Глубинное обучение

Лекция 8: Оптимизация против нейросетей или Adversarial X

Лектор: Максим Рябинин
Старший исследователь, Yandex Research

Программа ML Residency: yandex.ru/yaintern/research_ml_residency

ФКН ВШЭ, 2022

План лекции

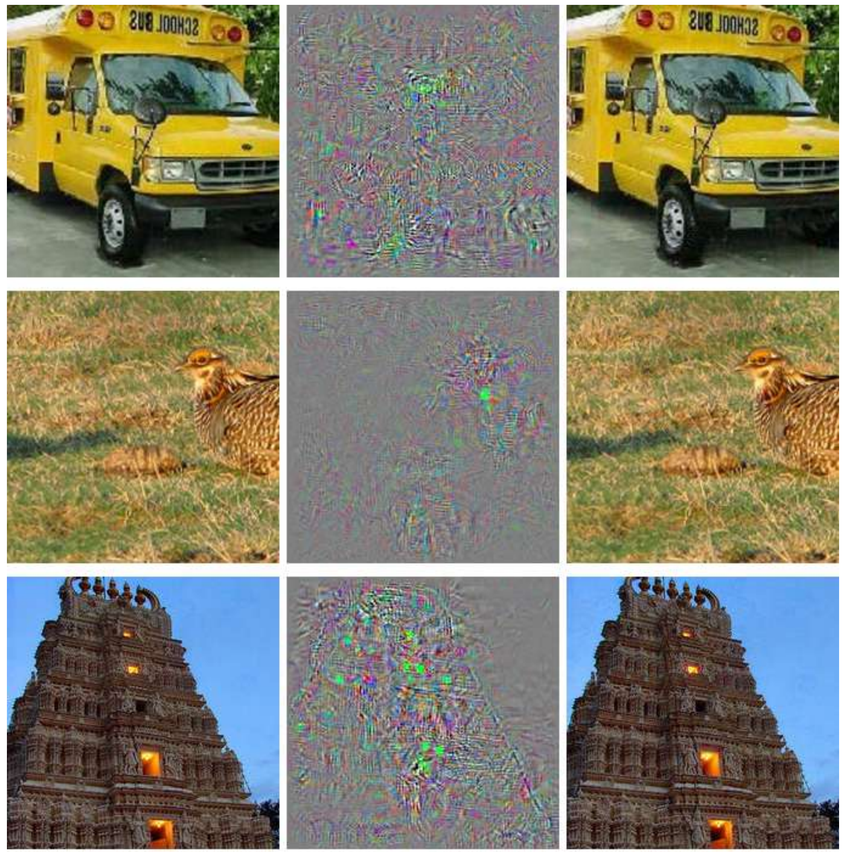
- Противоборствующие/состязательные примеры (adversarial examples)
  - Примеры, причины, следствия
  - Способы построения, борьбы, применения

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

- Нейросети как функции потерь
  - Generative Adversarial Networks
  - ELECTRA

Adversarial examples

- Невидимые глазу изменения меняют результат сети!



правильная классификация | шум | «ostrich»

[Szegedy et al., 2013]

Adversarial examples for text

- 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 ***Cast1e*** 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

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

Adversarial атаки и защиты от них

- Пусть у нас есть обученная сеть (например, на 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

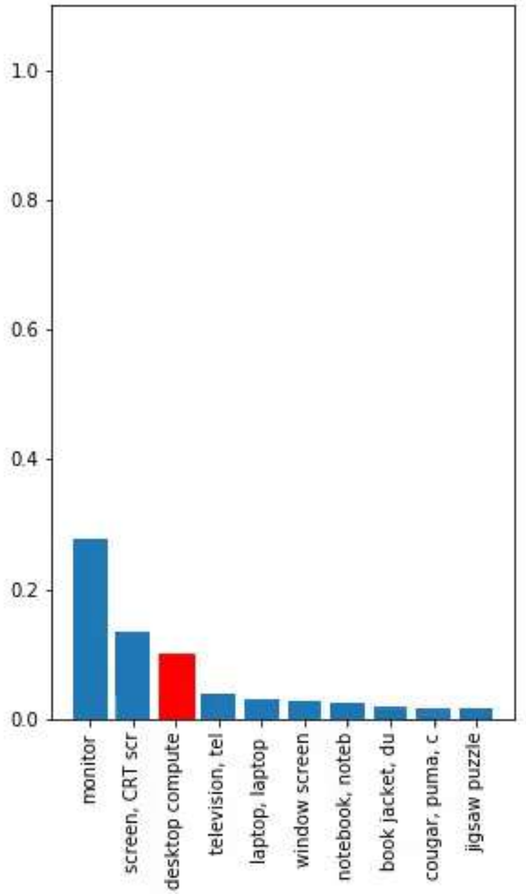
[Szegedy et al., 2013]

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!

- Можно строить устойчивые атаки!



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

[Athalye et al., 2017]

Adversarial training

- Обучение на adversarial-атаках полезно не только для устойчивости к ним же, но и для общего качества модели
- Free Large-Batch (FreeLB): градиентный подъем по шуму, накапливаем градиенты на каждой из итераций
- Стабильно улучшает качество на NLP-бенчмарках

Algorithm 1 "Free" Large-Batch Adversarial Training (FreeLB-K)

