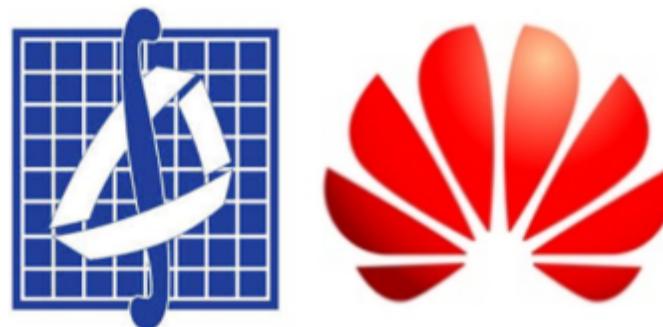


# Image to image translation с использованием нейронных сетей

Михайлов Дмитрий

Лаборатория Интеллектуальных Систем,  
Российский Исследовательский Институт Huawei

29 октября 2019



# План лекции

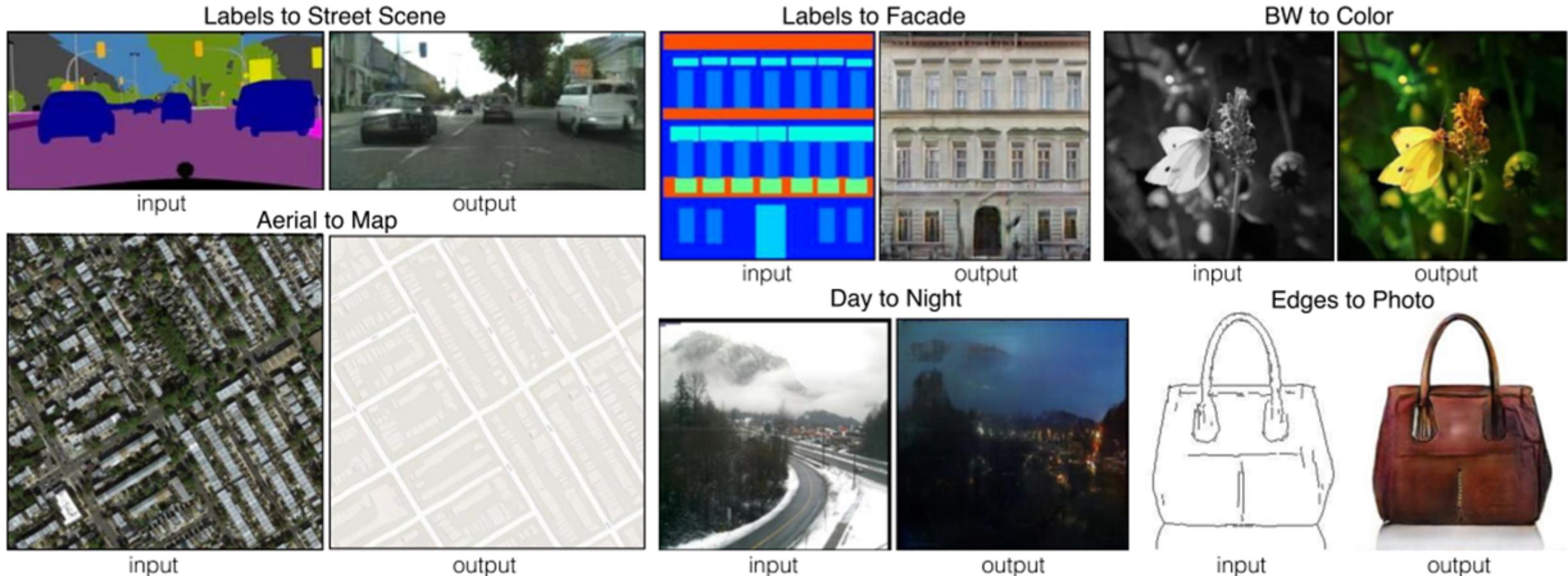
- Задачи, решаемые нейронными сетями
- Задача image to image translation
- Генеративные состязательные сети
- Image Style Transfer
- Pix2Pix
- CycleGAN
- UNIT
- StarGAN
- Vid2Vid
- GANPaint, GauGAN

# Задачи, решаемые нейронными сетями

- Полносвязные сети
  - Поиск сложных полиномиальных отношений (задачи регрессии)
- Рекуррентные сети
  - Поиск отношений во временных рядах (видео, аудио, показания датчиков)
  - Обработка естественных языков
    - перевод
    - генерация текстов
    - семантический/синтаксический разбор текстов
- Графовые сети
  - Поиск отношений в графах
  - Обработка облаков точек

- Свёрточные сети
  - Поиск отношений в данных, имеющих регулярную пространственную структуру (1D, 2D, 3D изображения)
  - Классификация изображений
  - Детекция объектов на изображениях
  - Сегментация
  - Улучшение изображений (Image Enhancement)
    - Улучшение разрешения (Super Resolution)
    - Удаление шума, размытия, погодных эффектов (дождь, снег, туман)
    - Улучшение визуального качества изображения
  - Генерация изображений:
    - Генерация изображений из шума
    - Генерация изображений по вектору параметров и/или шуму
    - Image to Image Translation (генерация изображения на основе другого изображения)

# Image to image translation



# Применения Image to image translation

- Увеличение размера наборов данных для обучения
- Генерация изображений в дизайне, компьютерной графике
- «Умное» редактирование изображений и видео
- Колоризация чёрно-белого кино
- Перенос стиля изображения (Image style transfer)

# Image Style Transfer

Перенос стиля изображения предполагает перенос локальной структуры изображения (стиля изображения) при сохранении глобальной структуры (содержание изображения).

