Глубинное обучение
Лекция 9: Оптимизация против нейросетей или Adversarial X

Лектор: Антон Осокин

ФКН ВШЭ, 2021

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

План лекции

- Нераспознаваемые примеры (adversarial examples)
  - Примеры, причины, следствия
  - Способы построения и борьбы

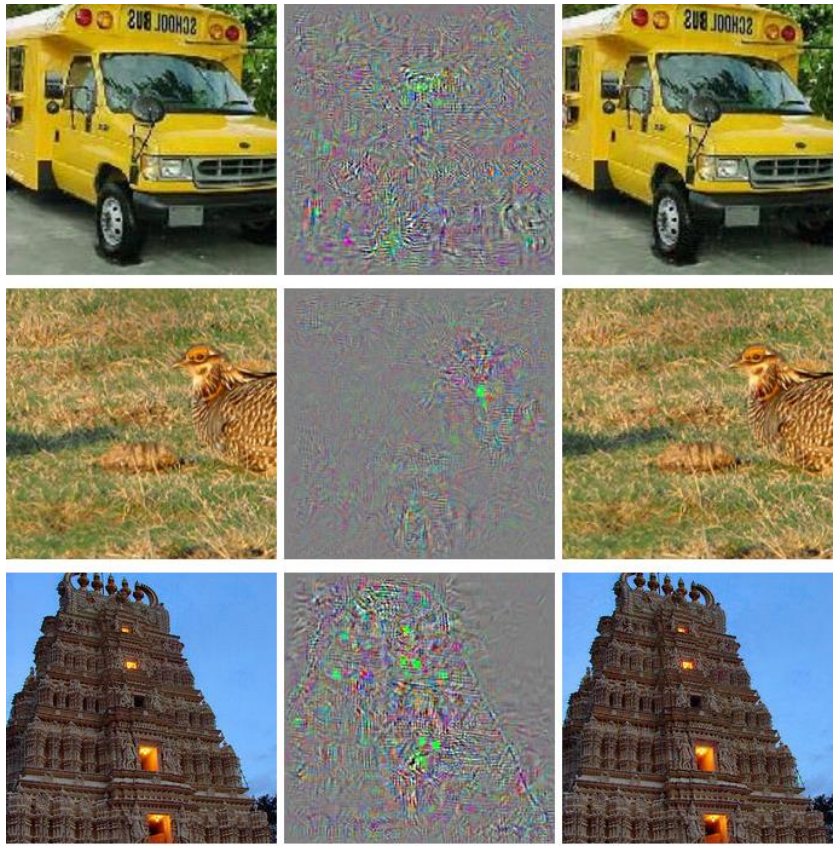
- Адаптация к данным (domain adaptation)
  - Что это? Зачем?
  - Простейшие методы

- Нейросети как функции потерь
  - GANs

Adversarial examples

[Szegedy et al., 2013]

- Невидимые глазу изменения меняют результат сети!



правильная классификация | шум | «ostrich»

Adversarial examples for text!

[Liang et al., 2017; Ebrahimi et al. 2018]

- Изменения текста (дискретные) меняют результат сети!
  - Удаления и вставки текста (настоящий текст и опечатки)

The Old Harbor Reservation Parkways are three ~~*historic*~~ roads in the Old Harbor area of Boston. ***Some exhibitions of Navy aircrafts were often held here.*** They are part of the Boston parkway system designed by Frederick Law Olmsted. They include all of William J. Day Boulevard running from ***Castle*** Island to Kosciuszko Circle along Pleasure Bay and the Old Harbor shore. The part of Columbia Road from its northeastern end at Farragut Road west to Pacuska Circle (formerly called Preble Circle). Old Harbor Reservation

83.7% Building => 88.7% Means of Transportation

Adversarial атаки и защиты от них

[Szegedy et al., 2013]

- Пусть у нас есть обученная сеть (классификация ImageNet)
  - Сеть и функция потерь $J_\theta(x, l)$
  - Атака оптимизацией ($\eta = x' - x$, l' – целевая метка для атаки):

$$\min_{x'} \quad c\|\eta\| + J_\theta(x', l')$$
$$s.t. \quad x' \in [0, 1].$$

  - Быстрая атака знаком градиента (Fast Gradient Sign Method, FGSM)

$$x' = x - \epsilon sign(\nabla_x J(\theta, x, l'))$$

- Простые варианты защит:
  - Обучение на атаках, ансамбли, дистилляция сетей

Что все это значит?

- Adversarial examples – область с большими следствиями
  - Обзор: arXiv: 1712.07107 [Yuan et al., 2017]

- Атаки обобщаются! [Papernot et al., 2016]

