

Generative Adversarial Networks

Илья Захаркин, Даниил Лыков

ФПМИ МФТИ

DLSchool, 2018

Сегодня в программе

1 Что нужно знать

- Воодушевляющий пример
- Идея GAN
- Формальная постановка задачи

2 Алгоритм

- Псевдокод
- Реализация на Python

3 Применения

- Картинки
- Видео
- Тексты
- Аудио

План

1 Что нужно знать

- Вдохновляющий пример
- Идея GAN
- Формальная постановка задачи

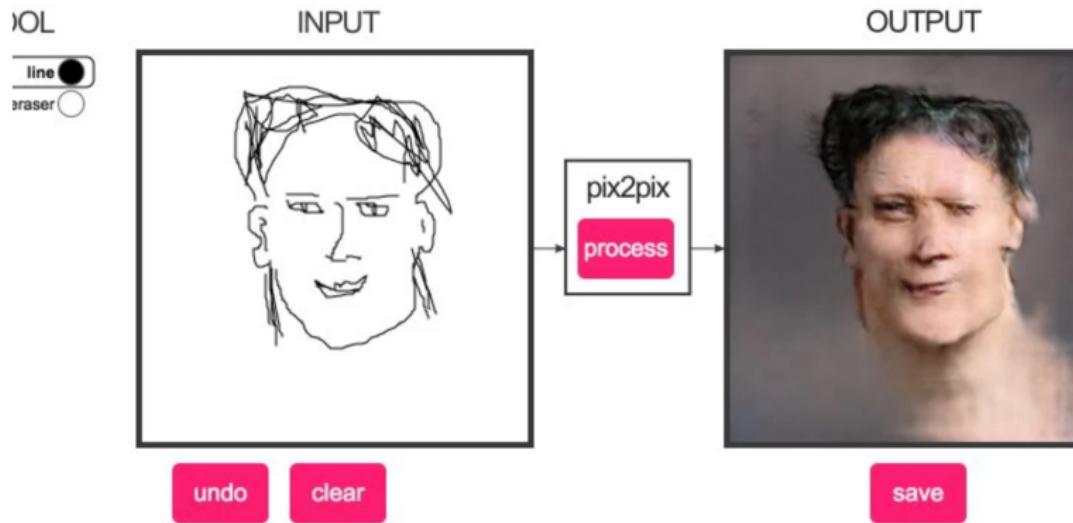
2 Алгоритм

- Псевдокод
- Реализация на Python

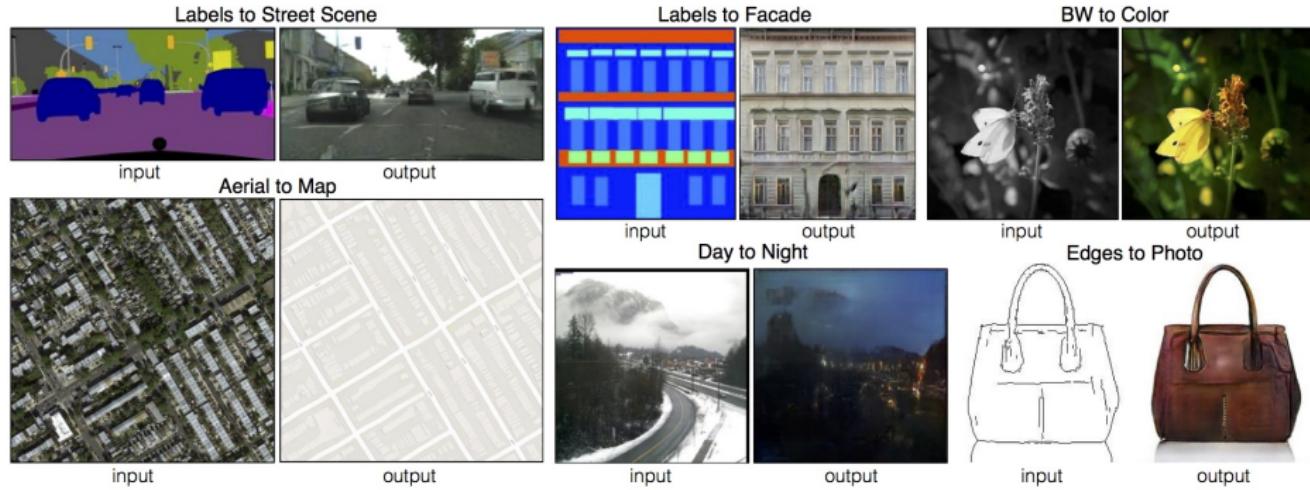
3 Применения

- Картинки
- Видео
- Тексты
- Аудио

Ну почти...