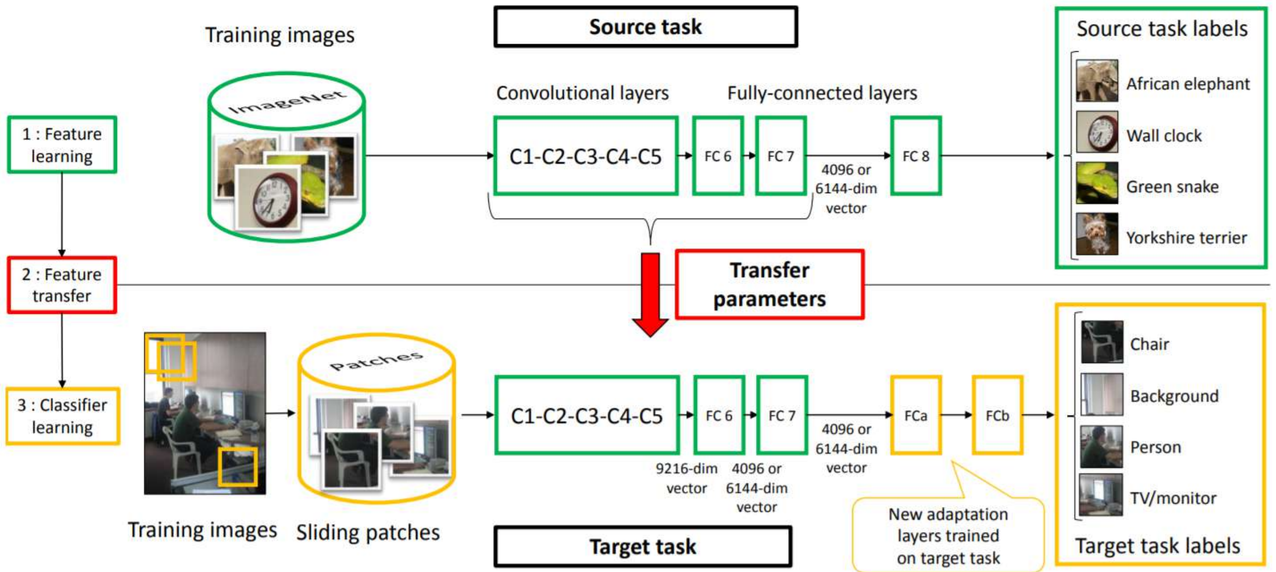
Require: Training samples $X = \{(\boldsymbol{Z}, y)\}$, perturbation bound ϵ, learning rate τ, ascent steps K, ascent step size α

Initialize $\boldsymbol{\theta}$
for epoch $= 1 \ldots N_{ep}$ **do**
  for minibatch $B \subset X$ **do**
    $\boldsymbol{\delta}_0 \leftarrow \frac{1}{\sqrt{N_{\boldsymbol{\delta}}}} U(-\epsilon, \epsilon)$
    $\boldsymbol{g}_0 \leftarrow 0$
    for $t = 1 \ldots K$ **do**
      Accumulate gradient of parameters θ
        $\boldsymbol{g}_t \leftarrow \boldsymbol{g}_{t-1} + \frac{1}{K} \mathbb{E}_{(\boldsymbol{Z}, y) \in B} [\nabla_{\boldsymbol{\theta}} L(f_{\boldsymbol{\theta}}(\boldsymbol{X} + \boldsymbol{\delta}_{t-1}), y)]$
      Update the perturbation δ via gradient ascend
        $\boldsymbol{g}_{adv} \leftarrow \nabla_{\boldsymbol{\delta}} L(f_{\boldsymbol{\theta}}(\boldsymbol{X} + \boldsymbol{\delta}_{t-1}), y)$
        $\boldsymbol{\delta}_t \leftarrow \Pi_{\|\boldsymbol{\delta}\|_F \leq \epsilon} (\boldsymbol{\delta}_{t-1} + \alpha \cdot \boldsymbol{g}_{adv} / \|\boldsymbol{g}_{adv}\|_F)$
    end for
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \tau \boldsymbol{g}_K$
  end for
end for

[Zhu et al., 2019]

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-модель, обученную на ImageNet
  - Откинуть голову сети, дообучиться на свою задачу



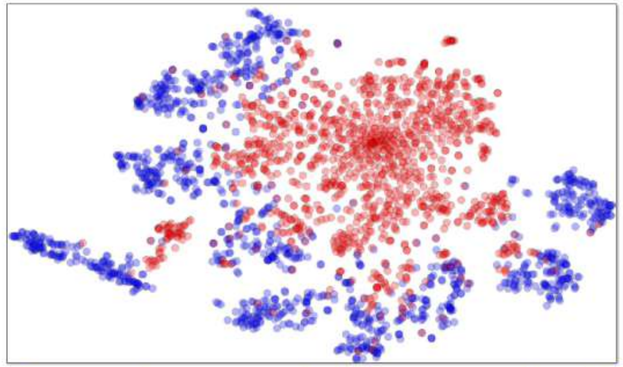
[Oquab et al., 2014]

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-модель, обученную на ImageNet
  - Откинуть голову сети, дообучиться на свою задачу
- Формула работает хуже, если данные слишком различны
  - Явление называют domain/distribution shift
- Доменная адаптация
  - Есть много размеченных данных в source domain
  - Есть много неразмеченных данных в target domain
  - Как это использовать?

Как убрать разницу (domain shift)?

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



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

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

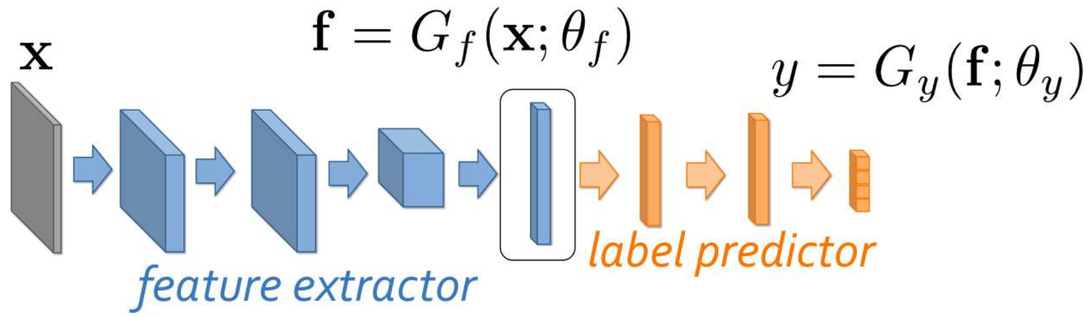


Image credit: Victor Lempitsky

[Ganin&Lempitsky, 2014]

Как убрать разницу (domain shift)?