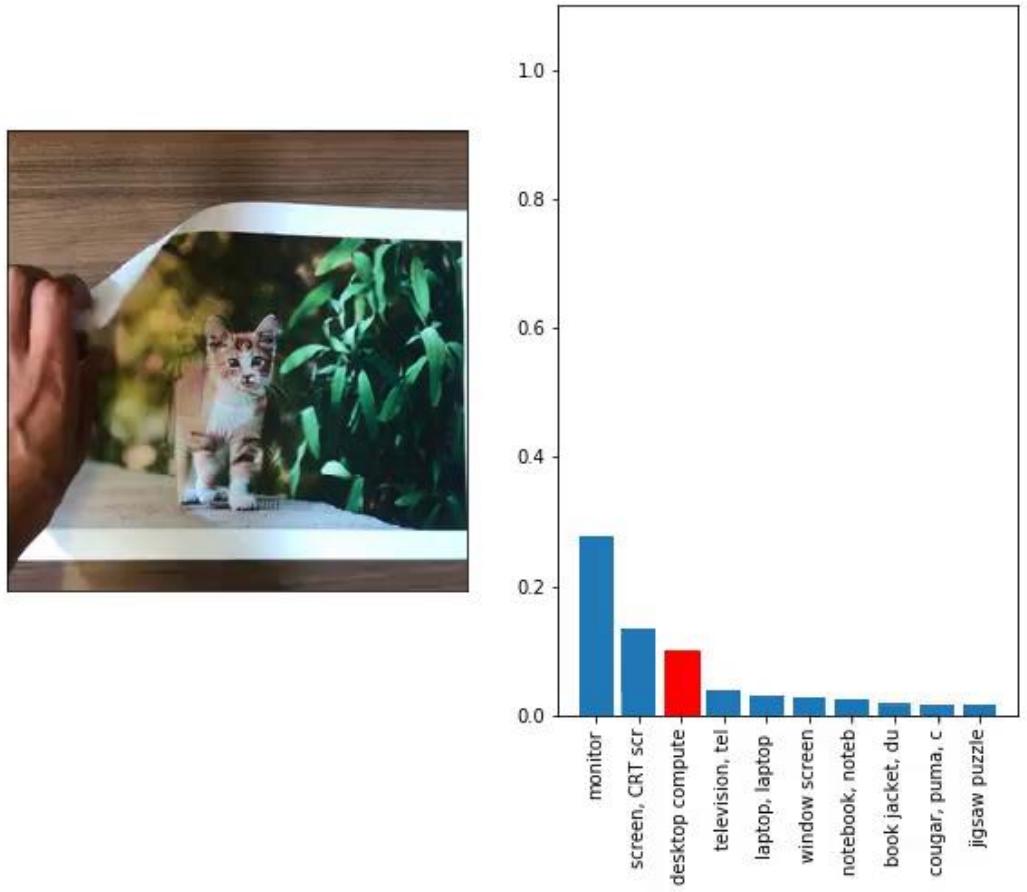
- Виды атак:
  - White-box vs. black-box
  - Targeted vs. non-targeted

- Другие алгоритмы ML тоже можно атаковать!

Adversarial example можно напечатать!

[Athalye et al., 2017]

- Можно строить устойчивые атаки!

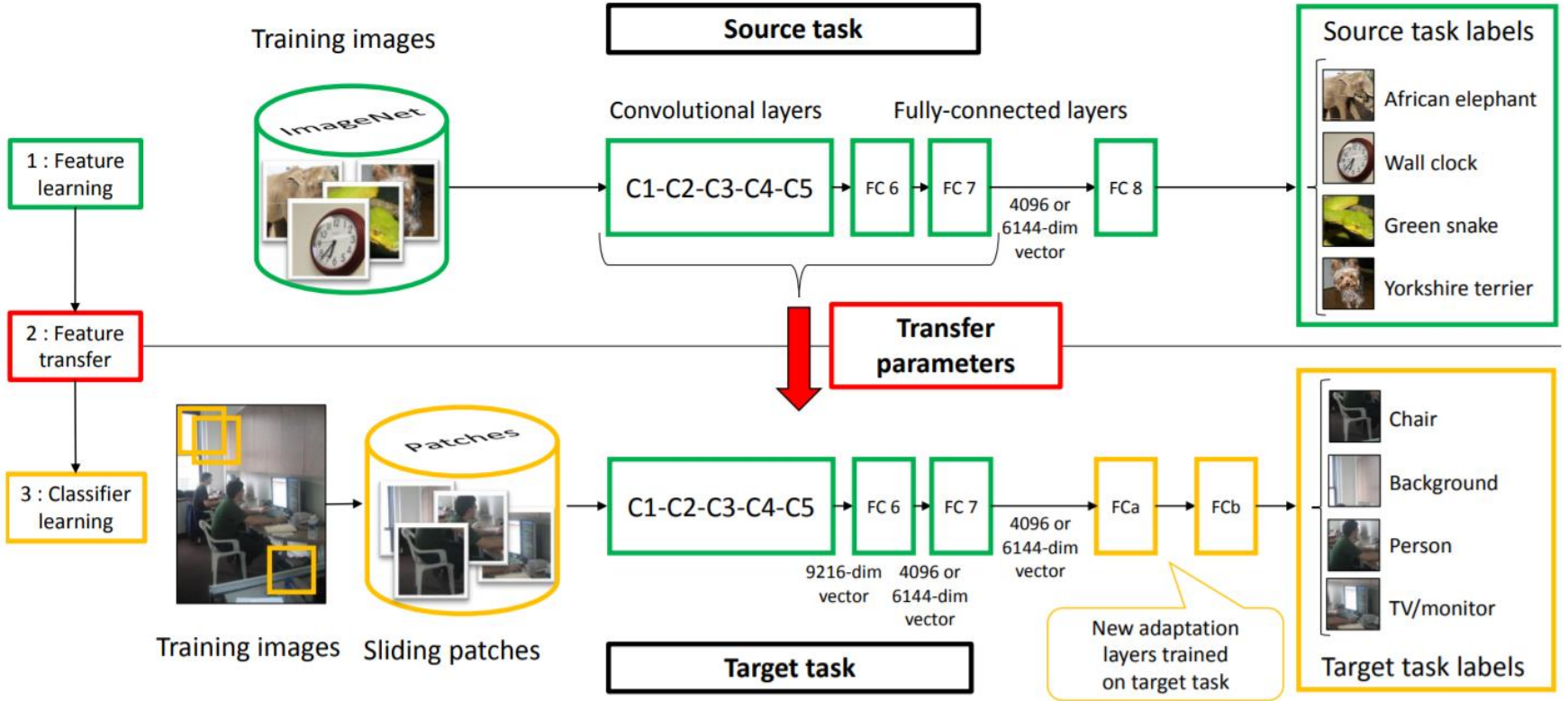


Source: https://blog.openai.com/robust-adversarial-inputs/

Адаптация к данным (domain adaptation)