Нормальный пример



План

1 Что нужно знать

- Воодушевляющий пример
- Идея GAN
- Формальная постановка задачи

2 Алгоритм

- Псевдокод
- Реализация на Python

3 Применения

- Картинки
- Видео
- Тексты
- Аудио

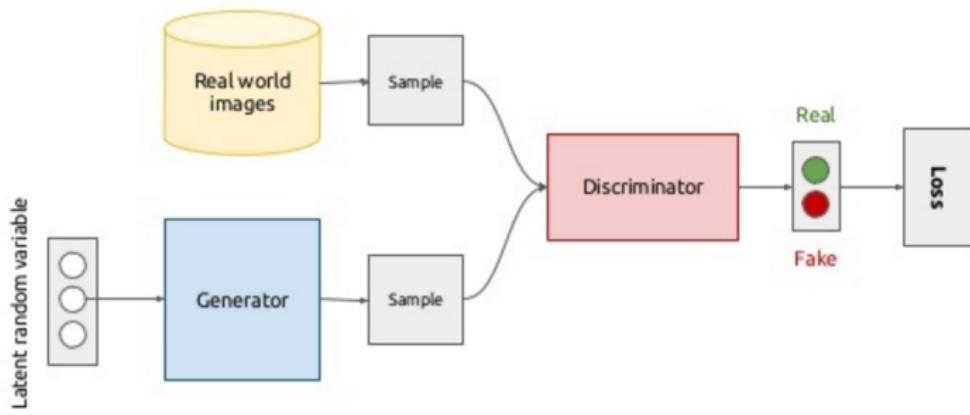
Оригинальная статья

Статья от Ian Goodfellow 2014 года, с которой начались GAN'ы:

<https://arxiv.org/pdf/1406.2661.pdf>

Обмани меня

По-русски GAN = "Генеративная состязательная сеть". Состязание состоит в том, что генератор (G) хочет сгенерировать из случайного шума (набора из случайных чисел) как можно более правдоподобную картинку, а дискриминатор (D) хочет как можно лучше отличать подделку (сгенерированную генератором) от истинных данных (и подделка, и истинные данные подаются дискриминатору для обучения).



План

1 Что нужно знать

- Воодушевляющий пример
- Идея GAN
- Формальная постановка задачи

2 Алгоритм

- Псевдокод
- Реализация на Python

3 Применения

- Картинки
- Видео
- Тексты
- Аудио

Математическая постановка

Опишем ситуацию на техническом языке. У нас есть две нейросети - генератор (G) и дискриминатор (D), и мы хотим обучать их стандартным методом backpropagation. Для этого нам нужен Loss (функция потерь). Авторы оригинальной статьи её любезно предоставляют:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))].$$

То есть градиент дискриминатора в нашей задаче будет идти с плюсом, потому что по D мы максимизируем, а вот по G - минимизируем, поэтому по G будет как обычно: градиент с минусом (антиградиент).

План

1 Что нужно знать

- Вдохновляющий пример
- Идея GAN
- Формальная постановка задачи

2 Алгоритм

- Псевдокод
- Реализация на Python

3 Применения

- Картинки
- Видео
- Тексты
- Аудио

Псевдокод из оригинальной статьи

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D(\mathbf{x}^{(i)}) + \log (1 - D(G(\mathbf{z}^{(i)}))) \right].$$

end for

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(\mathbf{z}^{(i)}))).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

План

1 Что нужно знать

- Вдохновляющий пример
- Идея GAN
- Формальная постановка задачи

2 Алгоритм

- Псевдокод
- Реализация на Python

3 Применения

- Картинки
- Видео
- Тексты
- Аудио

Код на Python

Картишка сюда не влезла, поэтому ссылки на реализации GAN'ов:

- Простой GAN на PyTorch
- Много типов GAN'ов на PyTorch и Tensorflow

Важные моменты

- Генератор (G) не видит настоящих данных, он меняет свои веса путём влияния на него дискриминатора (D)
- На вход генератору (G) каждую итерацию подаётся случайный шум (из заранее выбранного распределения)
- Можно рассматривать тренировку G и D как тренировку одной двухкомпонентной нейросети, у которой разные компоненты меняют обновляют веса по-разному

Немного фактов

По GAN'ам 300+ публикаций (за 4 года) и 55+ открытых github-репозиториев. Информация отсюда:

<https://github.com/GKalliatakis/Delving-deep-into-GANs>

На следующих слайдах мы рассмотрим много примеров использования GAN'ов. В каждом случае это несколько модифицированная версия, она отличается либо архитектурой самой сети, либо видом оптимизируемой функции потерь (loss'a), либо и тем, и тем. Для каждого случая приведена оригинальная статья и Hithub-репозиторий с реализацией.