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

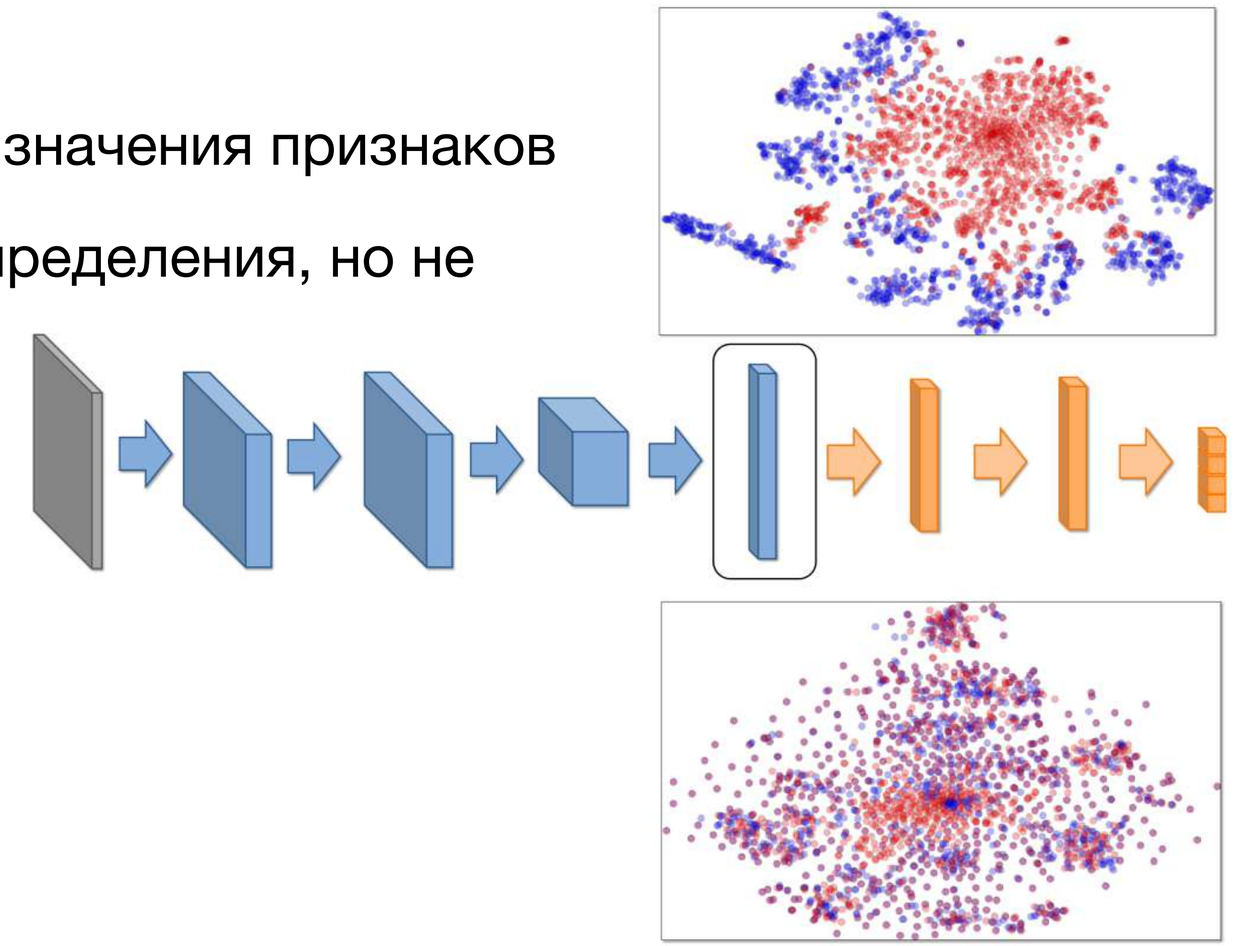


Image credit: Victor Lempitsky

[Ganin&Lempitsky, 2014]

Как убрать разницу (domain shift)?