- При обучении сетей часто сбор данных – узкое место
- Идея – обучиться на других данных и использовать эту сеть
- Формула CV 2013-2017:
  - Взять SOTA сеть, обученную на ImageNet
  - Откинуть голову сети, дообучиться на свою задачу

[Oquab et al., 2014]



Адаптация к данным (domain adaptation)

- При обучении сетей часто сбор данных – узкое место
- Идея – обучиться на других данных и использовать эту сеть
- Формула CV 2013-2017:
  - Взять SOTA сеть, обученную на ImageNet
  - Откинуть голову сети, дообучиться на свою задачу

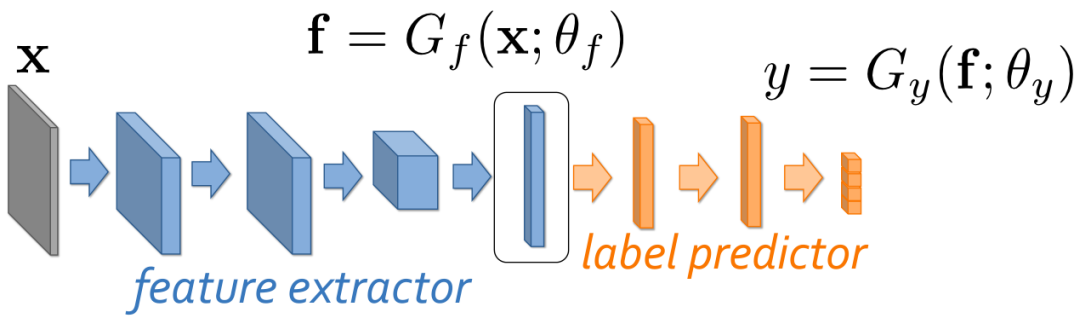
- Формула работает хуже, если данные слишком различны
  - Domain shift

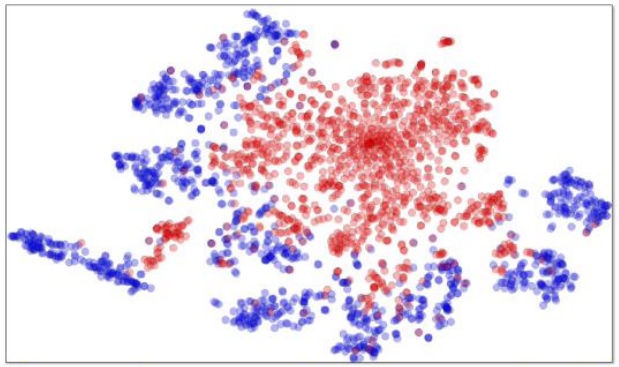
- Область domain adaptation
  - Есть много размеченных данных в source domain
  - Есть много неразмеченных данных в target domain
  - Как это использовать?

Как убрать разницу (domain shift)?

[Ganin&Lempitsky, 2014]

- Где проявляется разница?
  - На промежуточных слоях разные значения признаков





$$S(\mathbf{f}) = \{G_f(\mathbf{x}; \theta_f) \,|\, \mathbf{x} \sim S(\mathbf{x})\}$$

$$T(\mathbf{f}) = \{G_f(\mathbf{x}; \theta_f) \,|\, \mathbf{x} \sim T(\mathbf{x})\}$$

Image credit: Victor Lempitsky

Как убрать разницу (domain shift)?

[Ganin&Lempitsky, 2014]

- Где проявляется разница?
  - На промежуточных слоях разные значения признаков
  - Цель – одинаковые распределения
  - Но не сломать классификацию