План

1 Что нужно знать

- Вдохновляющий пример
- Идея GAN
- Формальная постановка задачи

2 Алгоритм

- Псевдокод
- Реализация на Python

3 Применения

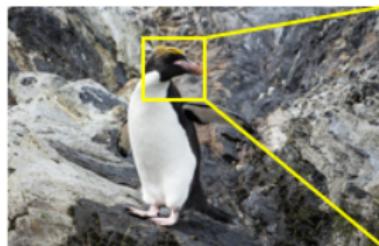
- Картинки
- Видео
- Тексты
- Аудио

Super Resolution (SRGAN)

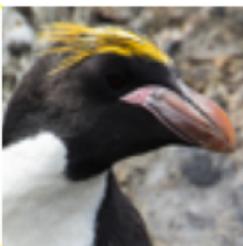
Кейс: улучшение качества (разрешения) изображения

Статья: "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network"

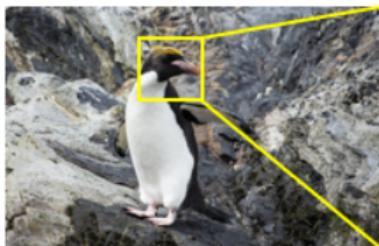
Код: <https://github.com/tensorlayer/srgan> (TensorFlow)



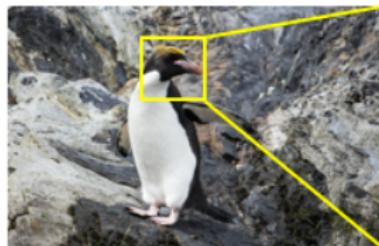
LG



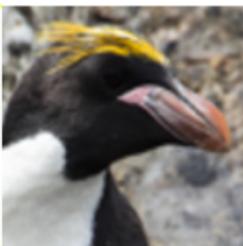
HR (GT)



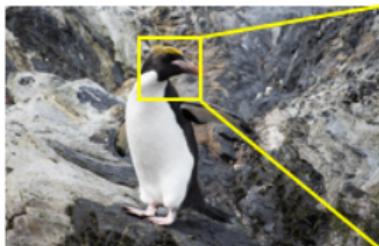
LG



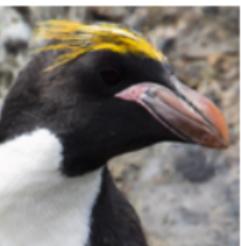
Bicubic



Generative Adversarial Networks

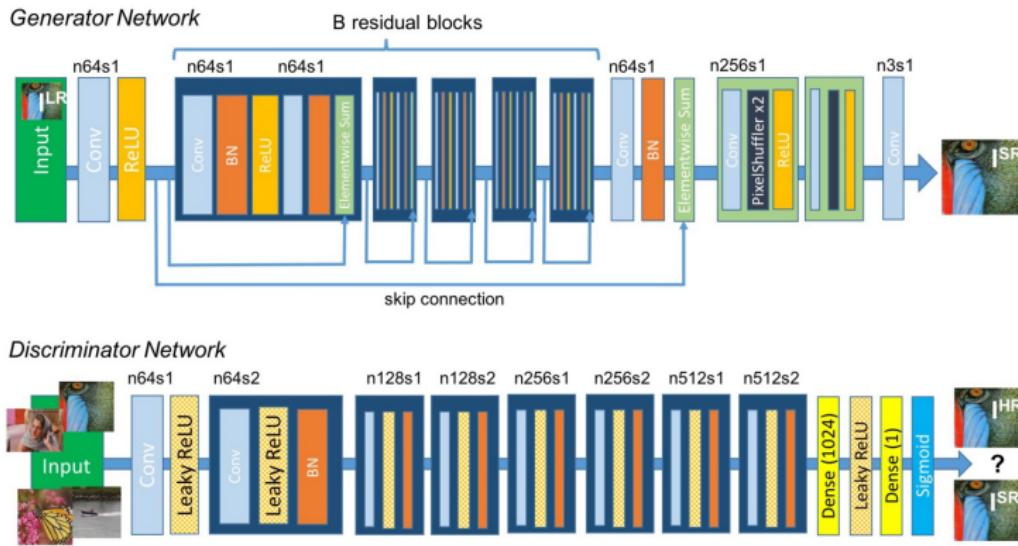


SRGAN



Super Resolution (SRGAN)

В статье всё хорошо расписано. Ключевой момент в этом решении - perceptual loss, определённый авторами как сумма двух loss'ов с грамотно подобранными весами, так что сетка учится именно улучшать изображение.

