Глубокие нейросети со структурированным выводом: обзор результатов и открытых проблем

Виктор Лемпицкий
СколТех

Skoltech
Skolkovo Institute of Science and Technology

Семинар НМУ, 17/12/2016

План

1. «Стандартные» свёрточные сети
2. «Практическая» оптимизация
3. Синтезирующие сети с простыми функциями потери
4. Синтез через повторение статистик
5. Играющие сети (adversarial networks)

План

1. **«Стандартные» свёрточные сети**
2. «Практическая» оптимизация
3. Синтезирующие сети с простыми функциями потери
4. Синтез через повторение статистик
5. Играющие сети (adversarial networks)

2006

CAPTCHA

asirra

Компьютерное зрение= 60%

$0.6^{12} = 0.00217$

2014

Completed • Swag • 215 teams

Dogs vs. Cats

Wed 25 Sep 2013 – Sat 1 Feb 2014 (8 months ago)

Dashboard

Private Leaderboard - Dogs vs. Cats

This competition has completed. This leaderboard reflects the final standings.

See someone

#	Δ1w	Team Name * in the money	Score	Entries	Last Submission UTC (Best – Las
1	–	Pierre Sermanet *	0.98914	5	Sat, 01 Feb 2014 21:43:19 (-
2	↑26	orchid *	0.98309	17	Sat, 01 Feb 2014 23:52:30
3	–	Owen	0.98171	15	Sat, 01 Feb 2014 17:04:40 (-
4	new	Paul Covington	0.98171	3	Sat, 01 Feb 2014 23:05:20
5	↓3	Maxim Milakov	0.98137	24	Sat, 01 Feb 2014 18:20:58

$$0.989^{12} = 0.875$$

2014

Microsoft Research

Search Mi

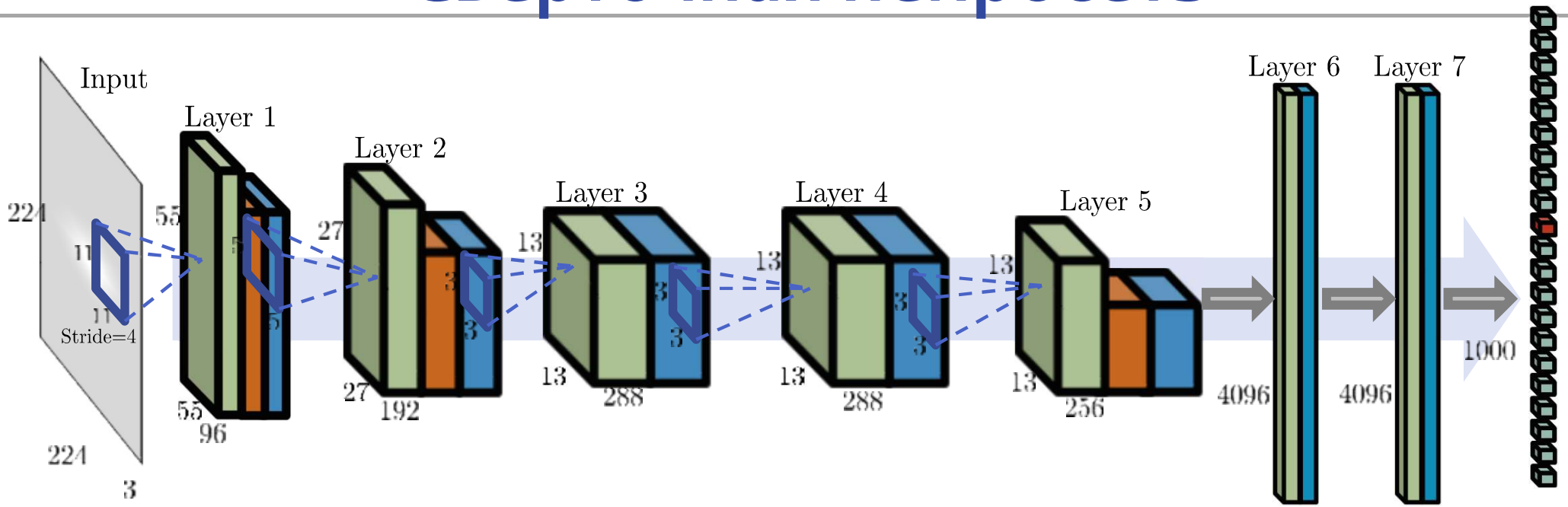
Our research Connections Careers About us

All Downloads Events Groups News People Projects Publications

ASIRRA

After 8 years of operation, Asirra is shutting down effective October 1, 2014. Thank you to all of our users!