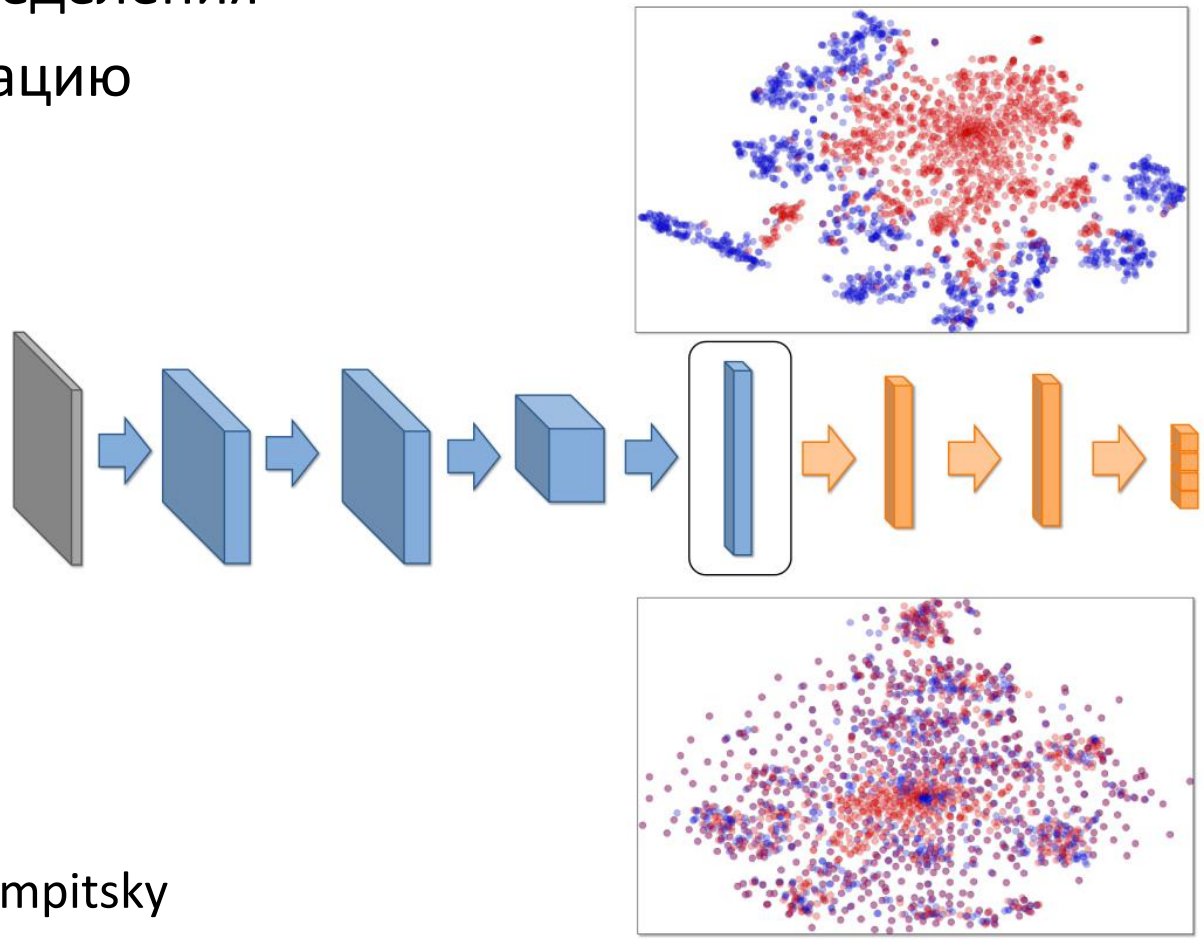
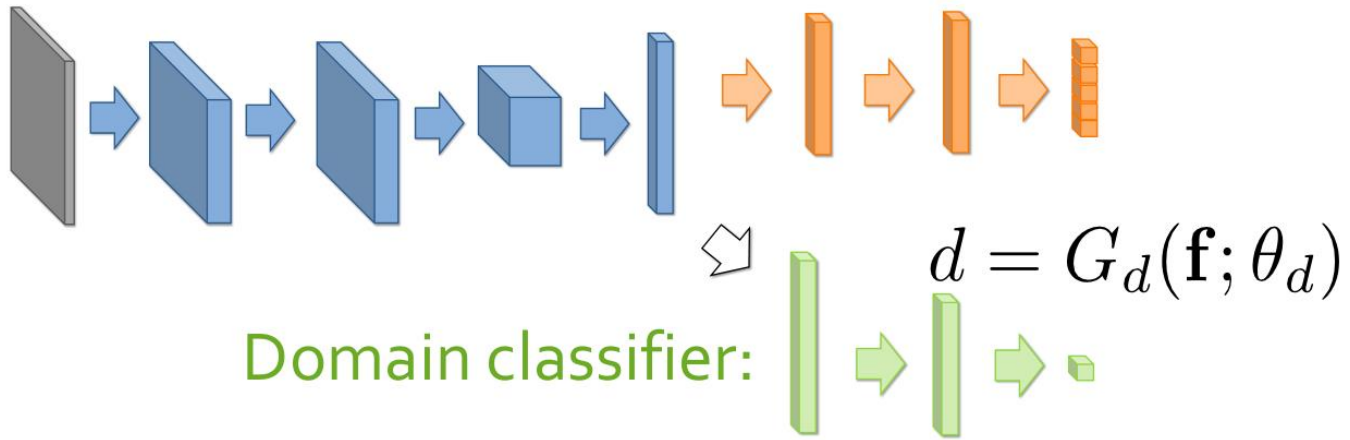


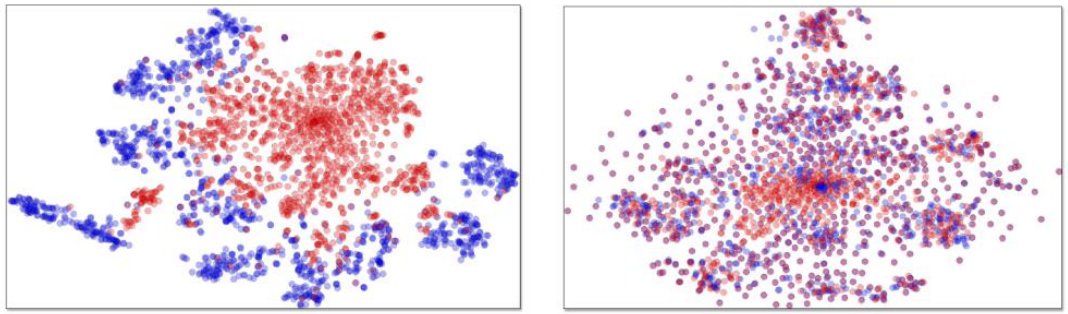
Image credit: Victor Lempitsky

Как убрать разницу (domain shift)?

[Ganin&Lempitsky, 2014]

- Идея – использовать классификатор доменов





Domain loss low Domain loss high

Image credit: Victor Lempitsky

Как убрать разницу (domain shift)?

[Ganin&Lempitsky, 2014]

- Идея – использовать классификатор доменов
- ~~Совместное обучение классификаторов?~~