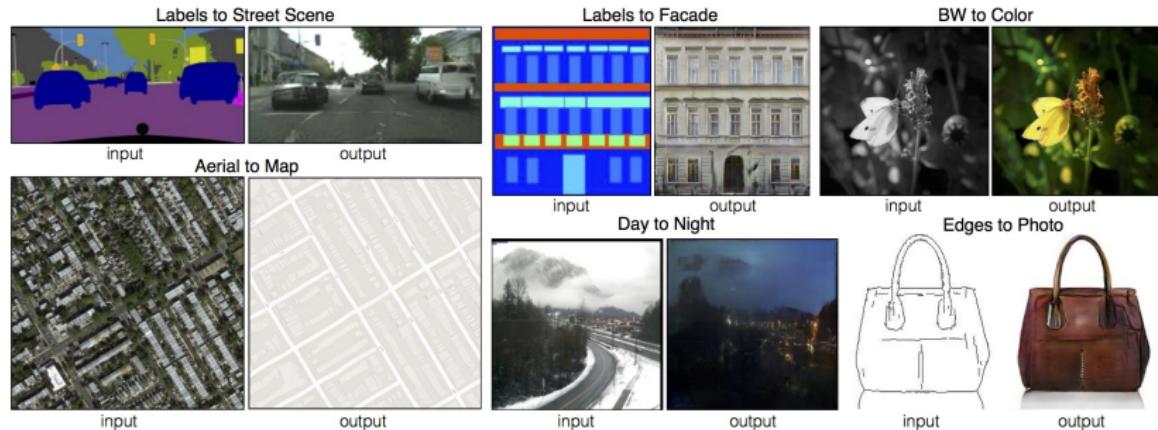


Перенос картинки в картинку (Conditional GAN)

Кейс: генерация по картинке другой картинки с изученным сеткой стилем и текстурой (pix2pix mapping)

Статья: "Image-to-Image Translation with Conditional Adversarial Networks"

Код: <https://github.com/phillipi/pix2pix> (Torch)



Перенос картинки в картинку (Conditional GAN)

Фишка этой статьи в том, что они сконструировали (снова) свой loss, который, по утверждению авторов, имеет некоторые универсальные для выучивания любого mapping'a свойства.

Основа для архитектуры - Conditional GAN. В качестве генератора (G) используется U-Net (хороша в задаче сегментации изображения).

Также авторы прокачали архитектуру, введя некий PatchGAN.

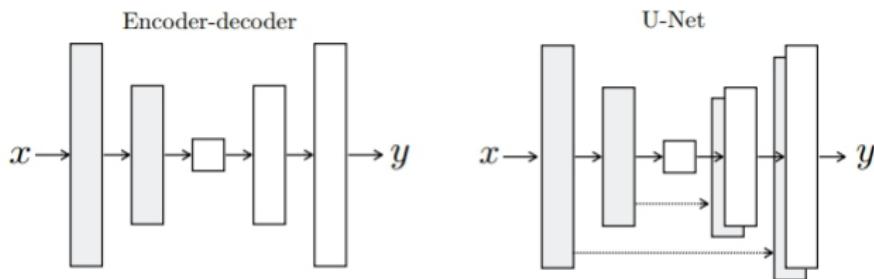


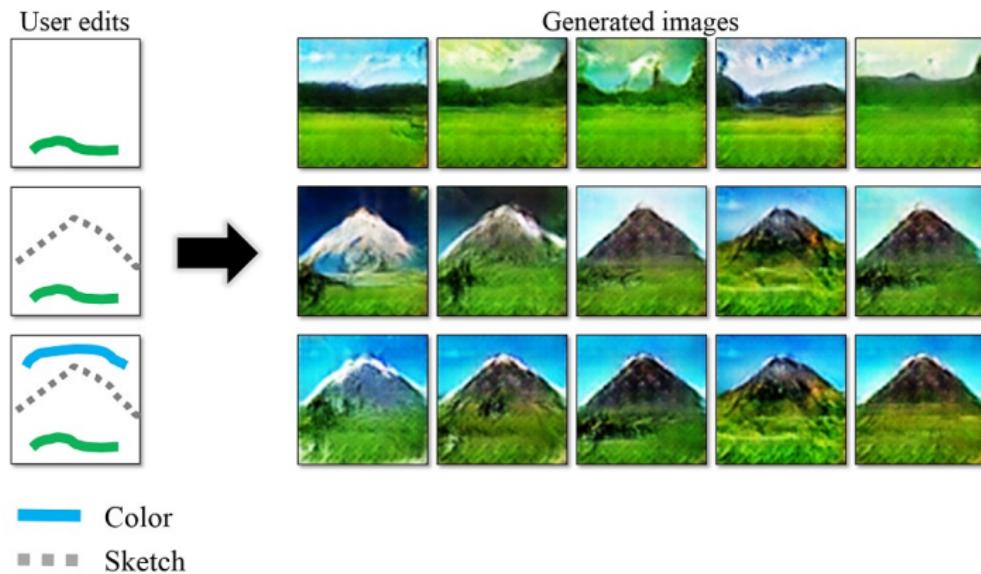
Figure 3: Two choices for the architecture of the generator. The “U-Net” [49] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