A



B



C



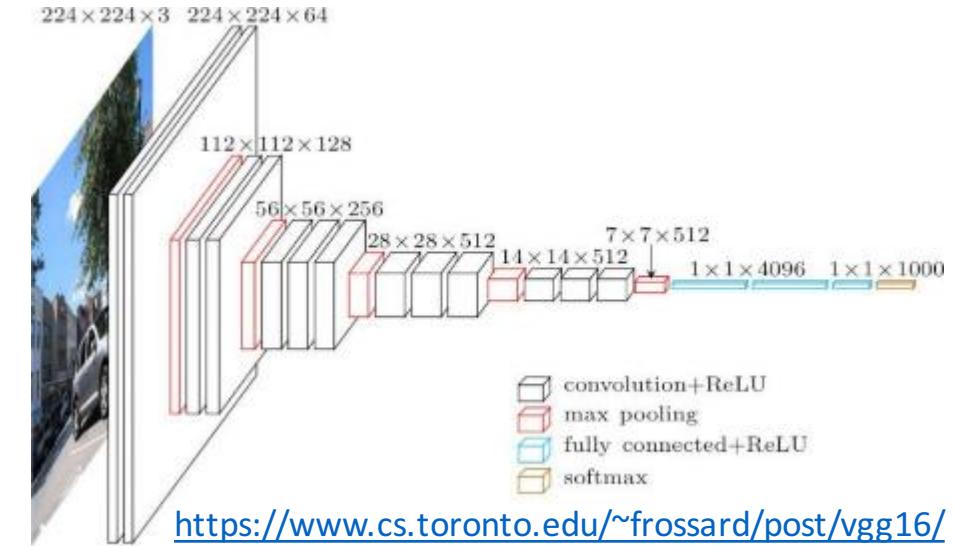
D



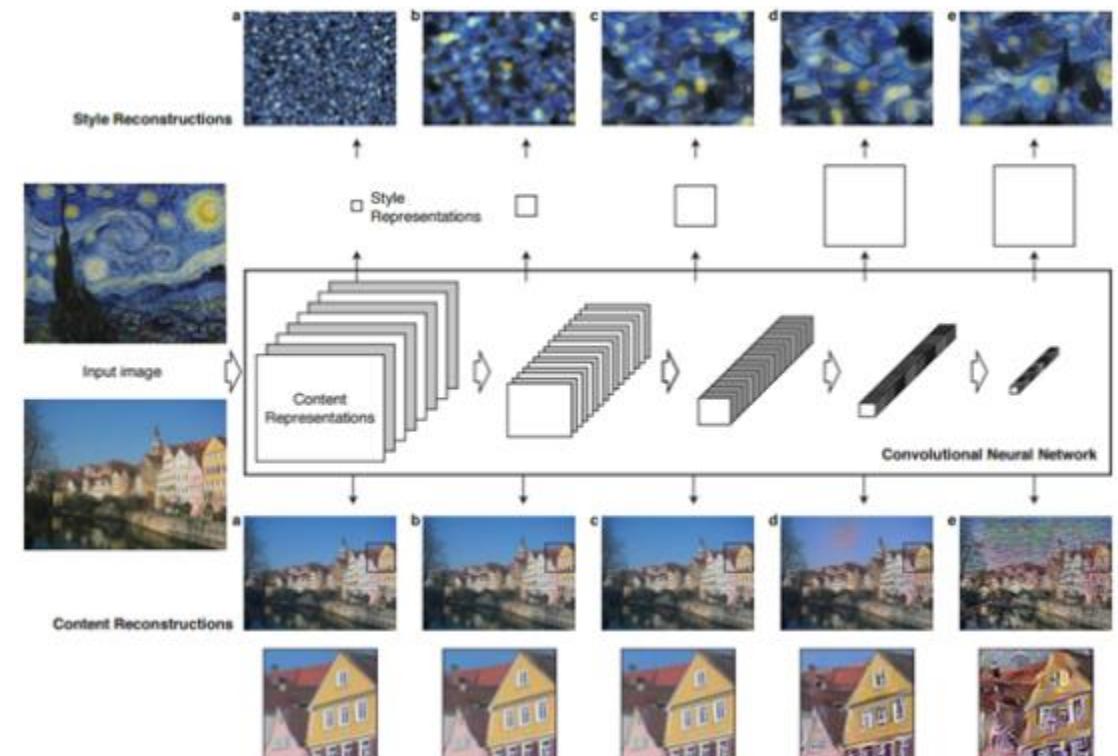
# Как это работает?

- Берём обученную VGG-сеть без полно связных слоёв на конце.
- Пропускаем через сеть изображение стиля и целевое изображение, получаем карты активаций сети F.
- Считаем матрицу Грама:  $G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$ .  
(l – слой, i,j – карты признаков, k – элементы карт)
- Замораживаем веса сети.
- Инициализируем изображение белым шумом.
- Прямое прохождение: пропускаем изображение через сеть, получаем карты активаций P. Для них тоже считаем матрицу Грама A.
- Считаем функцию потерь для содержимого изображения:  $\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$
- Считаем функцию потерь для стиля изображения:  
 $E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2 \quad \mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^L w_l E_l$
- Делаем обратное прохождение, где пиксели изображения выступают в качестве переменных.

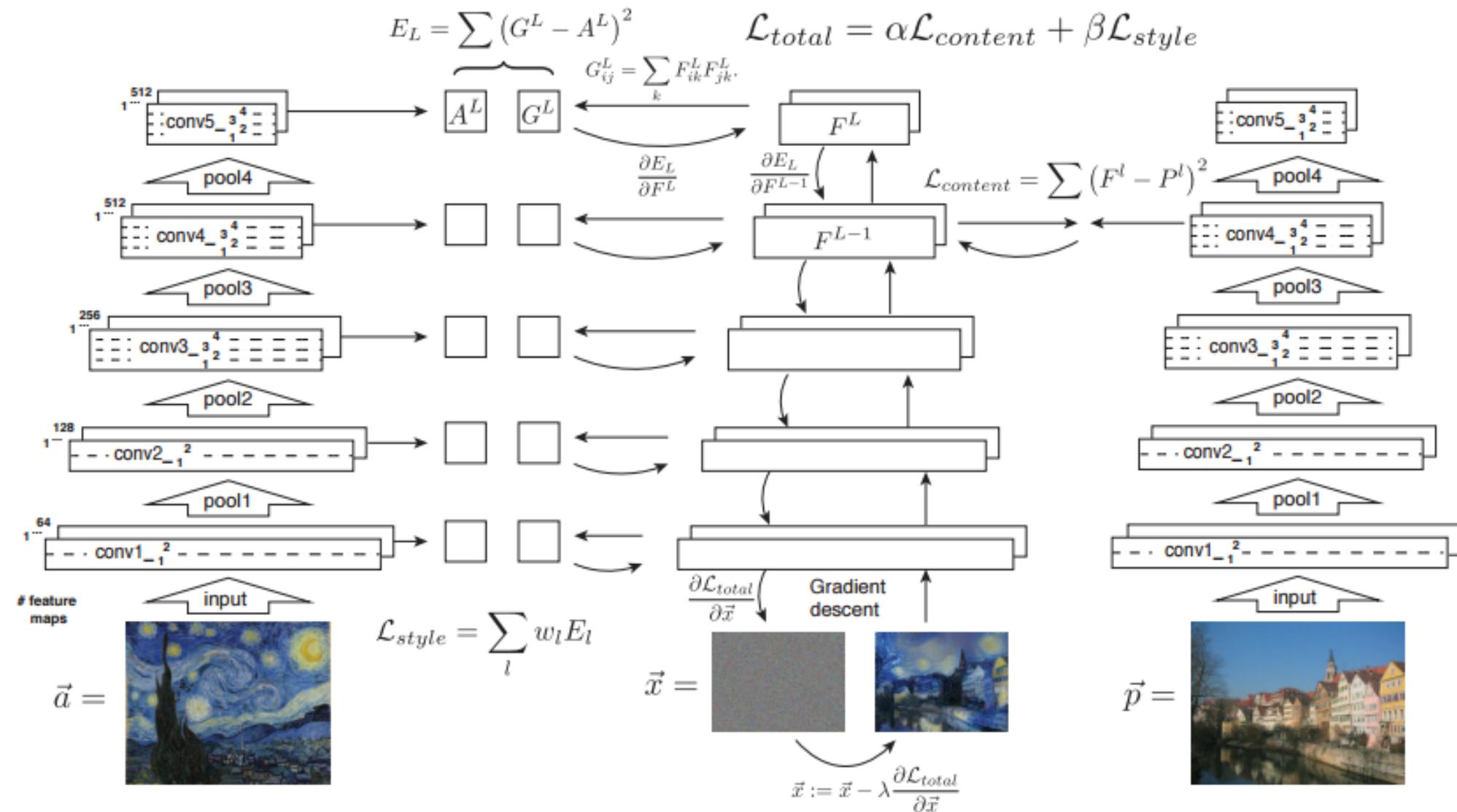
$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$