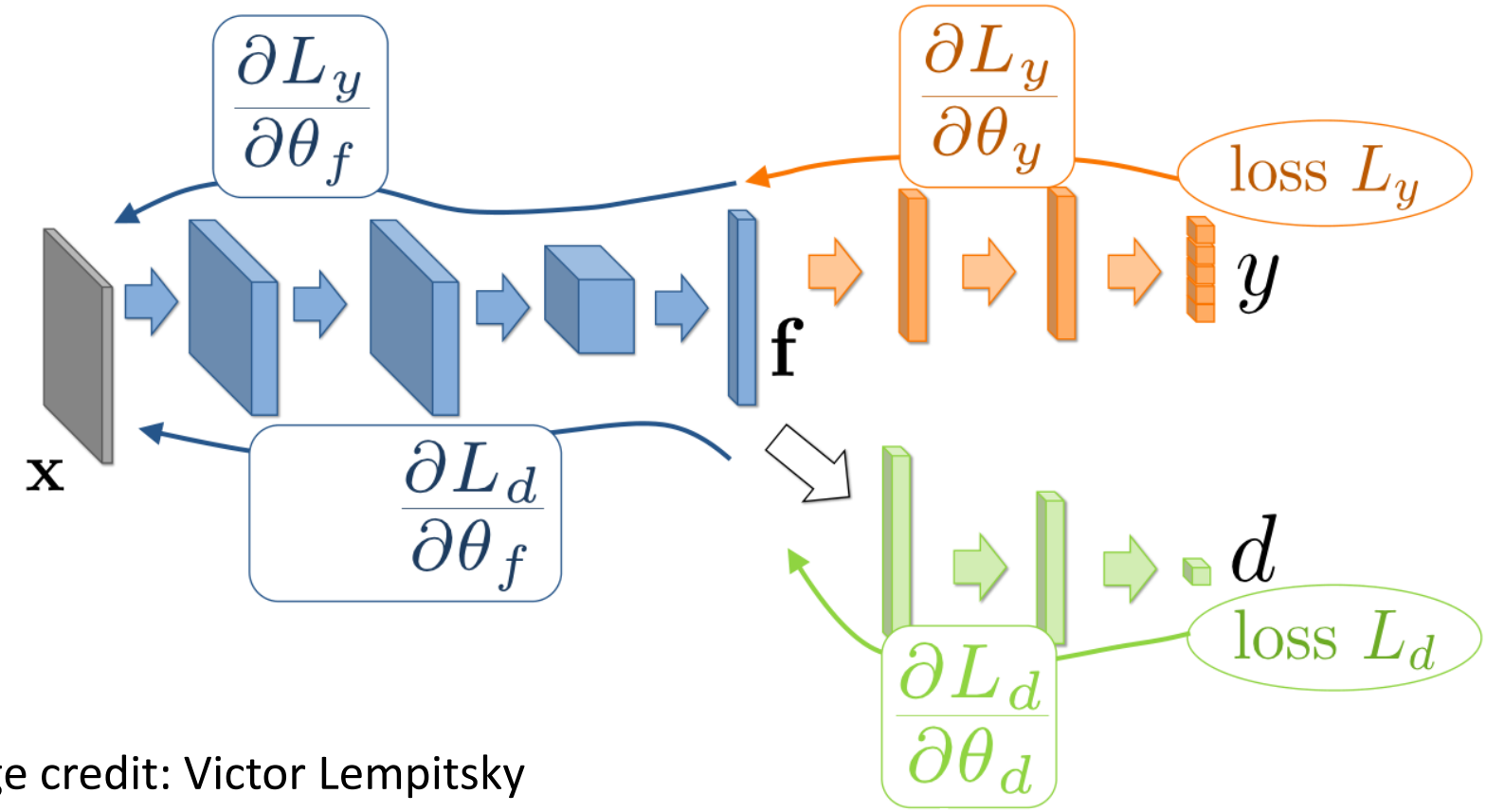


Image credit: Victor Lempitsky

Как убрать разницу (domain shift)?

[Ganin&Lempitsky, 2014]

- Идея – использовать классификатор доменов
- ~~Совместное обучение классификаторов?~~
- Нужно инвертировать градиент!