Интерактивная генерация по наброску (iGAN)

Кейс: интерактивная генерация изображения с заданными элементами

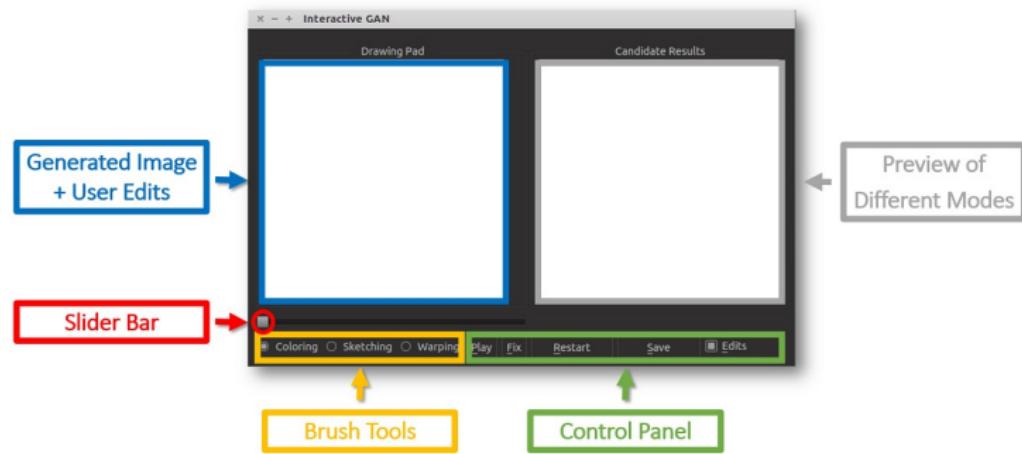
Статья: "Generative Visual Manipulation on the Natural Image Manifold"

Код: <https://github.com/junyanz/iGAN> (Theano)



Интерактивная генерация по наброску (iGAN)

Здесь фишка снова в специальном loss'e, в этот раз он контролирует "сглаженность" распределения генерируемых изображений на некотором высокоразмерном многообразии. Статья чуть сложнее к прочтению, чем предыдущие, зато можно скачать себе репозиторий и запустить интерактивное демо. Основа для архитектуры - Deep Convolutional GAN.



Генерация кода по картинке (pix2code)

Кейс: генерация кода для iOS или Web по картинке пользовательского интерфейса

Статья: "pix2code: Generating Code from a Graphical User Interface Screenshot"

Код: <https://github.com/tonybeltramelli/pix2code> (TF, Keras)



(a) iOS GUI screenshot

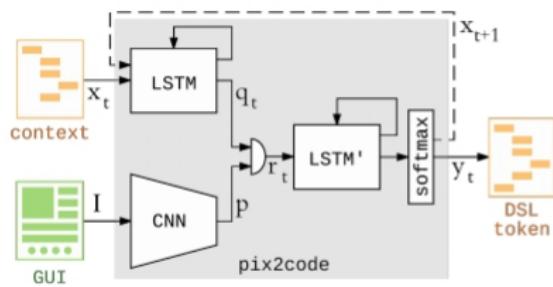
```
stack {  
    row {  
        label, switch  
    }  
    row {  
        label, btn-add  
    }  
    row {  
        label, slider, label  
    }  
    row {  
        img, label  
    }  
}  
footer {  
    btn-more, btn-contact, btn-search, btn-download  
}
```

(b) Code describing the GUI written in our DSL

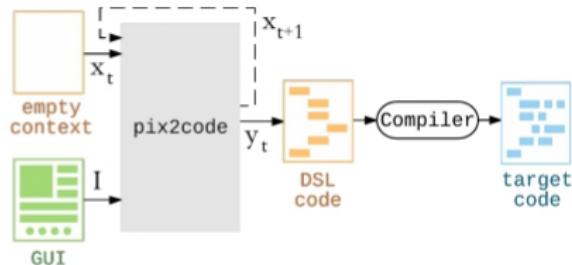
Figure 2: An example of a native iOS GUI written in our markup-like DSL.

Генерация кода по картинке (pix2code)

Здесь грамотно сочетаются CNN, RNN и GAN. То есть картинка интерфейса подаётся на вход, с помощью CNN извлекаются фичи, с помощью LSTM делается image captioning, а уже по тому описанию генерируют GAN'ом код.