Сверточная нейросеть



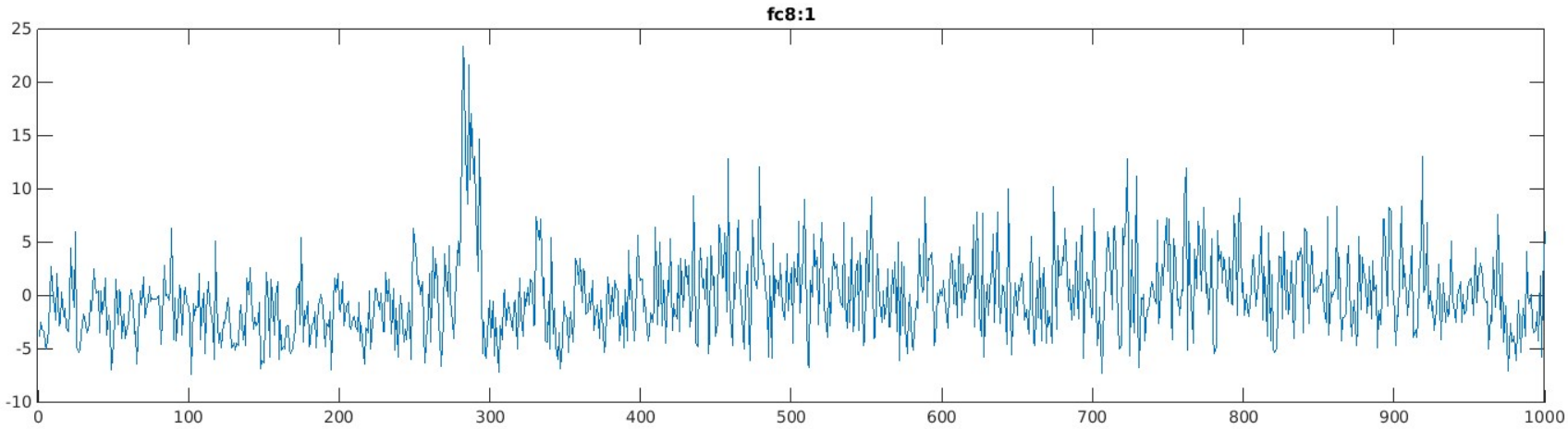
Операции (*слои*):

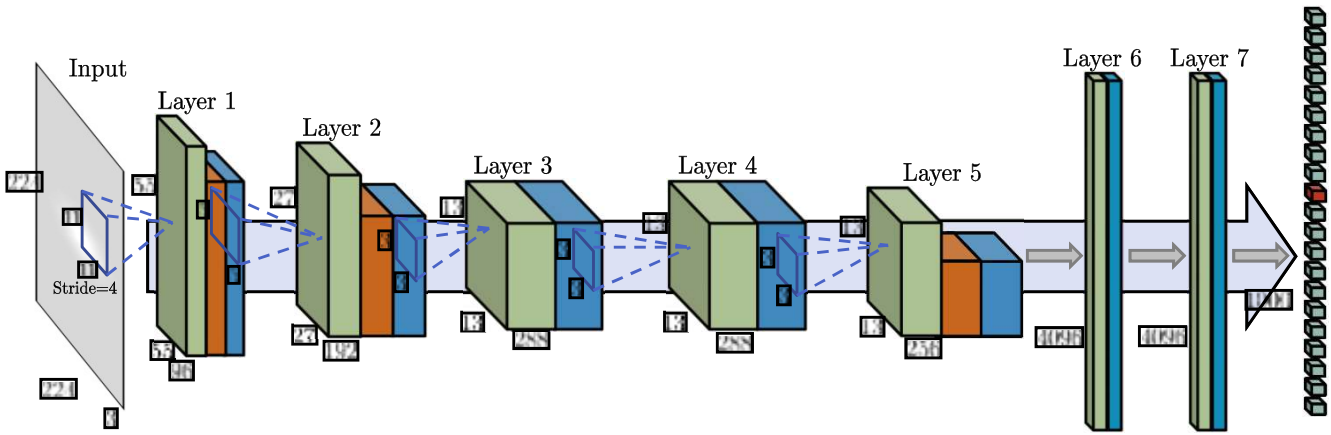
- обобщенные свертки
- макс-пулинг
- поэлементные нелинейности
- умножение на матрицу («полносвязный слой»)