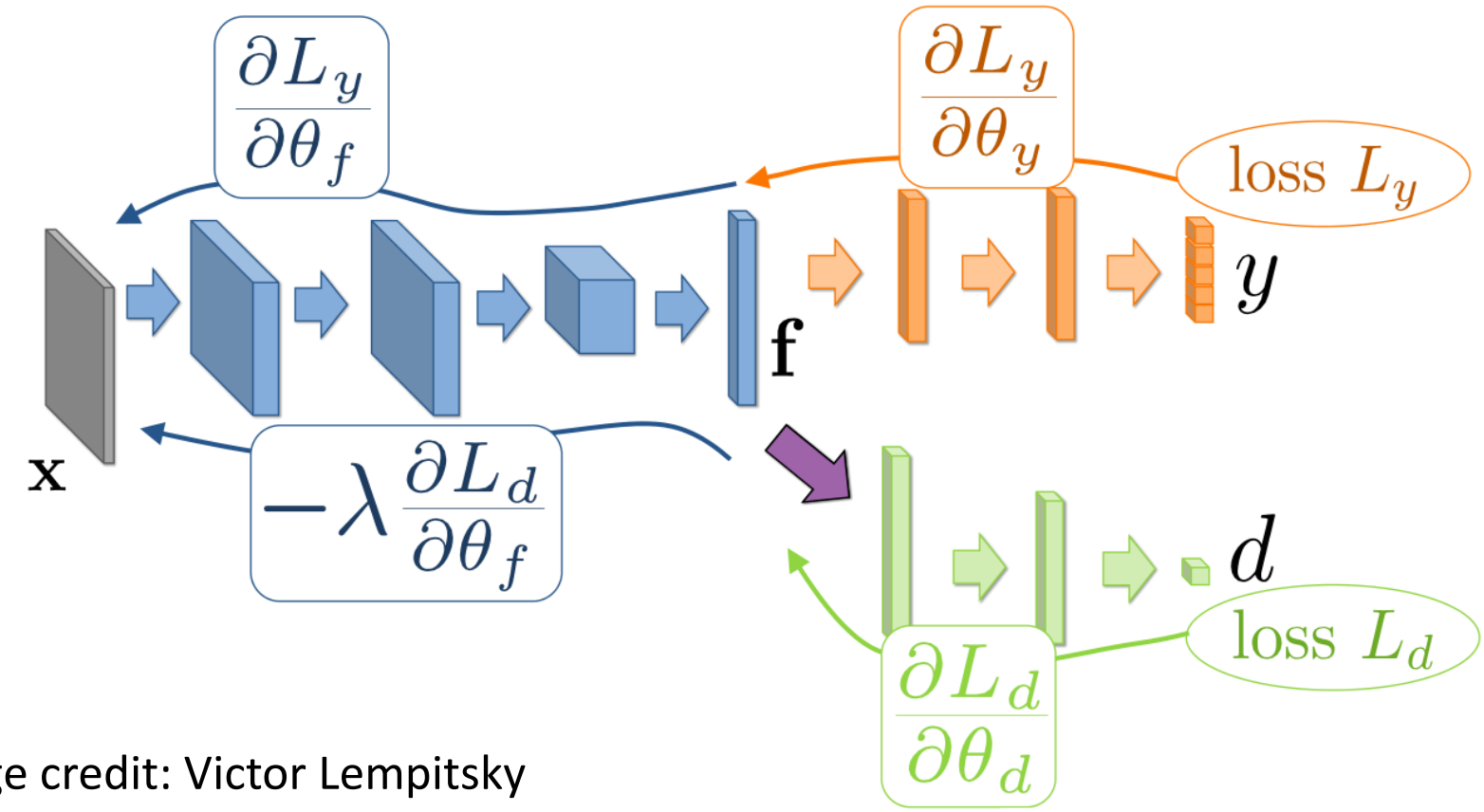


Image credit: Victor Lempitsky

Как убрать разницу (domain shift)?

[Ganin&Lempitsky, 2014]

- Идея – использовать классификатор доменов
- ~~Совместное обучение классификаторов?~~
- Нужно инвертировать градиент!
- Pytorch code:

```
class GradReverse(Function):
    def forward(self, x):
        return x.view_as(x)
    def backward(self, grad_output):
        return -lambd * grad_output
def grad_reverse(x):
    return GradReverse()(x)
```

- В обычном слое сети:

```
def forward(self, x):
    x = grad_reverse(x)
```

Как убрать разницу (domain shift)?

[Ganin&Lempitsky, 2014]

- Идея – использовать классификатор доменов
- ~~Совместное обучение классификаторов?~~
- Нужно инвертировать градиент!

- Интерпретация через седловую точку:

$$E(\theta_f, \theta_y, \theta_d) = \sum_{\substack{i=1..N \\ d_i=0}} L_y\left(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y), y_i\right) - \lambda \sum_{i=1..N} L_d\left(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d), y_i\right)$$

$$= \sum_{\substack{i=1..N \\ d_i=0}} L_y^i(\theta_f, \theta_y) - \lambda \sum_{i=1..N} L_d^i(\theta_f, \theta_d)$$

- Оптимизация:

$$(\hat{\theta}_f, \hat{\theta}_y) = \arg\min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d)$$

$$\hat{\theta}_d = \arg\max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d)$$

Generative Adversarial Networks

[Goodfellow et al., 2014]

- Генеративные модели (обычно для изображений)
- Основная идея: вместо определения целевой функции (правдоподобие, ошибка реконструкции)
  целевая функция обучается вместе с данными
- Генератор – сеть, синтезирующая картинки из шума
- Дискриминатор – сеть, отличающая настоящие от синтезированных

Generative Adversarial Networks

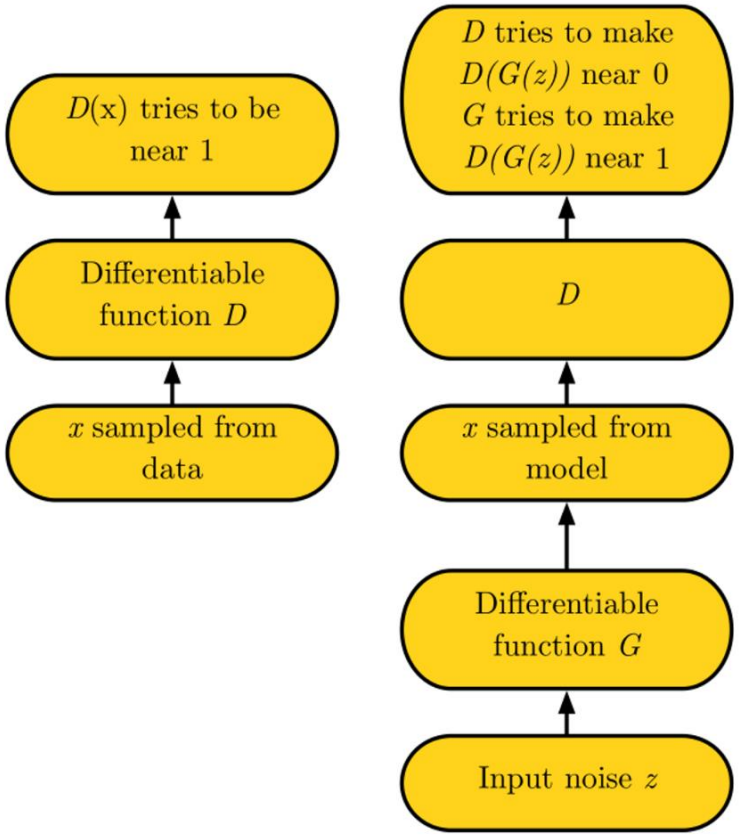
[Goodfellow et al., 2014]

Две сети:

- G – генератор – выдает изображение
- D – дискриминатор – выдает число из [0, 1]

Седловая точка:

$$\min_G \max_D \mathbb{E}_{x\sim\text{data}}[\log D(x)] + \mathbb{E}_{z\sim\text{noize}}[\log(1 - D(G(z)))]$$



Обучение дискриминатора:

$$\max_D \mathbb{E}_{x\sim\text{data}}[\log D(x)] + \mathbb{E}_{z\sim\text{noize}}[\log(1 - D(G(z)))]$$

соответствует минимизации лог лосса

Обучение генератора:

~~$$\min_G \mathbb{E}_{z\sim\text{noize}}[\log(1 - D(G(z)))]$$~~

Затухает градиент, когда плохой G (начало обучения)

$$\max_G \mathbb{E}_{z\sim\text{noize}}[\log D(G(z))]$$

Image credit: Ian Goodfellow

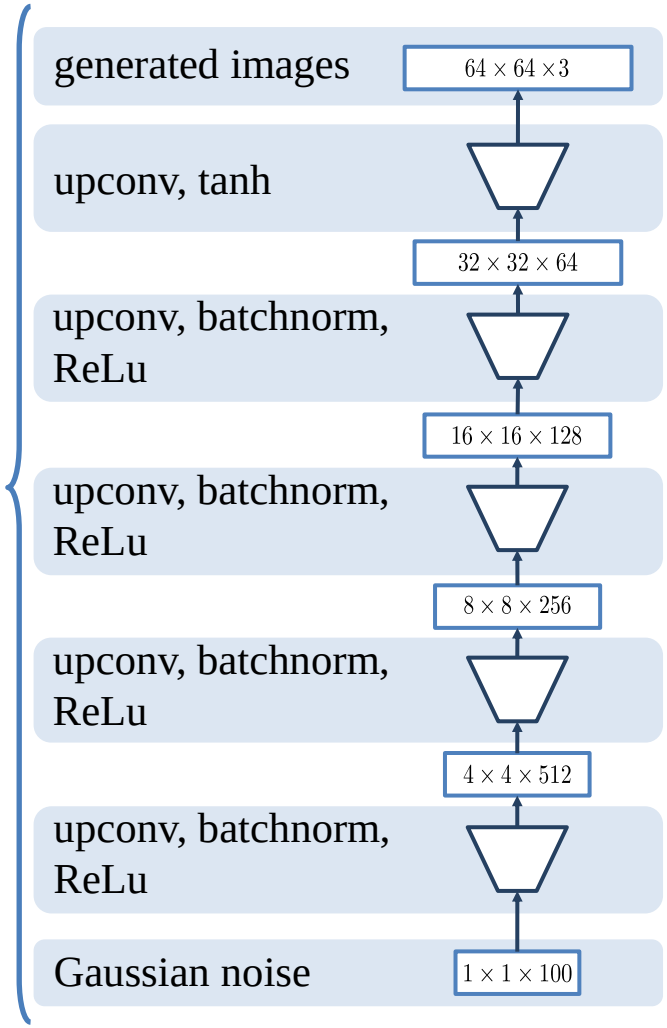
GANs работают?

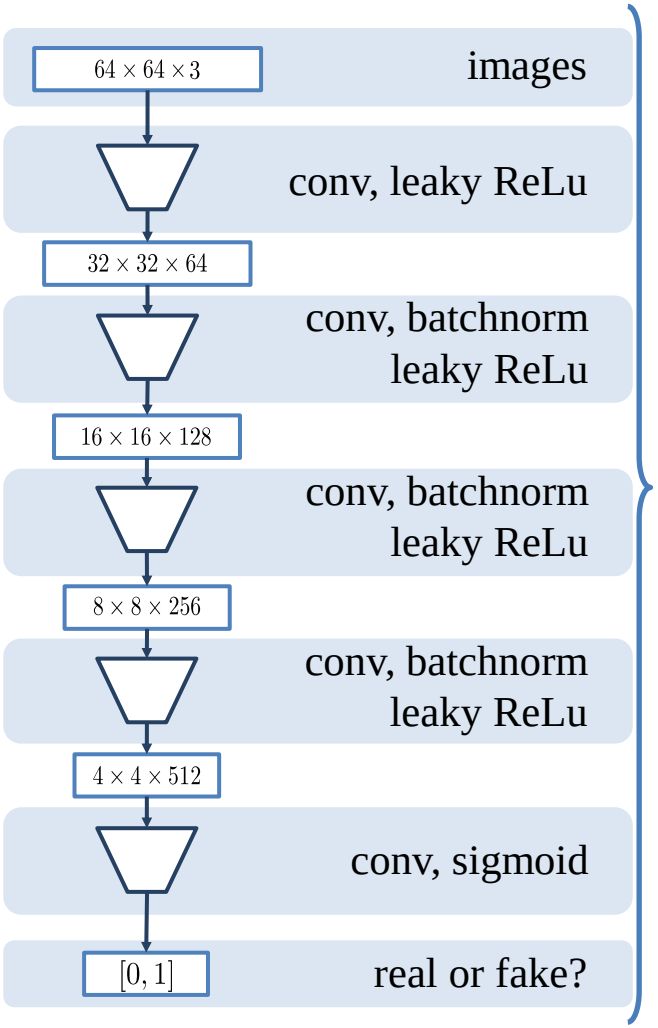
- Часто, модель сложно заставить работать
- Много тонкостей реализации – очень важны исходные коды
- Требуется очень много вычислительных ресурсов
- BigGAN [Brock et al.; 2018], StyleGANv2 [Karras et al.; 2020]

Архитектура DCGAN

[Radford et al. 2014]