<https://www.cs.toronto.edu/~frossard/post/vgg16/>



Leon A. Gatys et al. 2016



Style



Original



Result



Style



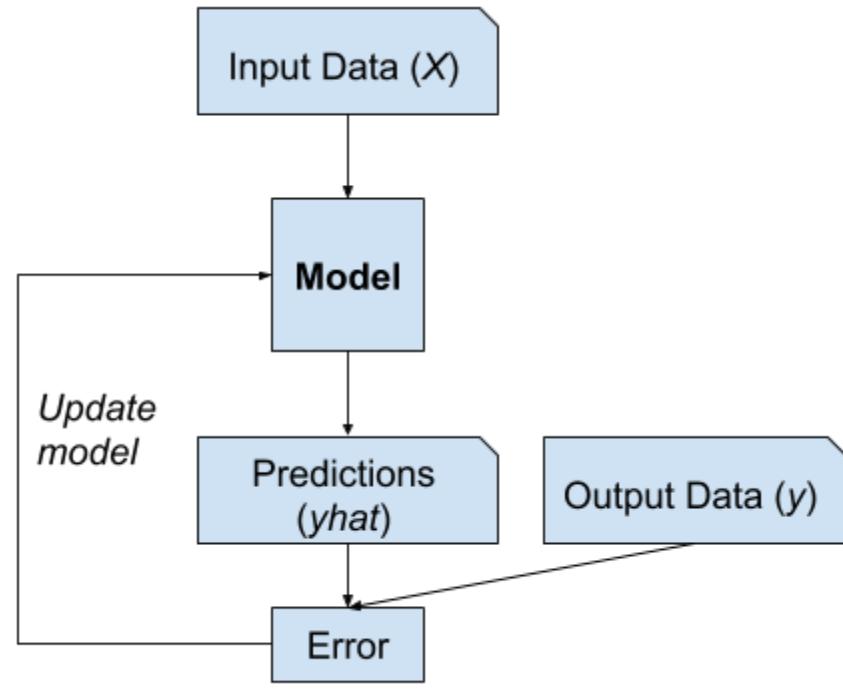
Original



Result



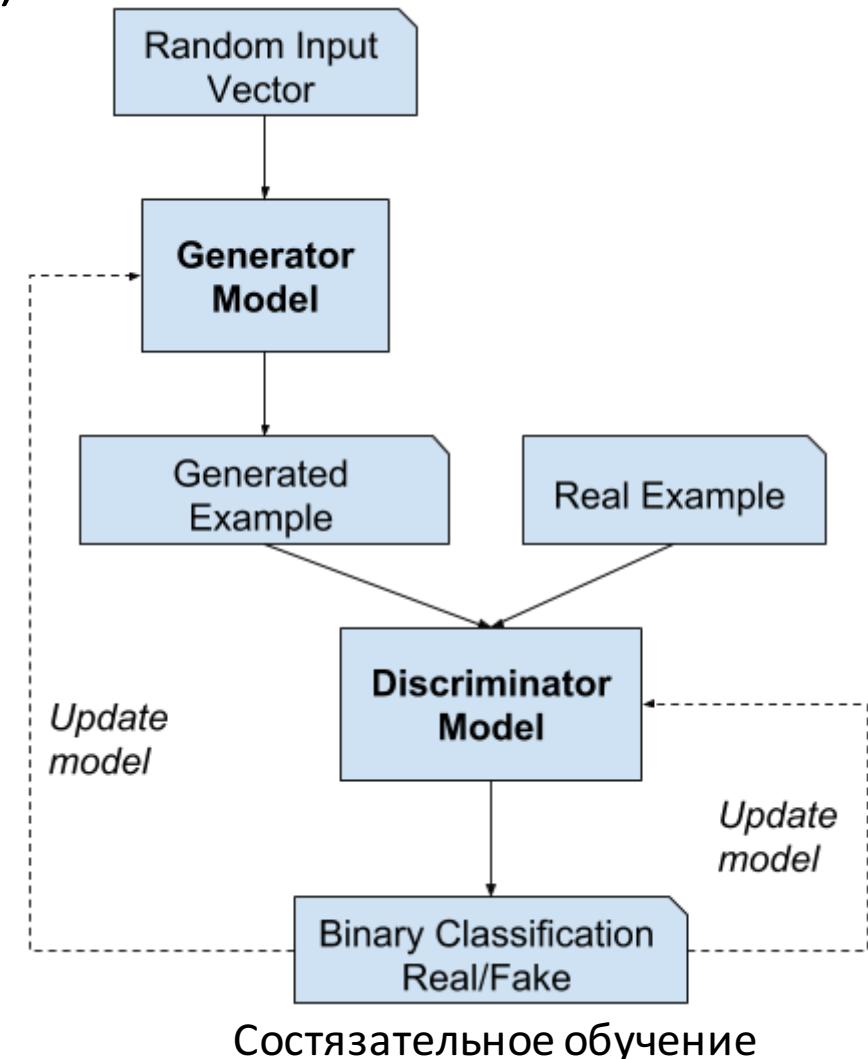
# Генеративные состязательные сети (Generative Adversarial Networks, GAN)



Обучение с учителем

$$J^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) = -\frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log D(\mathbf{x}) - \frac{1}{2} \mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z})))$$

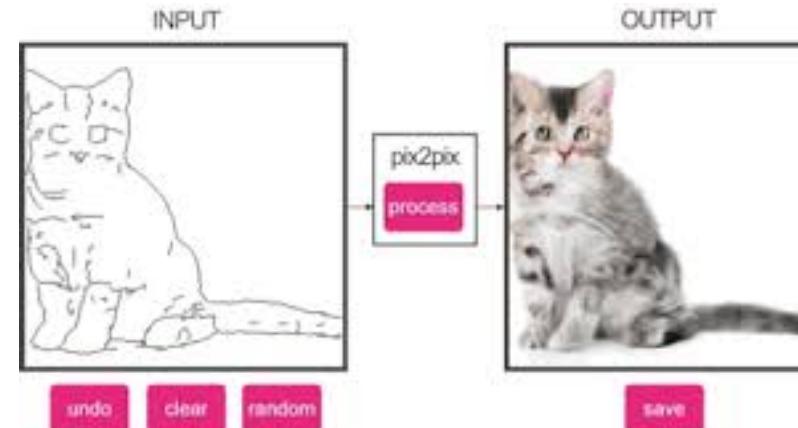
$$J^{(G)} = -\frac{1}{2} \mathbb{E}_{\mathbf{z}} \log D(G(\mathbf{z}))$$



Состязательное обучение

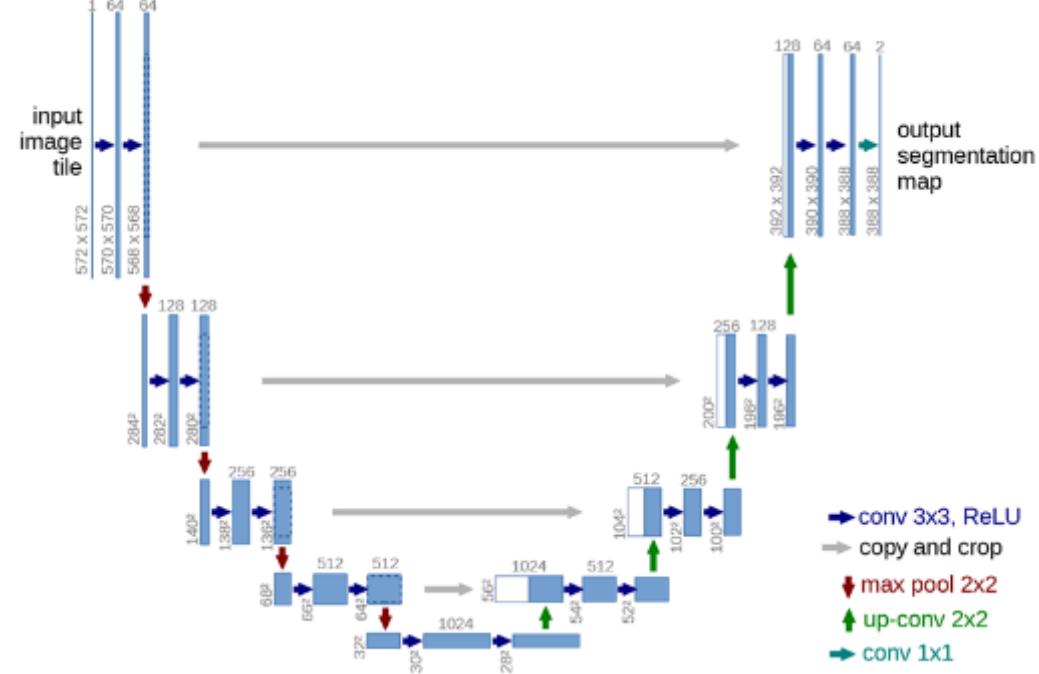
# Pix2pix

- Pix2pix – первая генеративная состязательная нейронная сеть, которая стала известна широкой публике.
- Она может быть применена для широкого круга задач:
  - Границы <-> Изображения
  - Изображения низкого разрешения <-> Изображения высокого разрешения
  - Сегментация <-> Изображение
  - Чёрно-белое изображение <-> Цветное изображение
  - Ночное изображение <-> Дневное изображение
- Online-демонстрация доступна по ссылке <https://affinelayer.com/pixsrv/>



# Устройство сети

Pix2Pix основана на архитектуре U-Net:



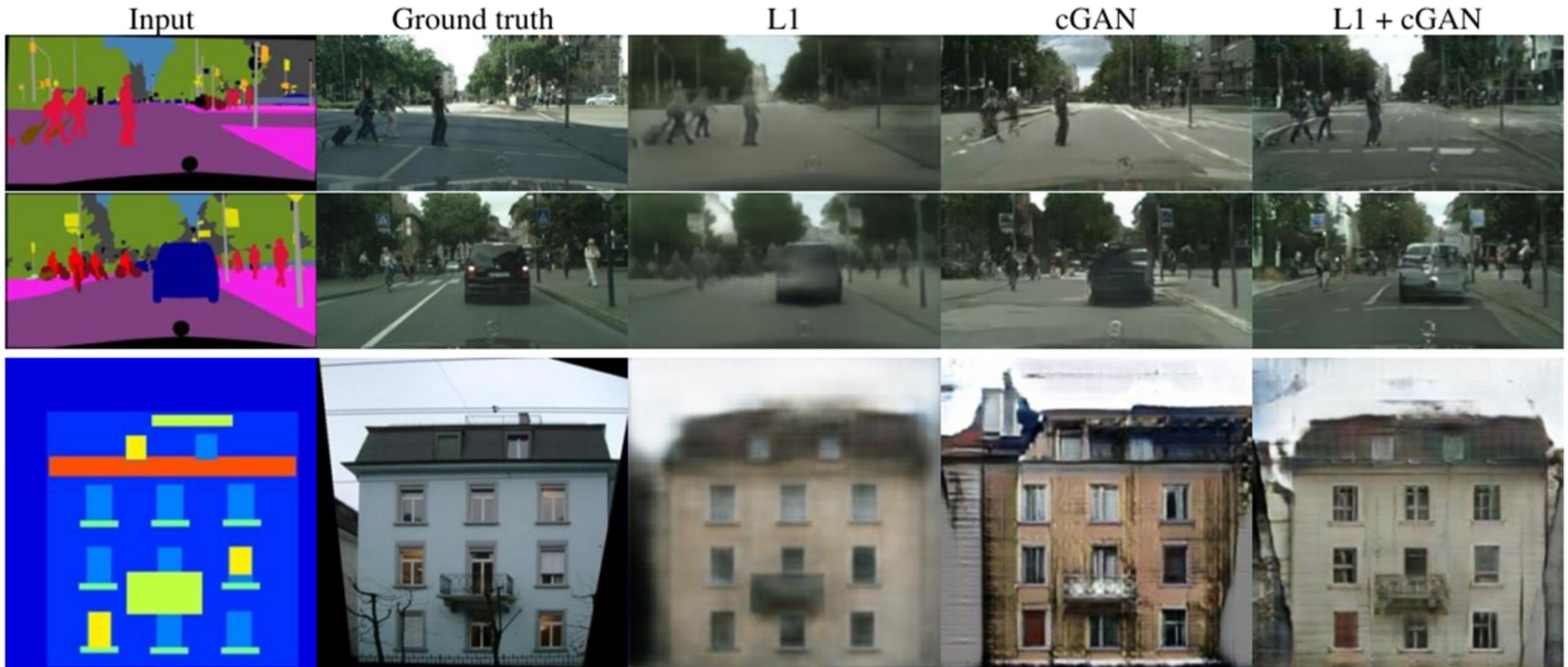
В качестве функции потерь выступает сумма:

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

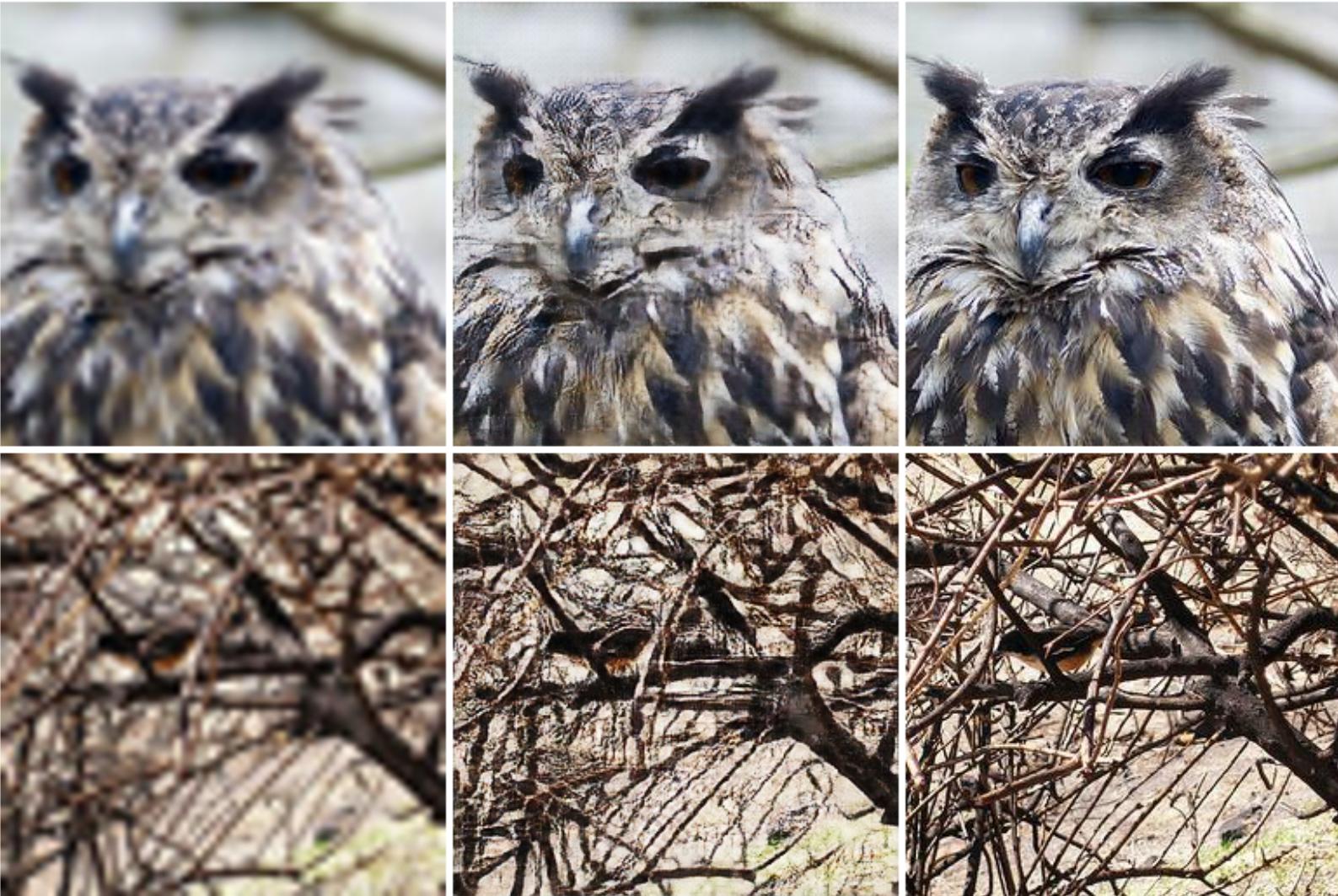
$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x, y \sim p_{data}(x, y), z \sim p_z(z)} [\|y - G(x, z)\|_1]$$

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

# Влияние функции потерь на результат

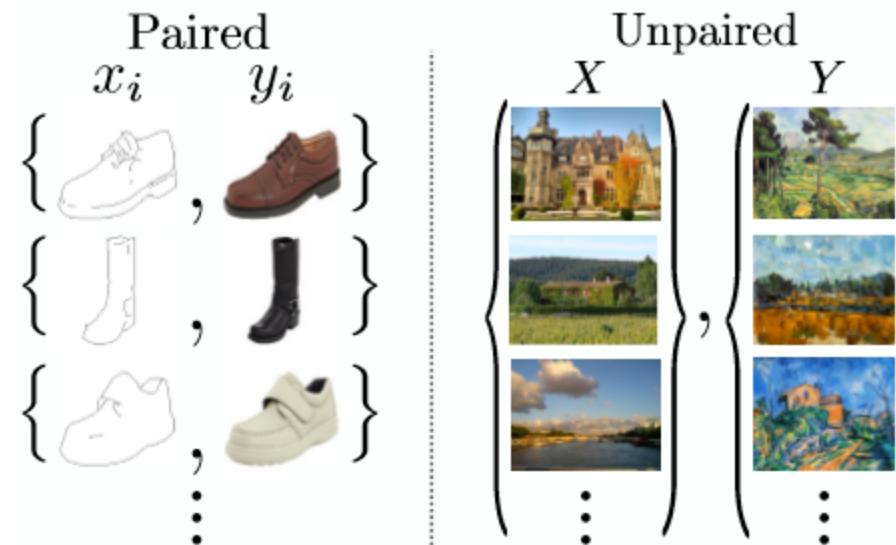


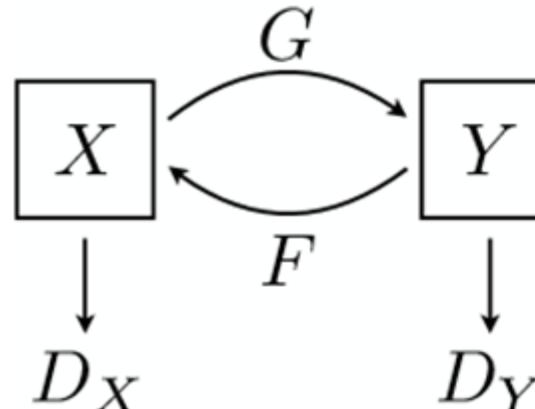
# Pix2Pix в задаче увеличения разрешения изображения