- Идея – использовать классификатор доменов

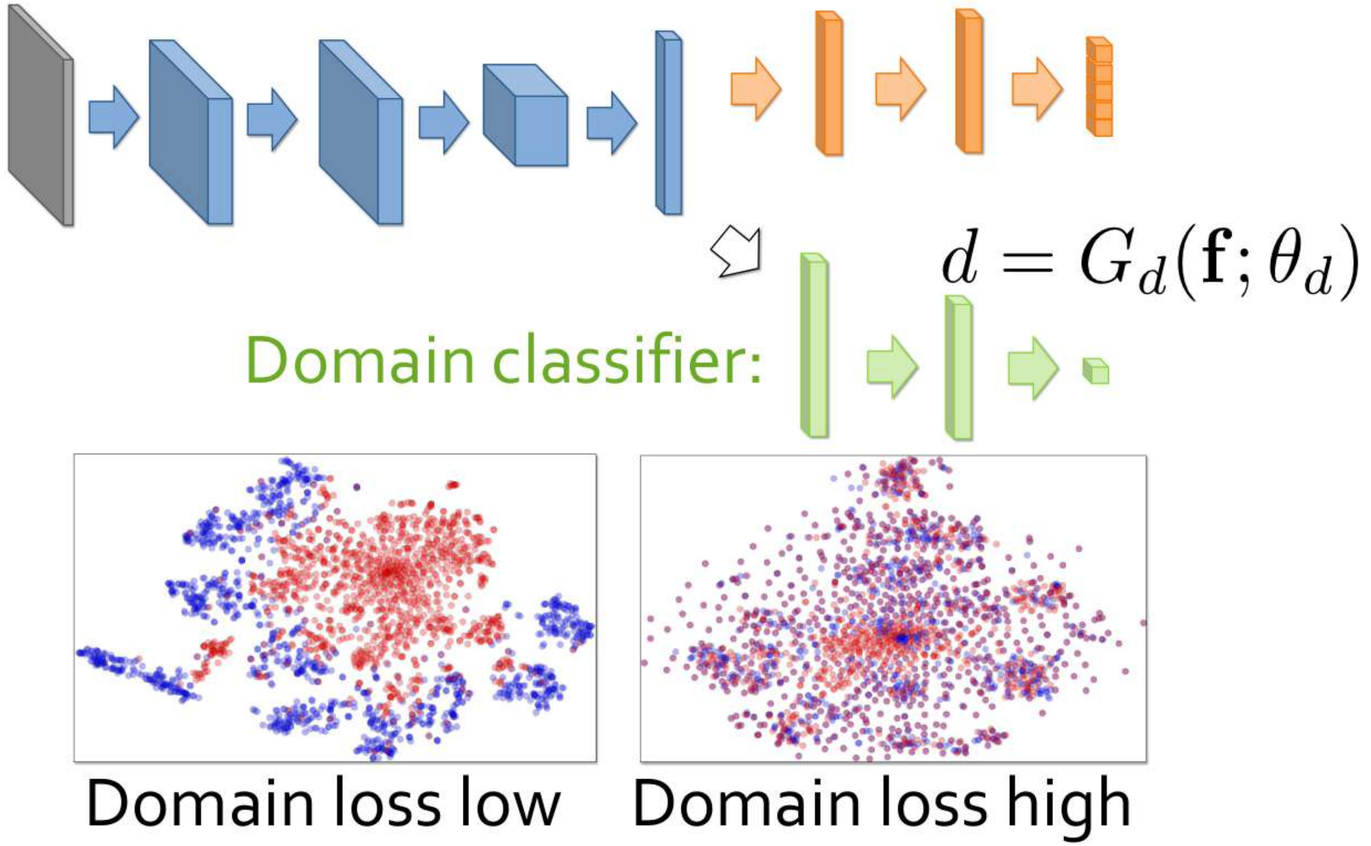


Image credit: Victor Lempitsky

[Ganin&Lempitsky, 2014]

Как убрать разницу (domain shift)?

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

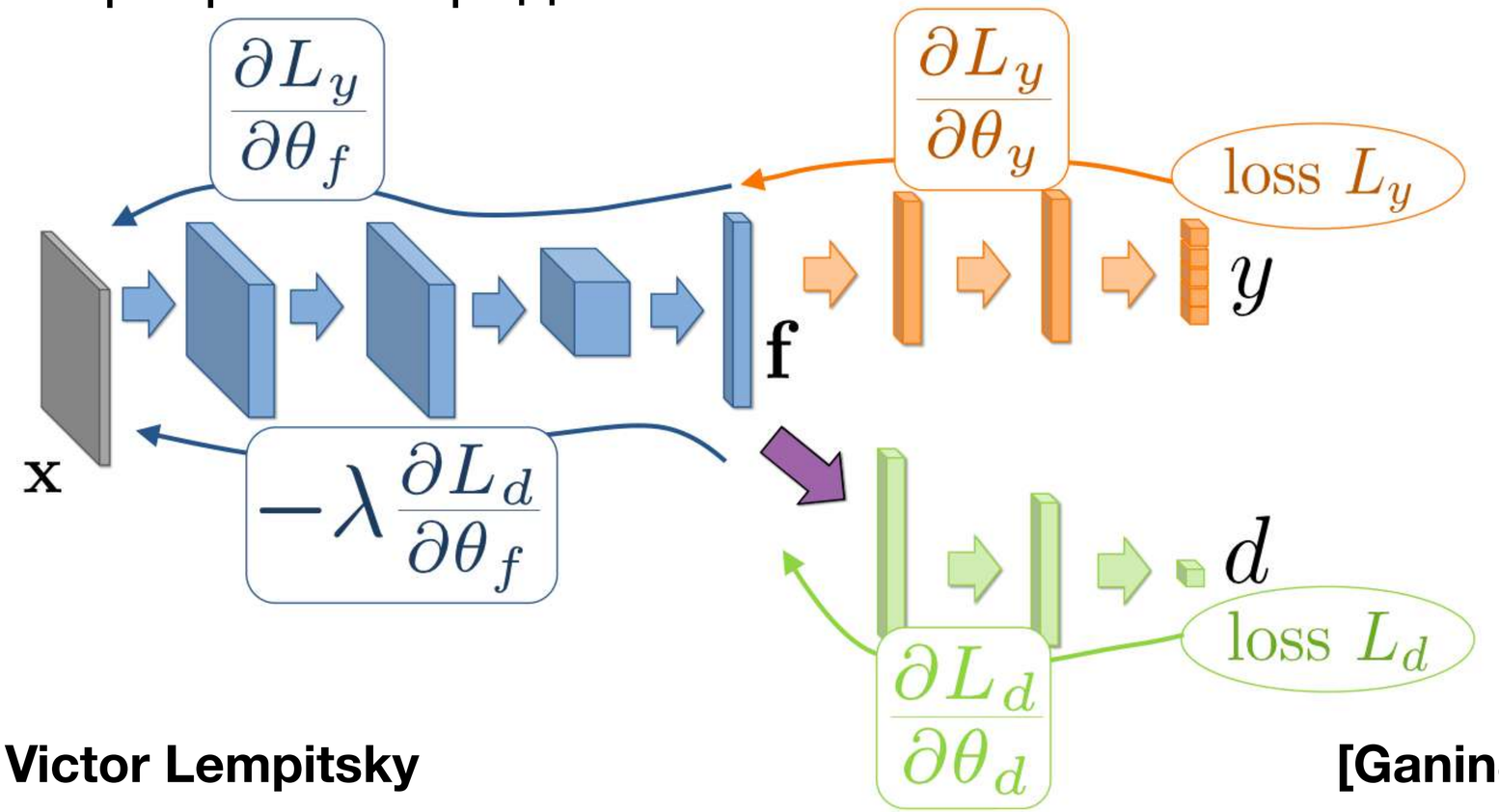


Image credit: Victor Lempitsky

[Ganin&Lempitsky, 2014]

Как убрать разницу (domain shift)?

