Generative adversarial network

Сомов Иван Сергеевич

МГУ имени М. В. Ломоносова, факультет ВМК, кафедра ММП

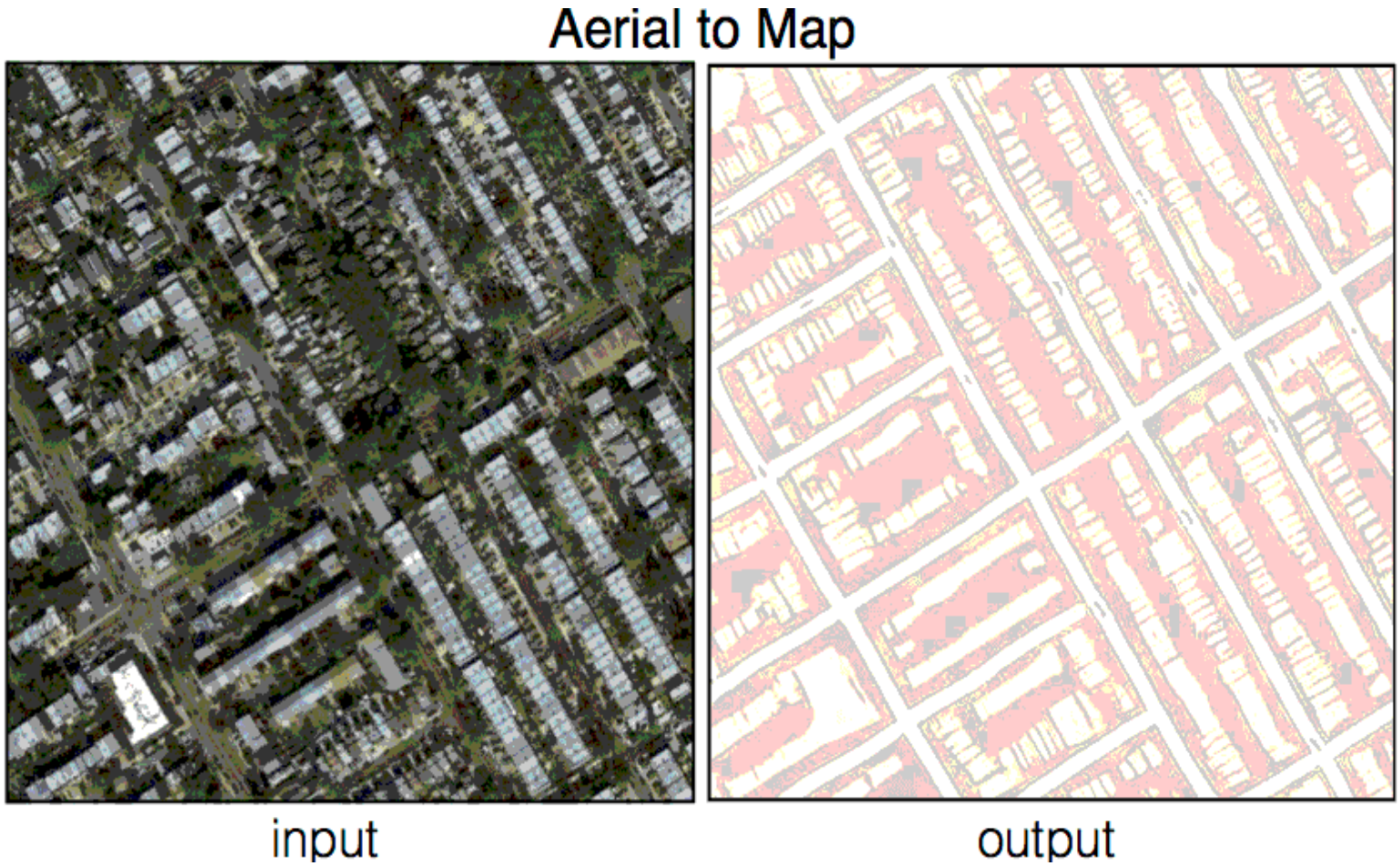
22 марта 2018 г.

Мотивация

- генеративная модель => можно извлечь пользу из неразмеченных данных
- некоторые задачи требуют генерации реалистичной данных
- некоторые задачи требуют генерацию сэмплов:
  - single image super resolution
  - image-to-image translation
- может быть настроена для обучения с подкреплением

Задачи

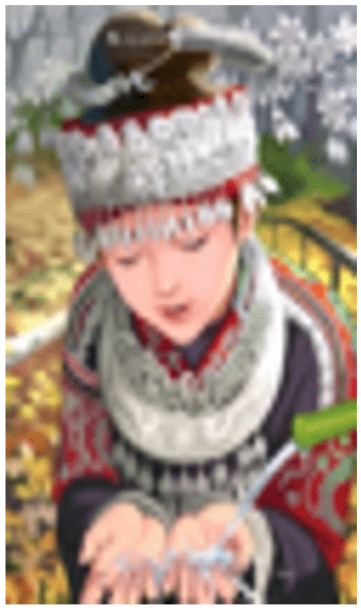
Image-to-image translation



Задачи

Single image super resolution

bicubic
(21.59dB/0.6423)



SRResNet
(23.53dB/0.7832)

SRGAN
(21.15dB/0.6868)

original

Генеративные и дискриминативные модели

Генеративная модель

- Explicit density models: пытаемся $p(x, y)$
- Implicit density models: строим функцию $(x, y) = g(z)$, где z – скрытая переменная

Дискриминативная модель

- Probabilistic discriminative: пытаемся оценить условную плотность $p(y|x)$
- Ordinary discriminative: строим $y = f(x)$

Постановка задачи

Имеется:

- набор данных $X = \{x\}_{i=0}^{n}$, $x \sim p_{data}(x)$
- скрытые переменные z из некоторого распределения $p_z(z)$
- дифференцируемая функция $x = G(z, \theta_G)$: $G(z, \theta_G) \sim p_{model}(x)$

Требуется найти такой параметр θ_G, что $p_{model}(x) = p_{data}(x)$

GAN

Решение:

- пусть $G(z, \theta_g)$ и $D(x, \theta_D)$ – дифференцируемые функции

GAN

Решение:

- пусть $G(z, \theta_g)$ и $D(x, \theta_D)$ – дифференцируемые функции
- G – генератор; D – дискриминатор

GAN

Решение:

- пусть $G(z, \theta_g)$ и $D(x, \theta_D)$ – дифференцируемые функции
- G – генератор; D – дискриминатор
- пусть $D(x, \theta_D) = P(\{x \sim p_{data}(x)\}) = 1 - P(\{x \sim p_{model}(x)\})$

GAN

Решение:

- пусть $G(z, \theta_g)$ и $D(x, \theta_D)$ – дифференцируемые функции
- G – генератор; D – дискриминатор
- пусть $D(x, \theta_D) = P(\{x \sim p_{data}(x)\}) = 1 - P(\{x \sim p_{model}(x)\})$
- цель дискриминатора – минимизировать некоторый функционал $J^D(\theta_D, \theta_G)$
- цель генератора – минимизировать некоторый функционал $J^G(\theta_D, \theta_G)$
- $J^G = -J^D$

Процесс обучения

- Функция потерь:
$J^D(\theta_D, \theta_G) = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_z \log\left(1 - D(G(z))\right)$

Процесс обучения

- Функция потерь:
  $J^D(\theta_D, \theta_G) = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_z \log\left(1 - D(G(z))\right)$

- Log loss:
  $J = \mathbb{E}_{x\sim p(x,y)} y \log \frac{1}{1+\exp M[x]} + \mathbb{E}_{x\sim p(x,y)}(y-1)\log\left(1 - \frac{1}{1+\exp M[x]}\right)$

Процесс обучения

- Функция потерь:
  $J^D(\theta_D, \theta_G) = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_z \log\left(1 - D(G(z))\right)$

- Log loss:
  $J = \mathbb{E}_{x\sim p(x,y)} y \log \frac{1}{1+\exp M[x]} + \mathbb{E}_{x\sim p(x,y)}(y-1)\log\left(1 - \frac{1}{1+\exp M[x]}\right)$

- $y = I[x \sim p_{model}(x)]$ – целевая переменная для дискриминатора

Процесс обучения

Алгоритм:

for number of training iterations **do**
  for k steps **do**
    Sample minibatch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from $p_z(z)$
    Sample minibatch of m examples $\{x^{(1)}, ..., x^{(m)}\}$ from X
    Update discriminator's parameter θ_D by ascending its stochastic gradient: $\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log (1 - D(G(z^{(i)})))]$
  Sample minibatch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from $p_z(z)$
  Update generator's parameter θ_G by descending its stochastic gradient: $\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} [\log (1 - D(G(z^{(i)})))]$

Обоснованность метода

- с точки зрения теории игр существует равновесие Нэша: $D(x) = \frac{1}{2}$ всюду; $p_{model}(x) = p_{data}(x)$

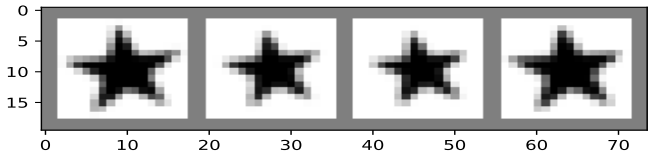
[1] https://arxiv.org/pdf/1406.2661.pdf

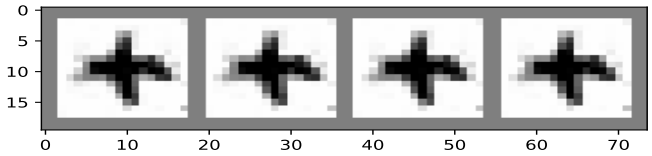
Обоснованность метода

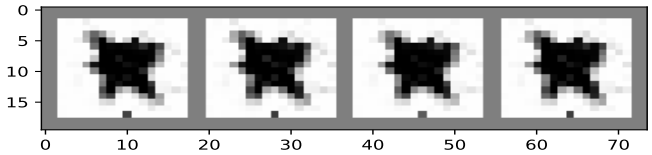
- с точки зрения теории игр существует равновесие Нэша: $D(x) = \frac{1}{2}$ всюду; $p_{model}(x) = p_{data}(x)$
- можно показать, что при фиксированном G оптимальный дискриминатор $D^*_G = \frac{p_{data}}{p_{data}+p_{model}}$
- можно показать, что в результате работы алгоритма p_{model} сходится к p_{data}[1]

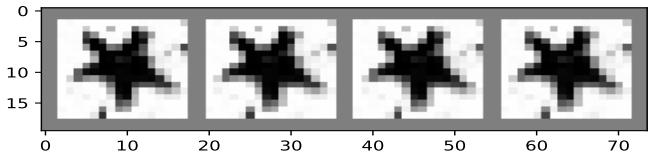
[1] https://arxiv.org/pdf/1406.2661.pdf

Датасет фигур









Выводы

Плюсы:

- много реализаций
- на данный момент обеспечивает наиболее реалистичную генерацию данных
- быстрая генерация сэмплов

Минусы:

- в базовой модели нельзя оценить плотность непосредственно
- требует много данных для обучения
- может проваливаться в локальные оптимумы, когда генератору выгодно дублировать один и тот же сэмпл