Два типа переменных:

- *Активации* («карты», «нейроны», «представление»)
- *Параметры слоёв* («фильтры», «ядра»)

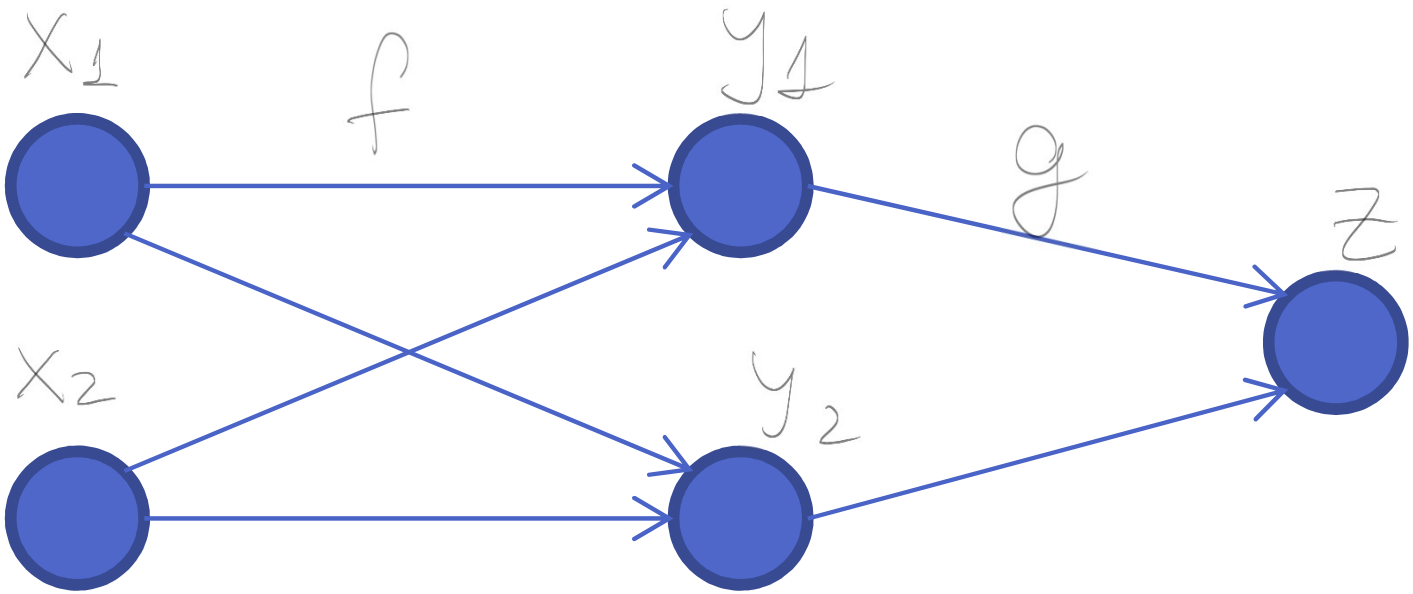
Представления внутри нейросети





"Глубокие нейросети со структурированным выводом"