- 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)
```

[Ganin&Lempitsky, 2014]

Как убрать разницу (domain shift)?

- 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:

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

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

[Ganin&Lempitsky, 2014]

Generative Adversarial Networks

- 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

[Goodfellow et al., 2014]

Generative Adversarial Networks

Две сети:

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

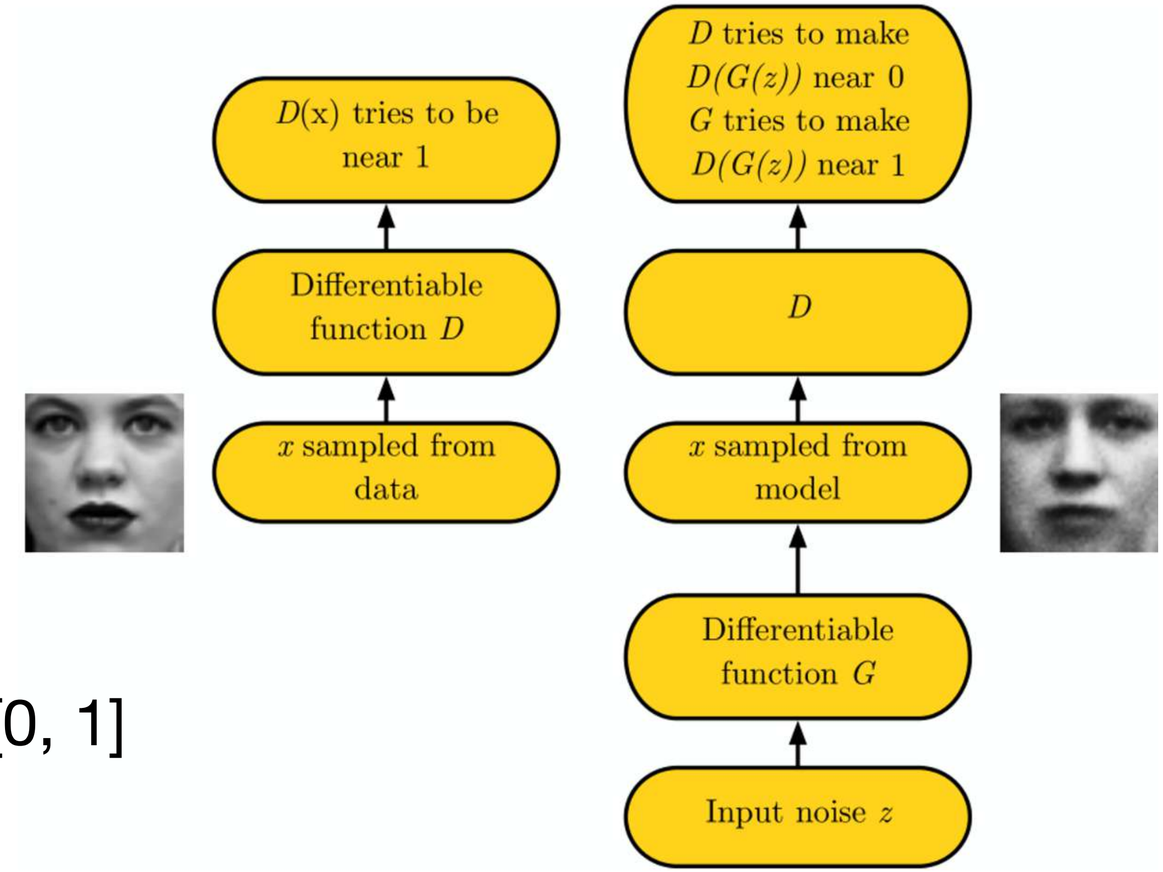


Image credit: Ian Goodfellow

[Goodfellow et al., 2014]

Generative Adversarial Networks

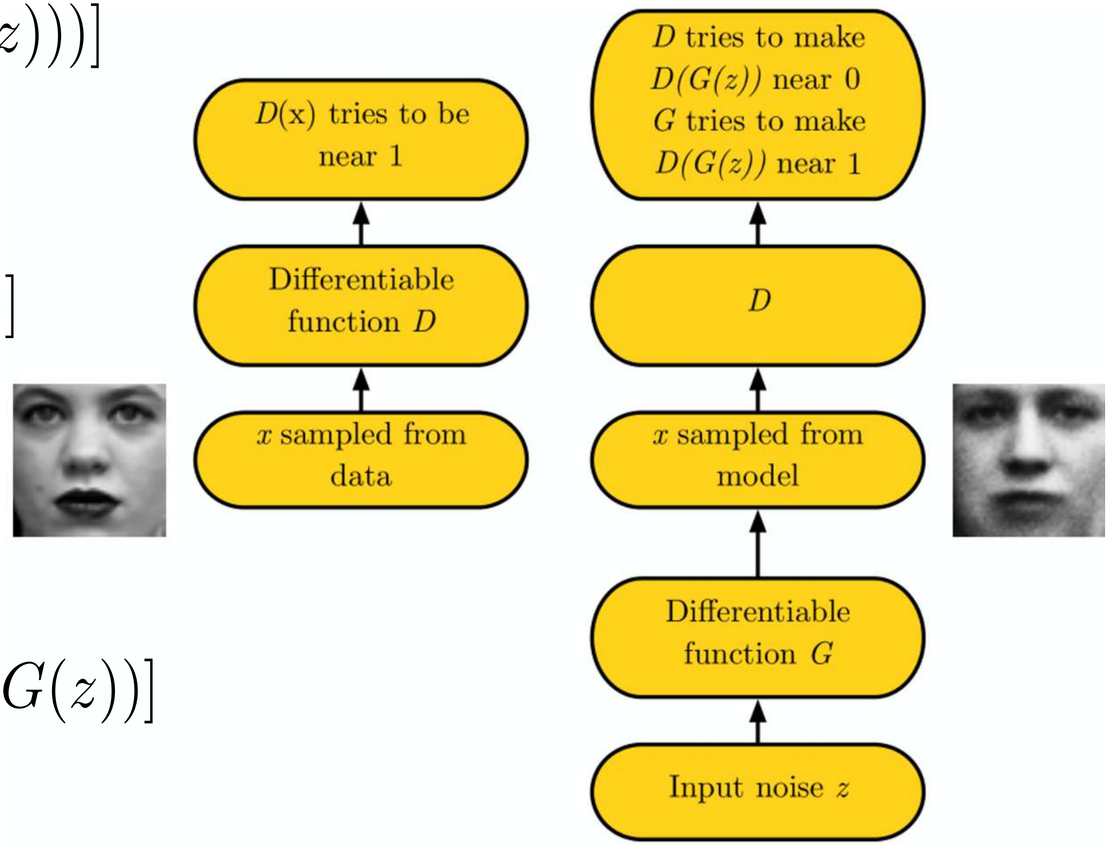
Седловая точка:

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

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 log loss



Обучение генератора:

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

Затухает градиент, когда плохой G (начало обучения)

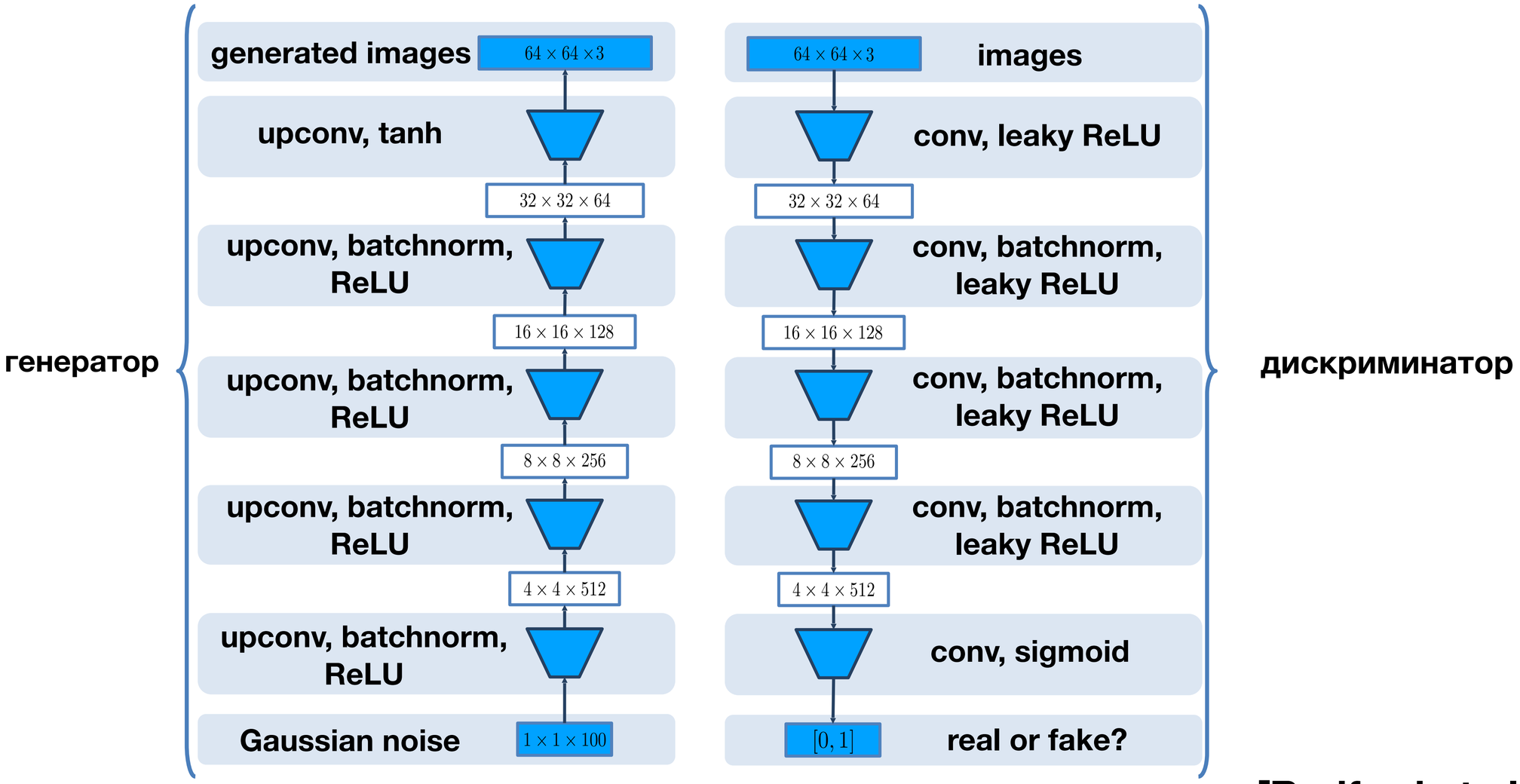
Image credit: Ian Goodfellow

[Goodfellow et al., 2014]

GANs работают?

- Часто модель сложно заставить работать
- Много тонкостей реализации – очень важен исходный код
- Требуется очень много вычислительных ресурсов
- BigGAN [Brock et al., 2018], StyleGANv2 [Karras et al., 2021], StyleGANv3 (aka Alias-Free GAN) [Karras et al., 2021]

Архитектура DCGAN

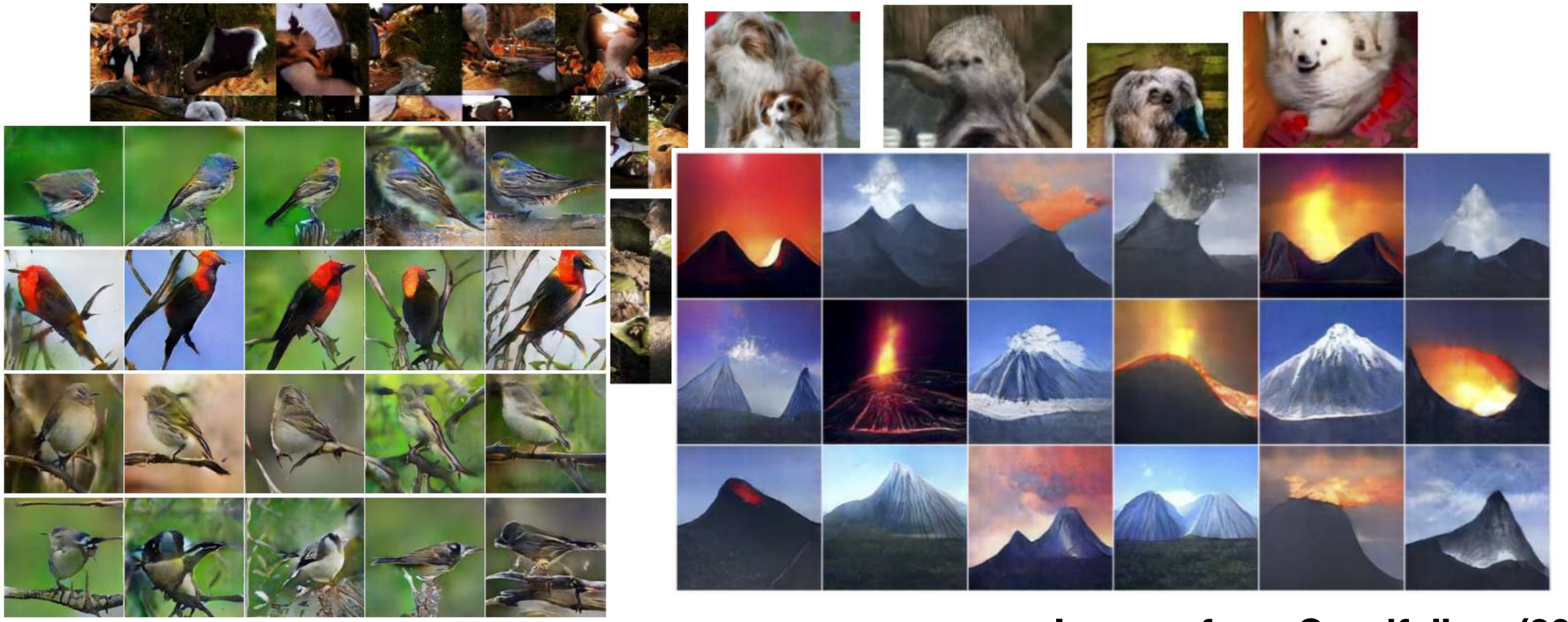


[Radford et al. 2014]

Что генерируют GANs?