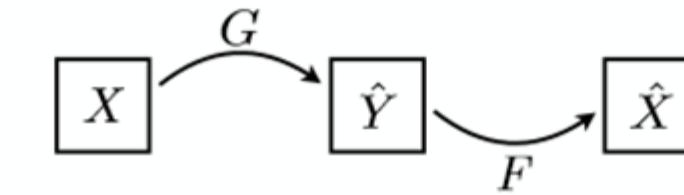
# Cycle-GAN

- Cycle-GAN – это дальнейшее развитие Pix2Pix.
- Главной проблемой Pix2Pix была необходимость сбора обучающей базы с попиксельным соответствием между входным и целевым изображениями (для L1-loss), что на практике очень сложно сделать.
- Cycle-GAN решает эту проблему обучением не на парах вход-выход, а на множествах входов и множествах выходов

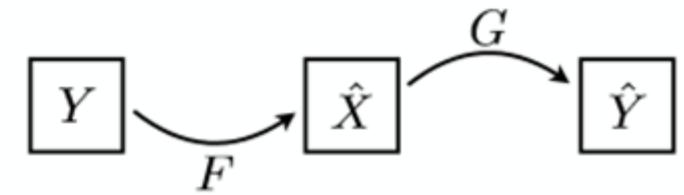




(a)

cycle-consistency  
loss

(b)

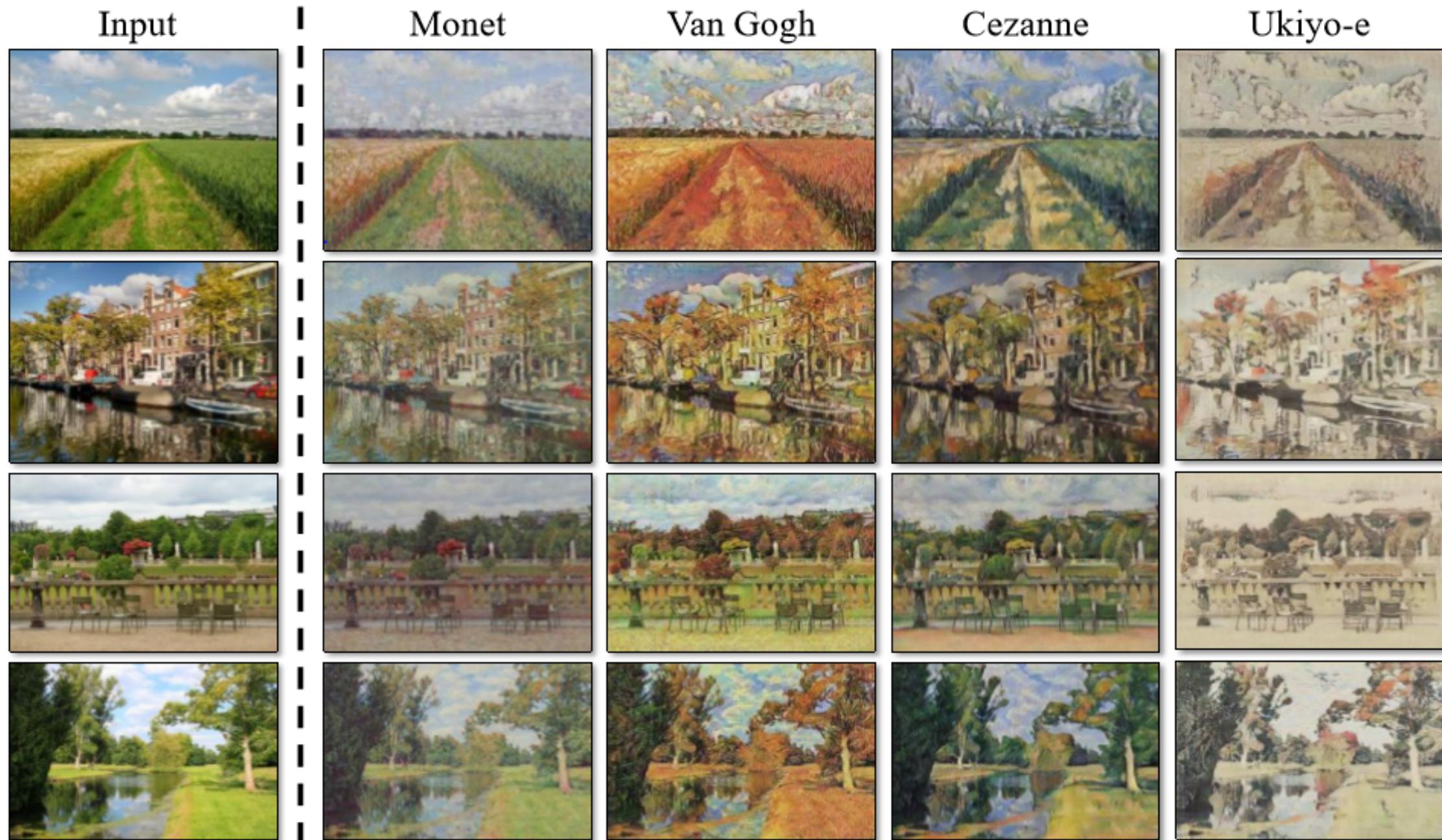
cycle-consistency  
loss

(c)

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) &= \mathbb{E}_{y \sim p_{\text{data}}(y)} [\log D_Y(y)] \\ &\quad + \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log(1 - D_Y(G(x)))]\end{aligned}$$

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) &= \mathbb{E}_{x \sim p_{\text{data}}(x)} [\|F(G(x)) - x\|_1] \\ &\quad + \mathbb{E}_{y \sim p_{\text{data}}(y)} [\|G(F(y)) - y\|_1].\end{aligned}$$

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



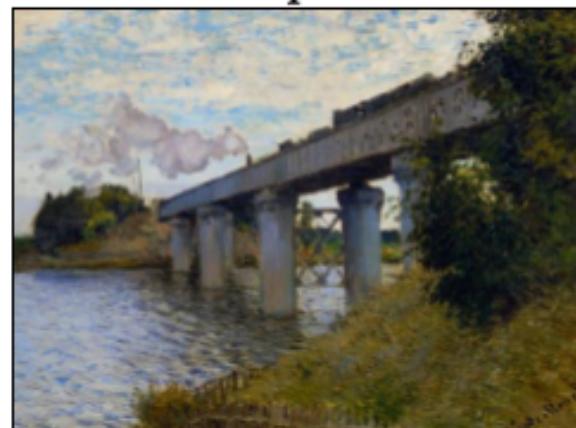
Input



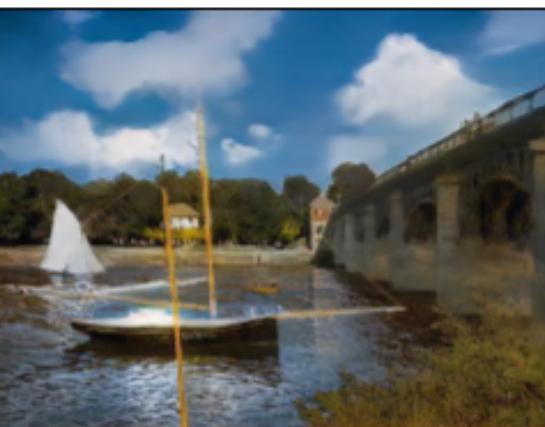
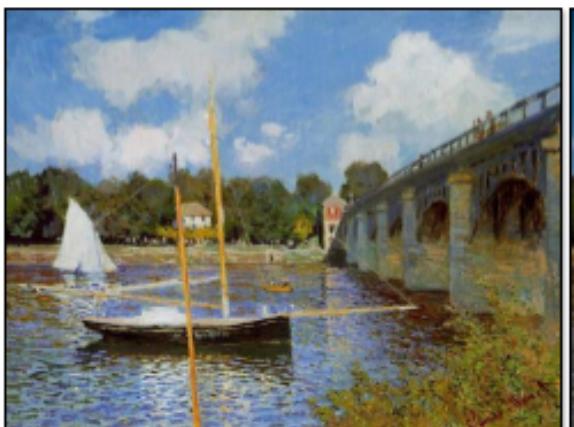
Output