(a) Training



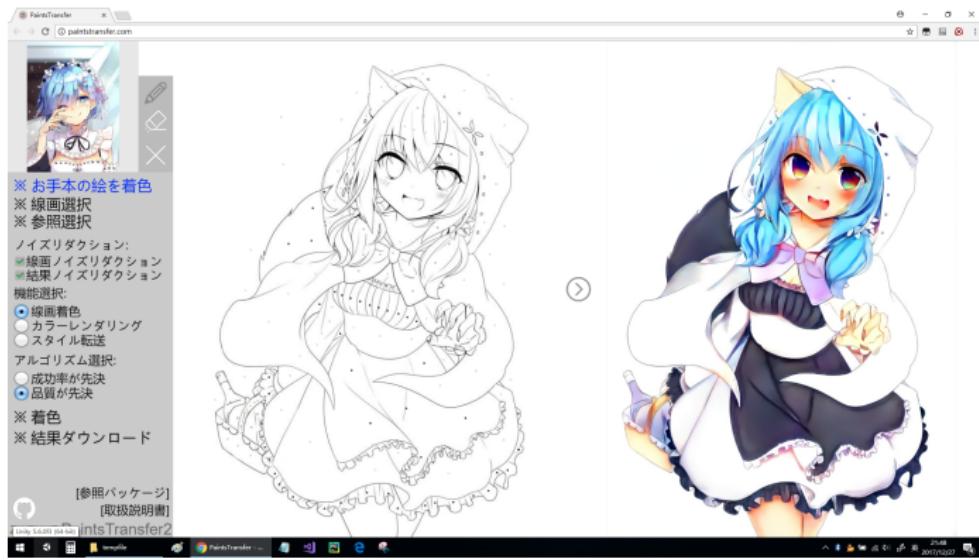
(b) Sampling

Раскрашивание рисунка (style2paint)

Кейс: раскрашивание графического рисунка в соответствие с поданной картинкой стиля

Статья: не нашёл :(

Код: <https://github.com/llyyasviel/style2paints> (TF, Keras)

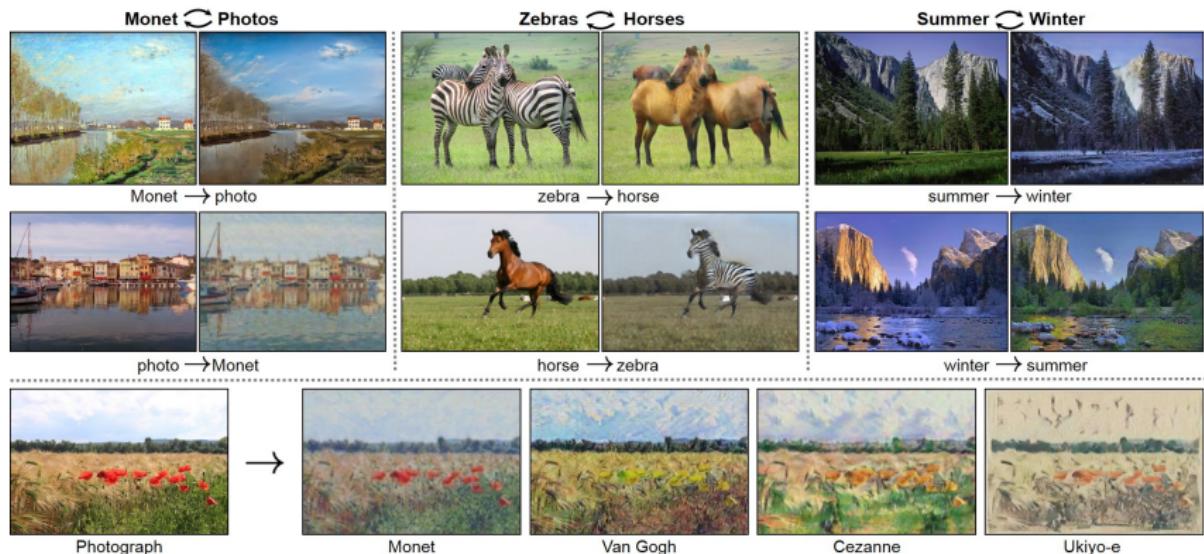


Перенос объекта в объект (CycleGAN)

Кейс: перенос объекта в объект на картинке, перенос стиля, эффективный mapping

Статья: "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks"

Код: <https://github.com/junyanz/CycleGAN> (Torch)



Перенос объекта в объект (CycleGAN)

Ключевой момент - специальный cyclic consistency loss. Подробности в статье :)