- Много хайпа

- Примеры из разных GAN-методов:



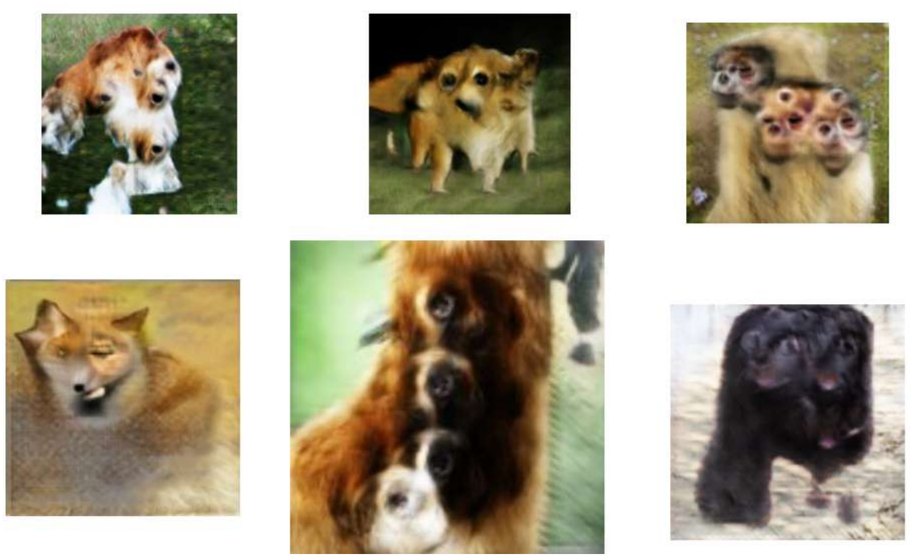
Images from Goodfellow (2016)

Что генерируют GANs?

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

Глобальная структура:

Подсчёт частей:



Images from Goodfellow (2016)

GANs быстро развиваются

2014

2015

2016

2017

2018

2020

https://youtu.be/XOxxPcy5Gr4

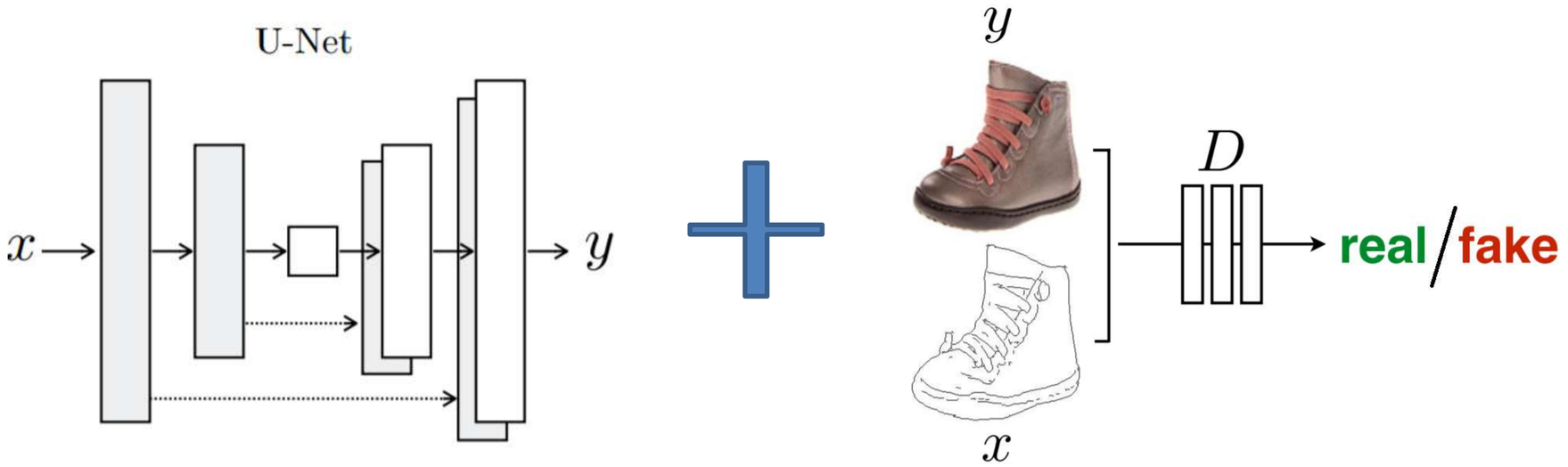
https://youtu.be/c-NJtV9Jvp0

GANs быстро развиваются: BigGAN

[Brock et al., 2018]

Adversarial-функции потерь

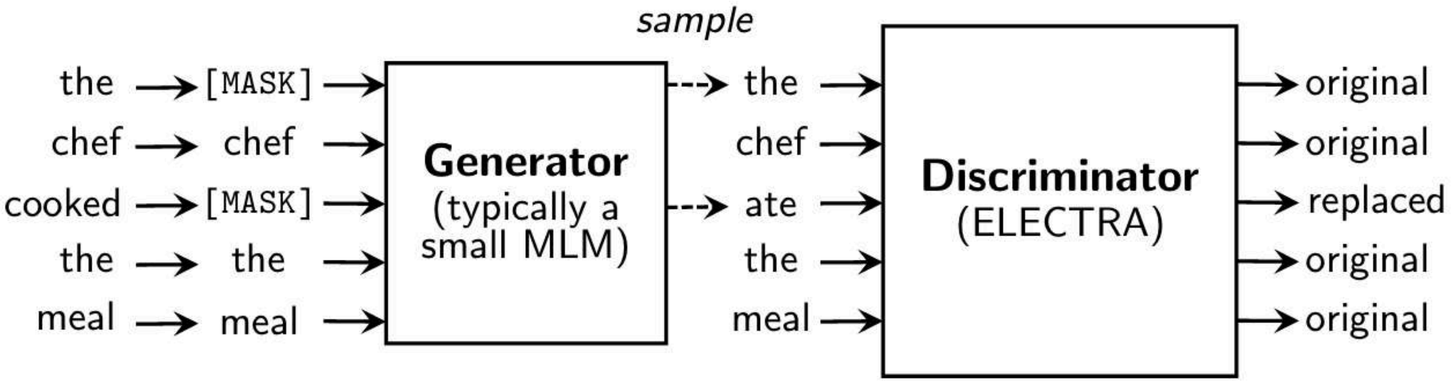
- Дискриминатор является дифференцируемой функцией потерь!
  - Обучаемой вместе с генератором
- Пример: pix2pix [Isola et al., 2017]

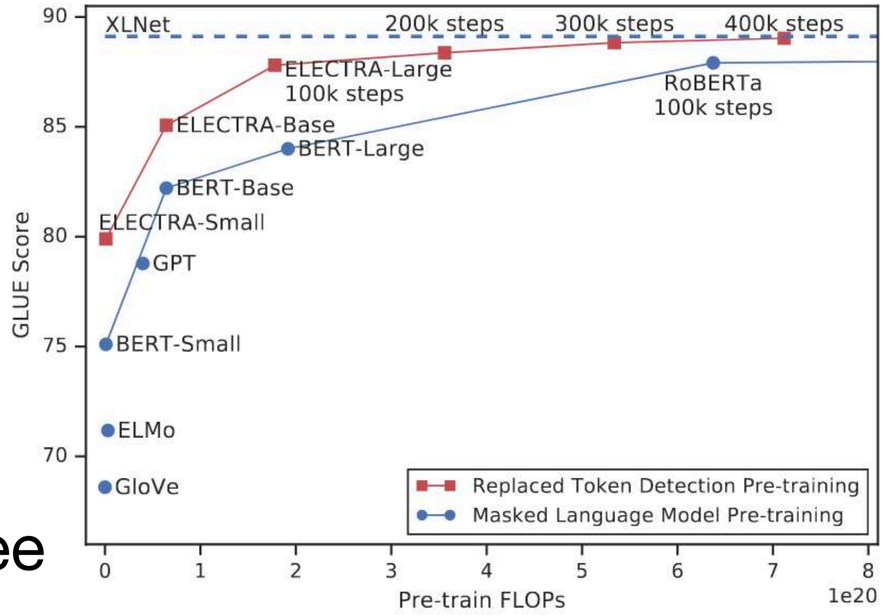


- Используется там, где нет хороших функций потерь
  - Изображения, звук, RL
  - Для текстов – делают, но все сложно

Bonus: ELECTRA

- Сигнал из нейросети бывает заметно богаче сигнала из данных
- Можно использовать для более эффективного обучения текстовых моделей





- В итоге модель сходится существенно быстрее
- Важно: **это не GAN** (генератор не обманывает)

[Clark et al., 2020]

Заключение

- Adversarial examples – фундаментальный феномен ML
  - Можно ли бороться?
  - Является ли ключом к пониманию сетей?
  - Adversarial для людей
- Domain adaptation – способ использовать больше данных с метками из другого домена
  - Очень важно на практике
- Adversarial-функции потерь – способ задавать функции потерь для сложных данных