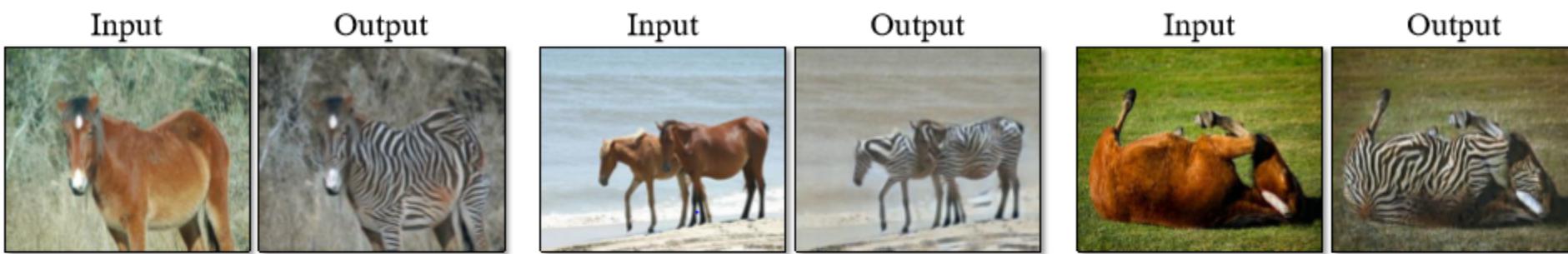


Input



Output





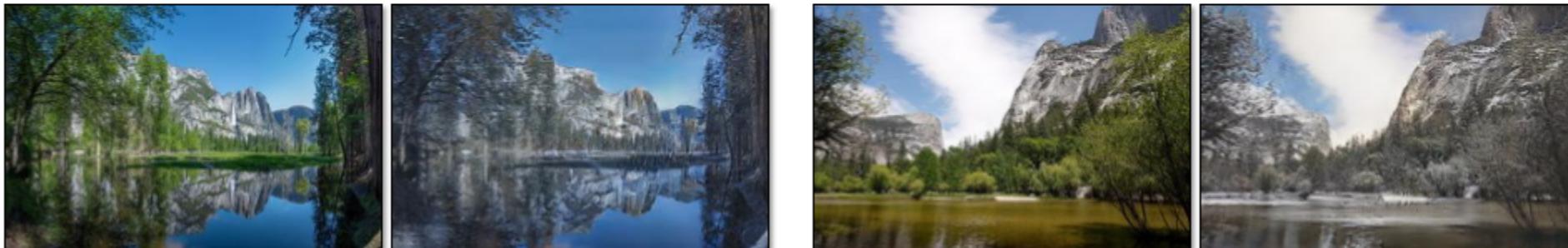
horse → zebra



zebra → horse



winter Yosemite → summer Yosemite



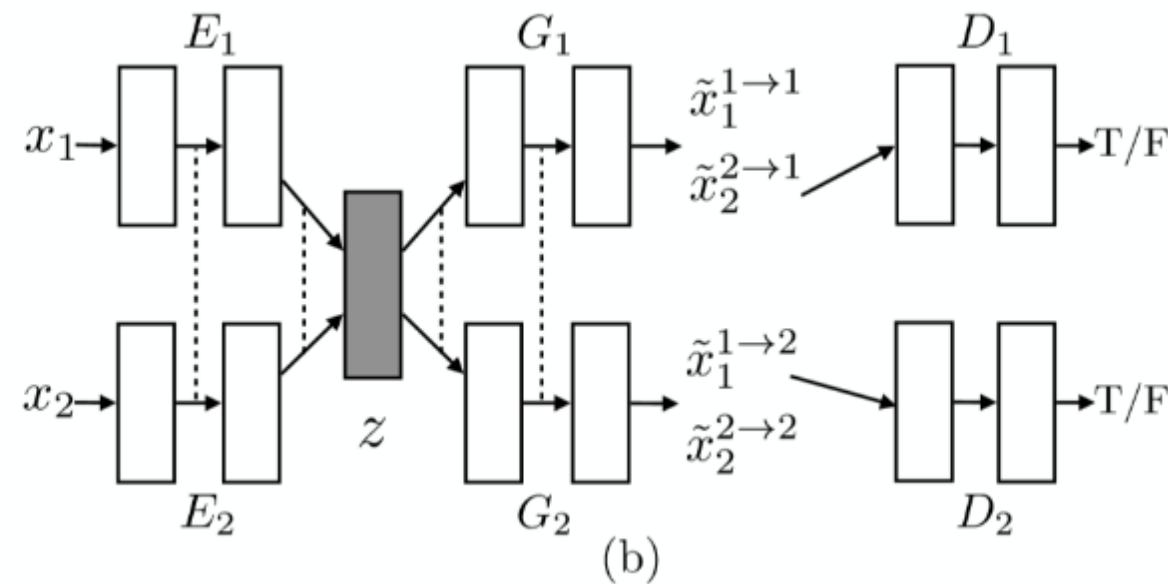
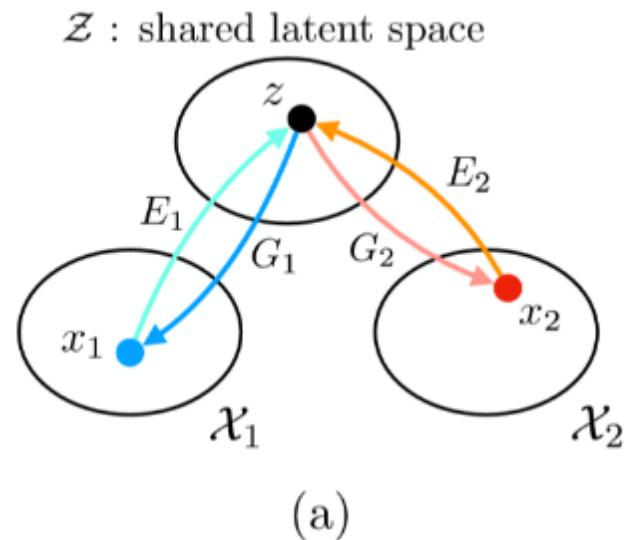
summer Yosemite → winter Yosemite