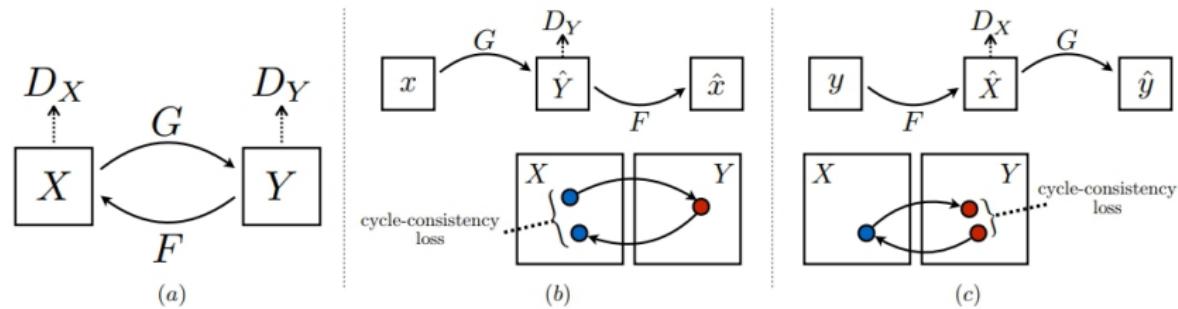


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X . D_Y encourages G to translate X into outputs indistinguishable from domain Y , and vice versa for D_X and F . To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

Генерация картинки по описанию (StackGAN)

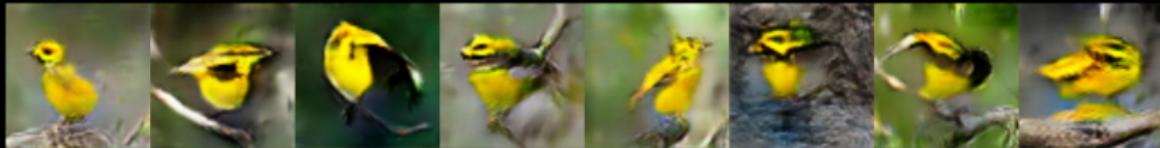
Кейс: генерация картинки хорошего качества по текстовому описанию

Статья: "StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks"

Код: <https://github.com/hanzhanggit/StackGAN> (TF, Torch)

A small yellow bird with a black crown and a short black pointed beak

Stage-I

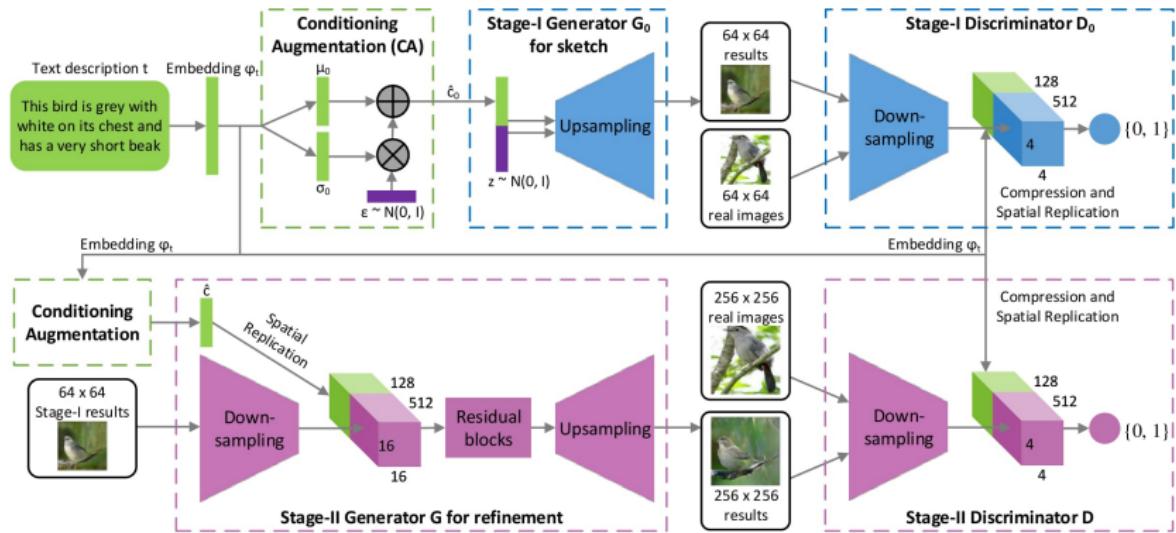


Stage-II



Генерация картинки по описанию (StackGAN)

Основная фишка - GAN после GAN'a (Stacked GAN), чтобы картинка была хорошего качества (разрешения). Ну и, как всегда, собственный loss для эффективного обучения.



Новый тип GAN'ов

Кейс: генерация картинки очень высокого качества

Статья: "Progressive growing of GAN's for improved quality, stability, and variations"

Код: https://github.com/tkarras/progressive_growing_of_gans
(TensorFlow)



Да, это сгенерировано сеткой. Ребята из NVIDIA знатно прошарились.

