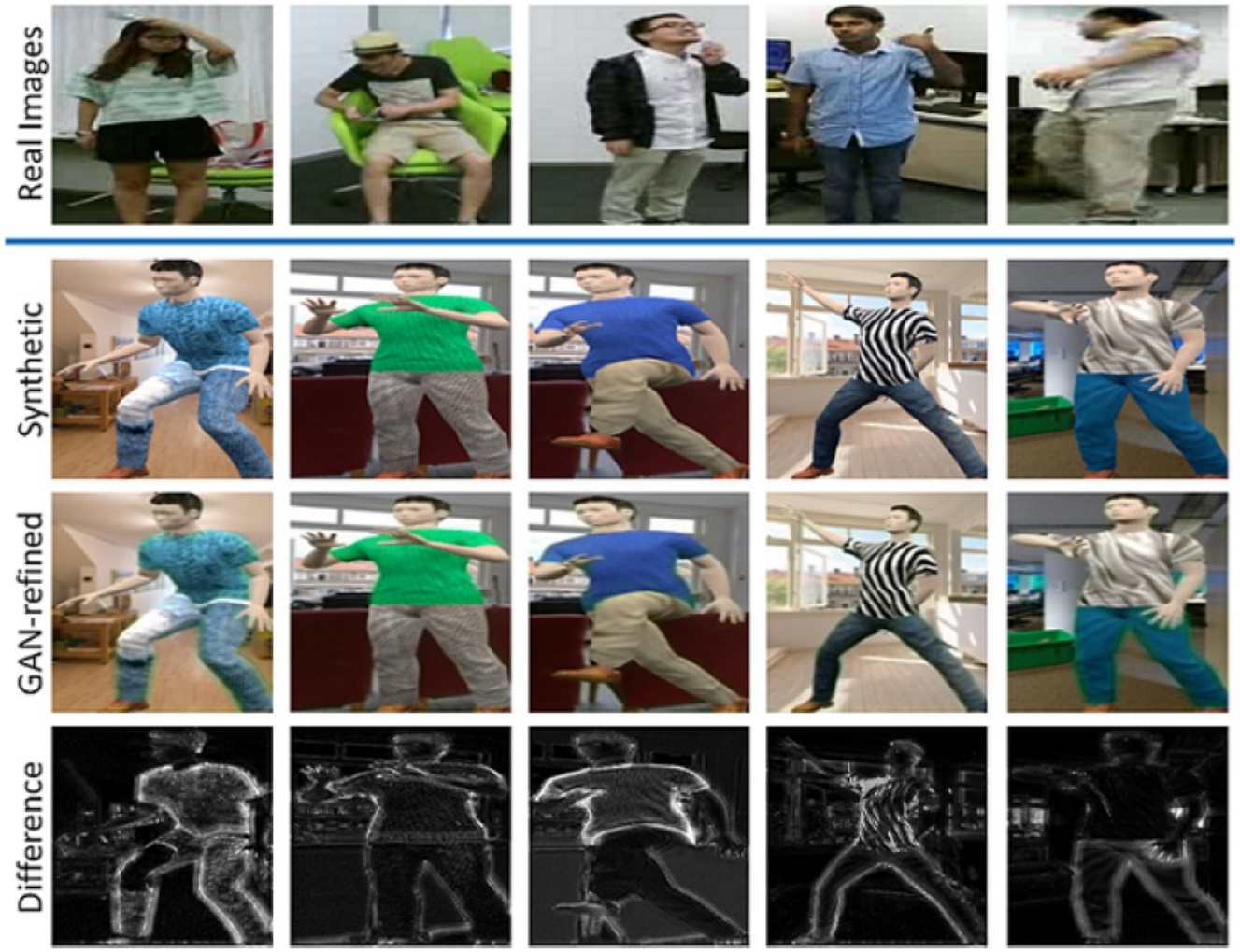


Применение - предсказание кадров



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

Резюме лекции

- Нейропризнаки кодируют и содержание, и внешность изображений.
- Корреляции между признаками описывают «стиль»
- Для преобразования изображений предложены сети кодировщик-декодеровщик
- Похожесть изображений хорошо оценивается с помощью Perception Loss – сравнения значений нейросетевых признаков
- Для оценки сохранения личности можно использовать Identity Loss, вариант Perception Loss по признакам сети, обученной на идентификации человека
- Adversarial Loss от сети классификатора, обучаемой совместно с генератором, позволяет добиться «реалистичности» изображений
- Cycle Loss позволяет работать с произвольными выборками, а не с парами соответствующих изображений
- AdaIN и SPADE слои позволяют управлять процессом «декодирования» изображений из латентов, а параметры для них генерируем управляющими сетями