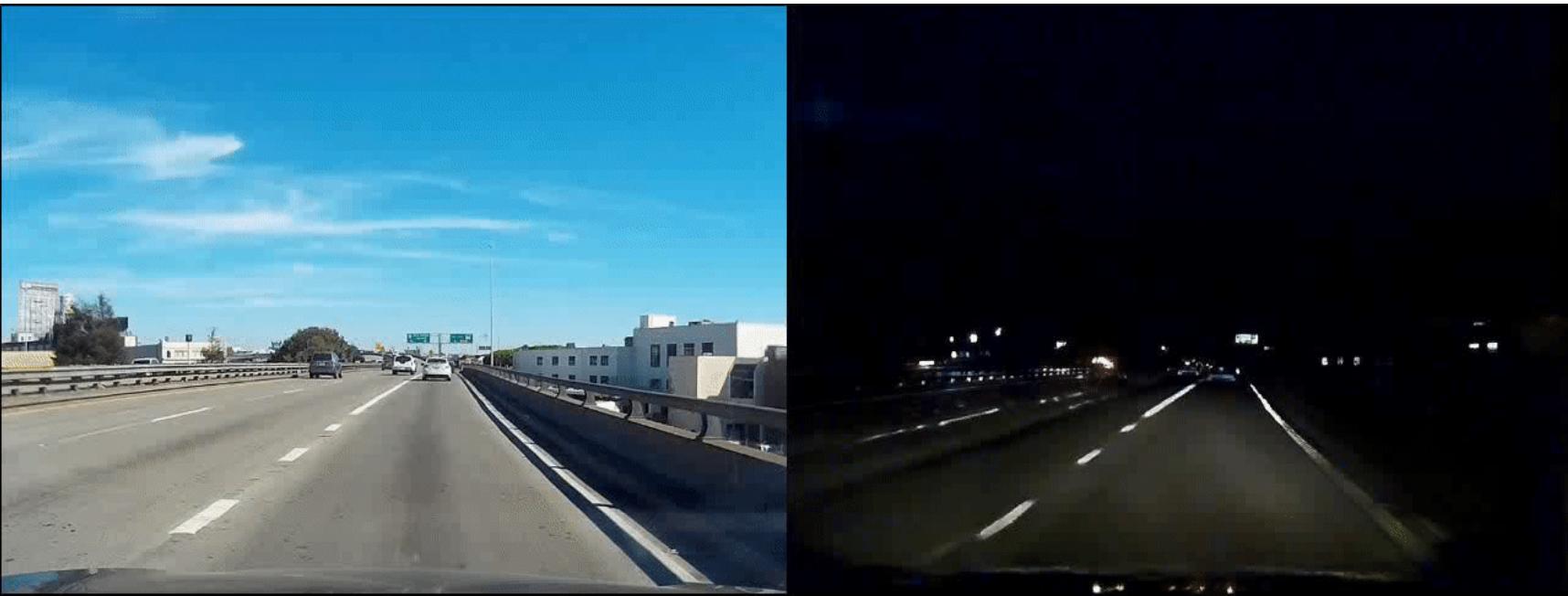
# UNIT

CycleGAN стал прародителем множества других сетей.

Один из примеров - сеть UNIT.

Она использует ту же идею, но каждый из генераторов CycleGAN она разбивает на кодировщик и декодировщик (т.е. модулей в сети становится 6) и объединяет скрытые пространства генераторов. Это сильно улучшает качество результата.





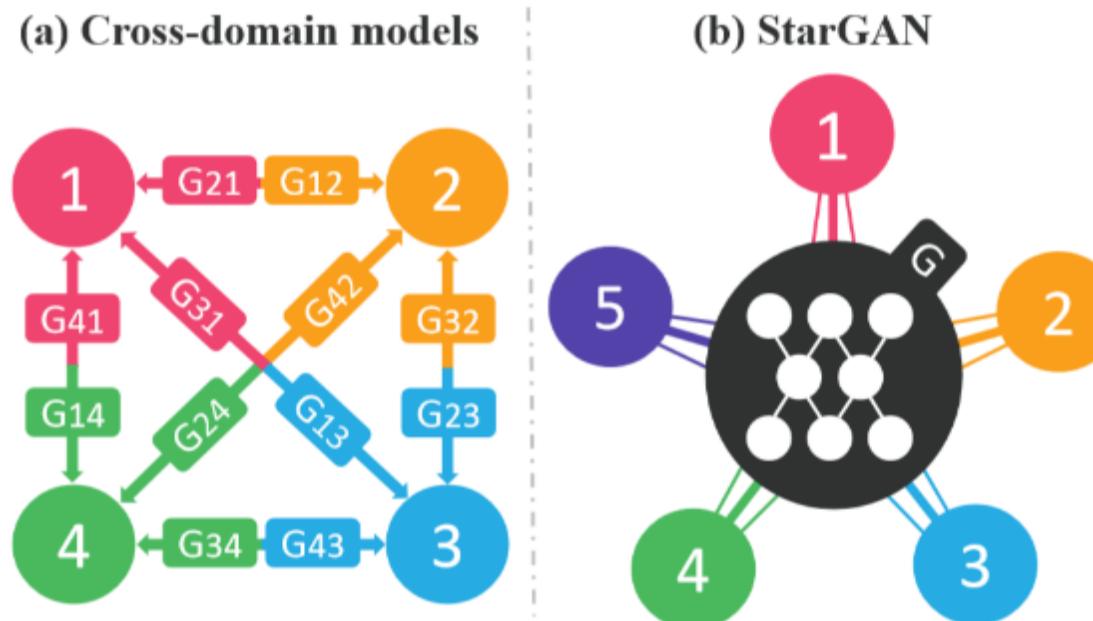
# StarGAN

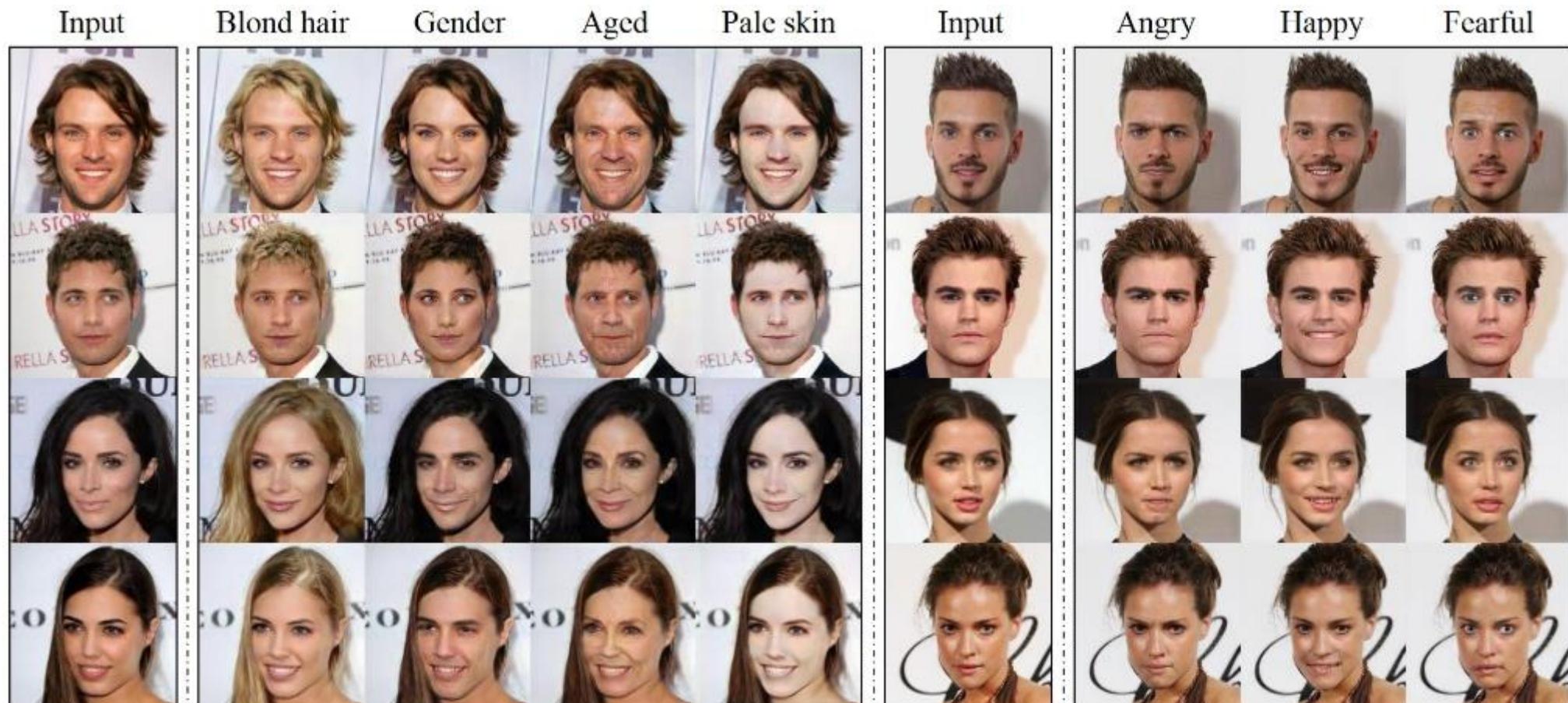
Pix2Pix, CycleGAN, UNIT работают с двумя множествами изображений (доменами). Но что, если доменов нужно больше?

Обучать  $N^*(N-1)/2$  сетей?