генератор



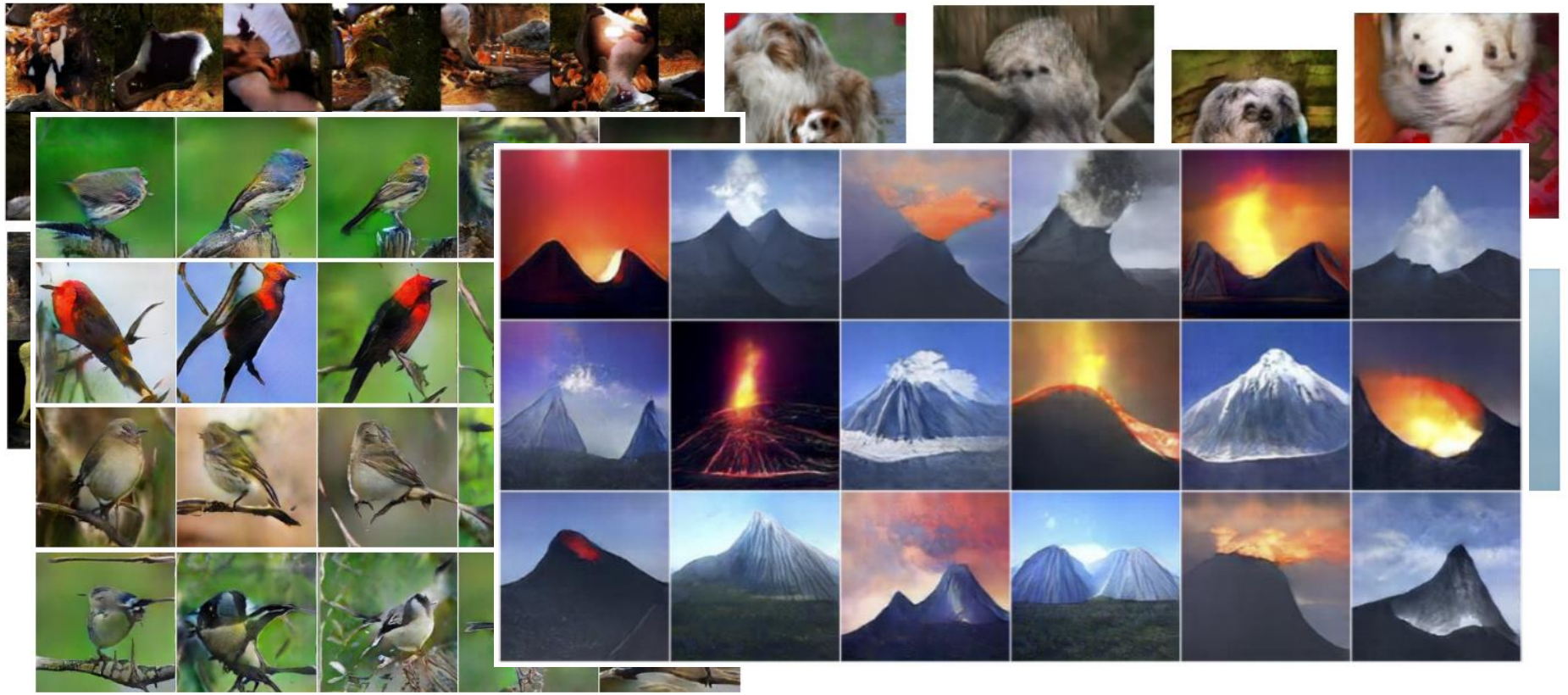


дискриминатор

Что генерируют GANs?

Images from Goodfellow (2016)

- Много хайпа
- Примеры из разных GAN-методов:



Что генерируют GANs?

- Сложно генерировать реалистичные изображения
- Но естественные изображения – сложный объект

Глобальная структура:

Подсчёт частей:

Images from Goodfellow (2016)

GANs быстро развиваются

- Фотореалистичные лица

2014

2015

2016

2017

2018

https://youtu.be/XOxxPcy5Gr4

2020

https://youtu.be/c-NJtV9Jvp0

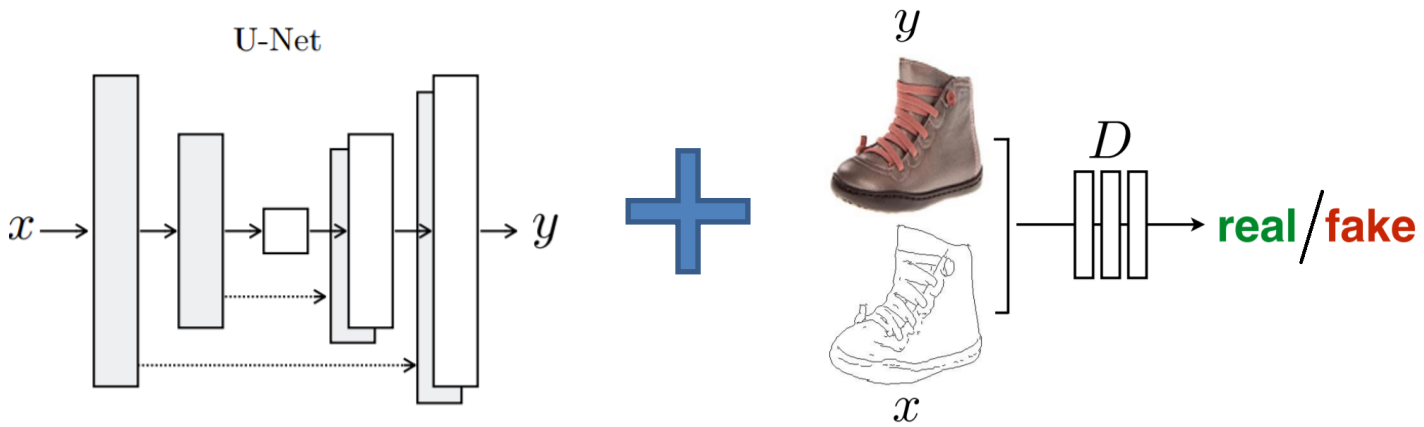
GANs быстро развиваются: BigGAN

[Brock et al. 2018]

Adversarial функции потерь

- Дискриминатор является дифференцируемой функцией потерь!
  - Обучаемой вместе с генератором

- Пример: pix2pix [Isola et al., 2017]



- Используется там, где нет хороших функций потерь
  - Изображения, звук, RL
  - Для текстов – делают, но все сложно

Заключение

- Adversarial examples – фундаментальный феномен ML
  - Можно ли бороться?
  - Является ли ключом к пониманию сетей?
  - Adversarial для людей

- Domain adaptation – способ получить больше данных с метками
  - Очень важно на практике

- Adversarial функции потерь – способ задавать функции потерь для сложных данных

@teenybiscuit