План

1 Что нужно знать

- Вдохновляющий пример
- Идея GAN
- Формальная постановка задачи

2 Алгоритм

- Псевдокод
- Реализация на Python

3 Применения

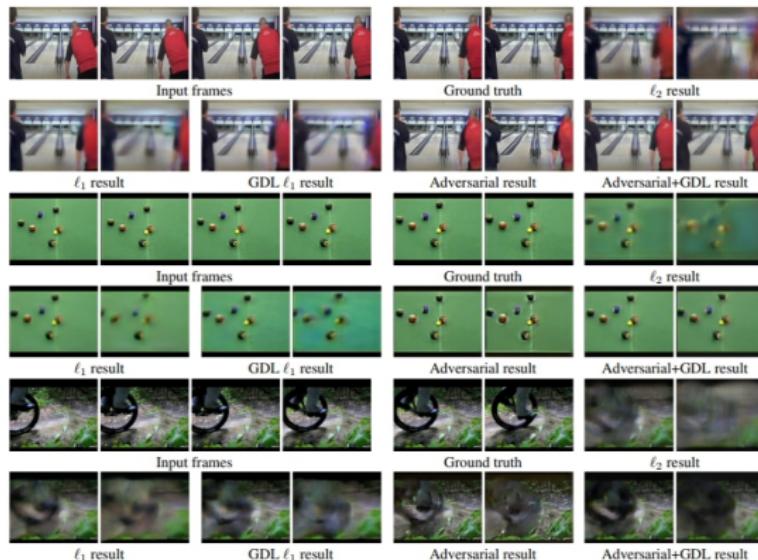
- Картинки
- Видео**
- Тексты
- Аудио

Генерация следующего кадра видео

Кейс: предсказание (генерация) следующего кадра видео по предыдущим

Статья: "Deep multi-scale video prediction beyond mean square error"

Код: https://github.com/dyelax/Adversarial_Video_Generation
(TensorFlow)



План

1 Что нужно знать

- Воодушевляющий пример
- Идея GAN
- Формальная постановка задачи

2 Алгоритм

- Псевдокод
- Реализация на Python

3 Применения

- Картинки
- Видео
- Тексты**
- Аудио

Генерация текстов - Neurona

Кейс: Яндекс написал свою сетку, способную генерировать текст для песен, похожий на тексты Курта Кобейна (солиста группы Nirvana)

Статья: не нашёл :(

Код: в Яндексе

The screenshot shows the Yandex Music website. At the top, there is a search bar with the placeholder "Трек, альбом, исполнитель" and a magnifying glass icon. Below the search bar are navigation links: Главное (Main), Рекомендации (Recommendations), Жанры (Genres), and Радио (Radio). The main content area features a large circular profile picture of a person singing into a microphone. To the right of the profile picture, the text "ИСПОЛНИТЕЛЬ" (Performer) is written above the name "Neurona" in a large, bold font. Below the name, the word "альтернатива" (Alternative) is written. Underneath the performer information, there are three buttons: a yellow "▶ Слушать" (Listen) button, a white button with a heart icon and the number "172", and a white button with a share icon. Below these buttons is a red circular icon with a white zero symbol. At the bottom of the main content area, there is a horizontal menu with three tabs: "ГЛАВНОЕ" (Main) which is underlined in red, "ТРЕКИ" (Tracks), and "АЛЬБОМЫ" (Albums). Below this menu, the section "Популярные треки" (Popular tracks) is displayed, showing four tracks with their names and durations: "Cadillac" (2:06), "I Don't Wanna See You" (3:41), "In the Back of Your Glass" (2:56), and "Thing of Yours" (2:37). Each track has a small thumbnail image next to its name.

Генерация картинки по описанию (AttentionGAN)

Кейс: генерация картинки хорошего качества по текстовому описанию

Статья: "AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks"

Код: <https://github.com/taoxugit/AttnGAN> (PyTorch)



План

1 Что нужно знать

- Вдохновляющий пример
- Идея GAN
- Формальная постановка задачи

2 Алгоритм

- Псевдокод
- Реализация на Python

3 Применения

- Картинки
- Видео
- Тексты
- Аудио

Генерация музыки

Можно генерировать музыку, однако конкретных статей найдено не было, только новости в Google.

Ещё немного ссылок

- Модуль для GAN'ов в Keras:
<https://github.com/bstriner/keras-adversarial>
- Похожий на iGAN: <https://github.com/ajbrock/Neural-Photo-Editor>
- Ещё больше примеров - здесь:
<https://github.com/GKalliatakis/Delving-deep-into-GANs>
- Про GAN'ы от OpenAI:
<https://blog.openai.com/generative-models/>
- Дельная статья на Хабре: <https://habrahabr.ru/post/352794/>