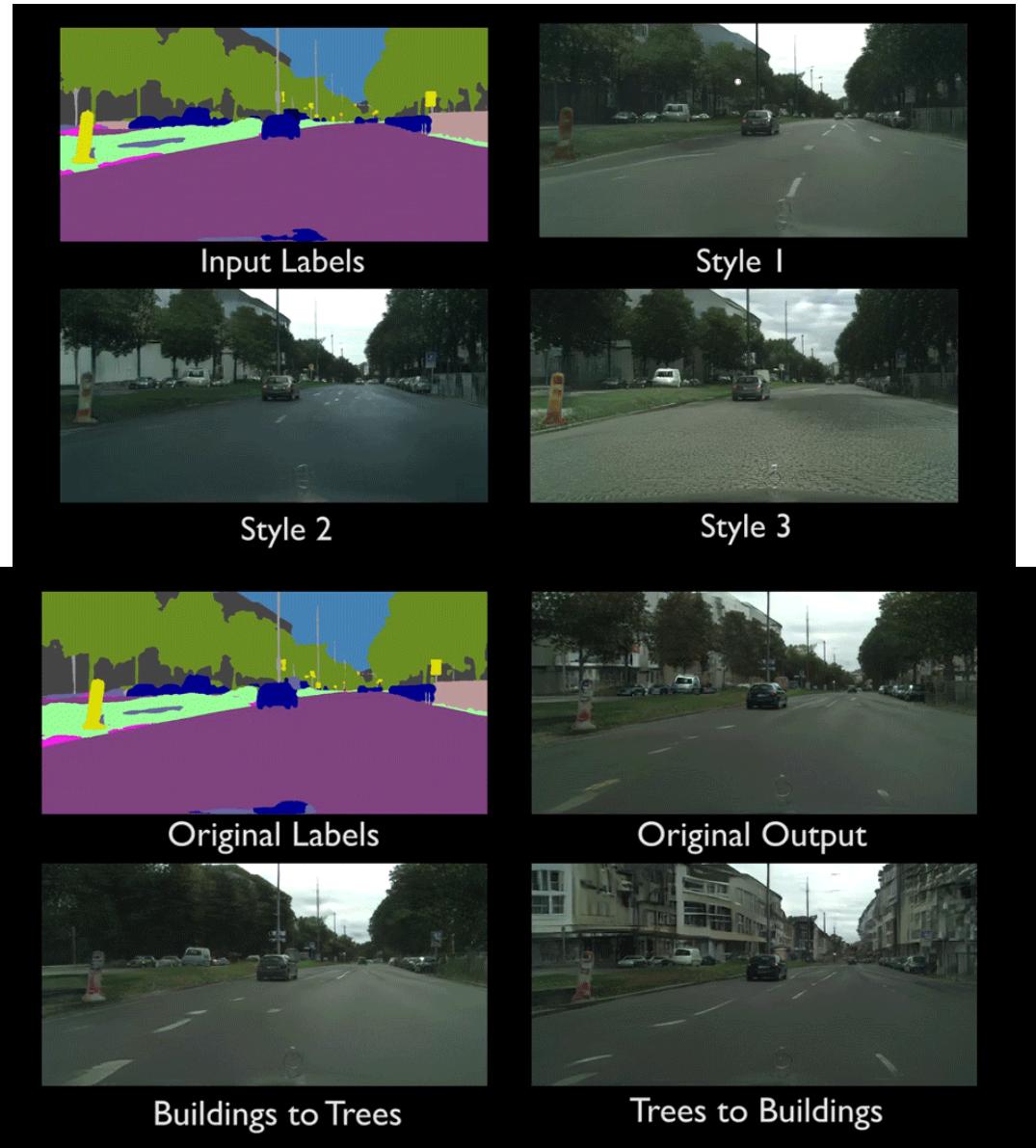
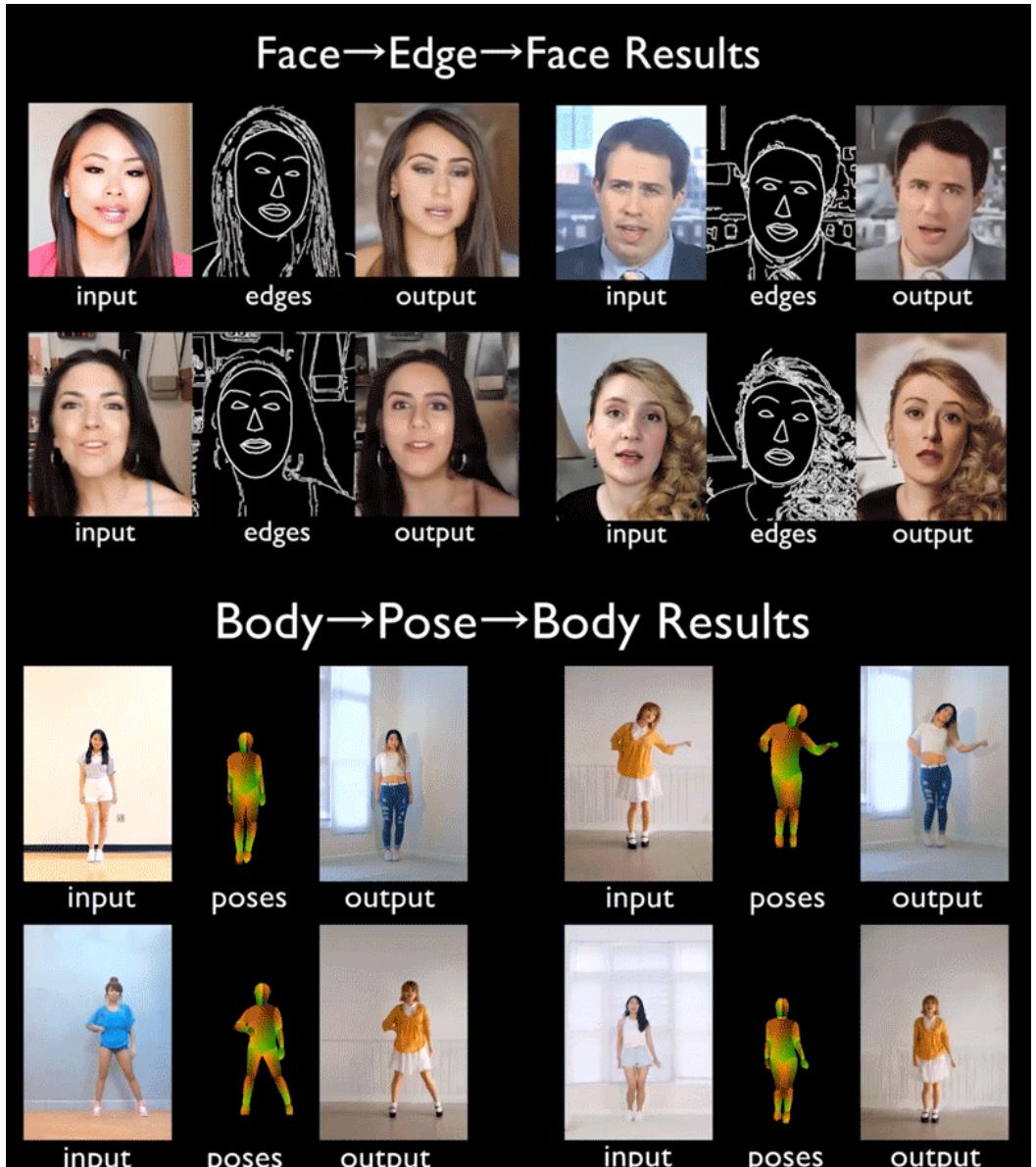
StarGAN (и MUNIT) развивают идею UNIT и решают эту проблему.

Они используют общее скрытое пространство признаков сразу для множества доменов.





# Vid2Vid



# GANPaint, GauGAN

Select a feature brush & strength and enjoy painting:

tree  
grass  
door  
sky  
cloud  
brick  
**dome**



<http://ganpaint.io/demo/?project=church>

<https://www.theverge.com/2019/3/19/18272602/ai-art-generation-gan-nvidia-doodle-landscapes>

# 4.5 years of GAN Progress



2014



2015



2016



2017



2018

<https://arxiv.org/abs/1406.2661>  
<https://arxiv.org/abs/1511.06434>  
<https://arxiv.org/abs/1606.07536>  
<https://arxiv.org/abs/1710.10196>  
<https://arxiv.org/abs/1812.04948>

[https://twitter.com/goodfellow\\_ian/status/1084973596236144640](https://twitter.com/goodfellow_ian/status/1084973596236144640)

Спасибо за внимание!