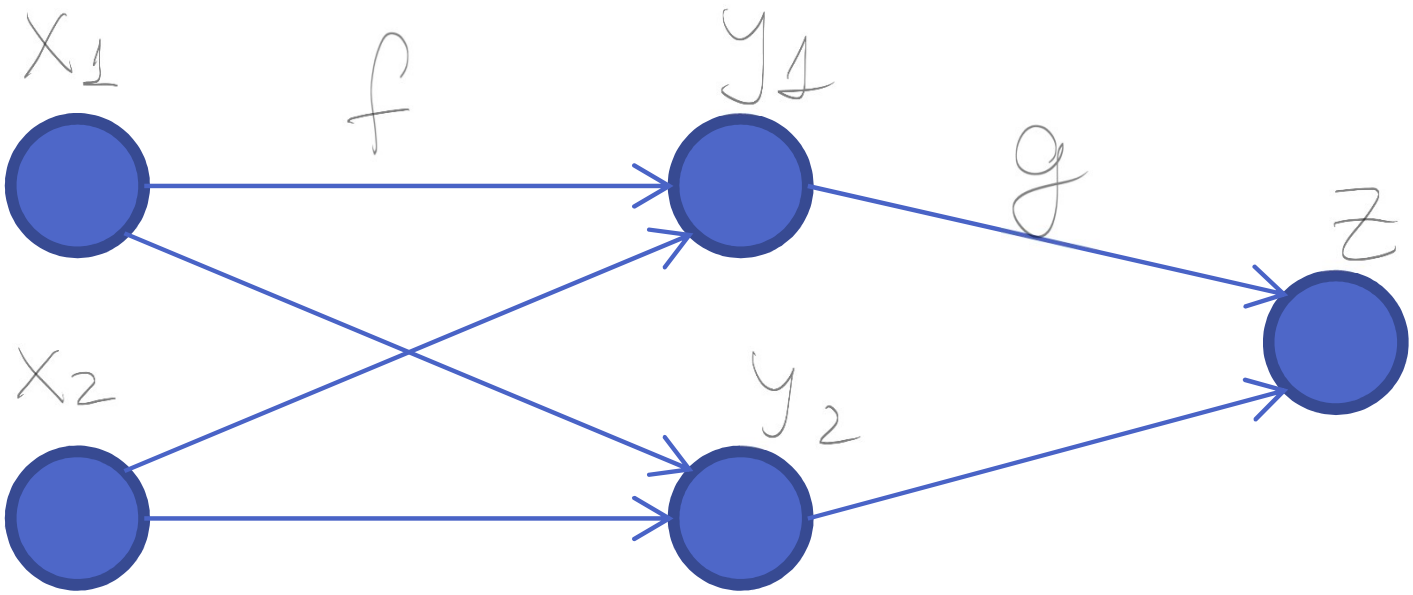
Производная сложной функции



$$\frac{dz}{dy} = \begin{bmatrix} \frac{\partial g}{\partial y_1}(y_1, y_2) \\ \frac{\partial g}{\partial y_2}(y_1, y_2) \end{bmatrix}$$

$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y} \cdot \frac{\partial y_1}{\partial x_1} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_1}$$

Производная сложной функции



$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y} \cdot \frac{\partial y_1}{\partial x_1} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_1}$$

$$\frac{\partial z}{\partial x_2} = \frac{\partial z}{\partial y} \cdot \frac{\partial y_1}{\partial x_2} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_2}$$

Производная сложной функции

$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y} \cdot \frac{\partial y_1}{\partial x_1} + \frac{\partial z}{\partial y} \cdot \frac{\partial y_2}{\partial x_1}$$

$$\frac{\partial z}{\partial x_2} = \frac{\partial z}{\partial y} \cdot \frac{\partial y_1}{\partial x_2} + \frac{\partial z}{\partial y} \cdot \frac{\partial y_2}{\partial x_2}$$

$$\frac{dz}{dx} \cdot \begin{bmatrix} \frac{\partial z}{\partial x_1} \\ \\ \frac{\partial z}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x} & \frac{\partial y_2}{\partial x} \\ \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial z}{\partial y_1} \\ \\ \frac{\partial z}{\partial y_2} \end{bmatrix}$$

Производная сложной функции

$$\frac{dz}{dx} = \begin{bmatrix} \frac{\partial z}{\partial x_1} \\ \frac{\partial z}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x} & \frac{\partial y_2}{\partial x} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial z}{\partial y_1} \\ \frac{\partial z}{\partial y_2} \end{bmatrix}$$

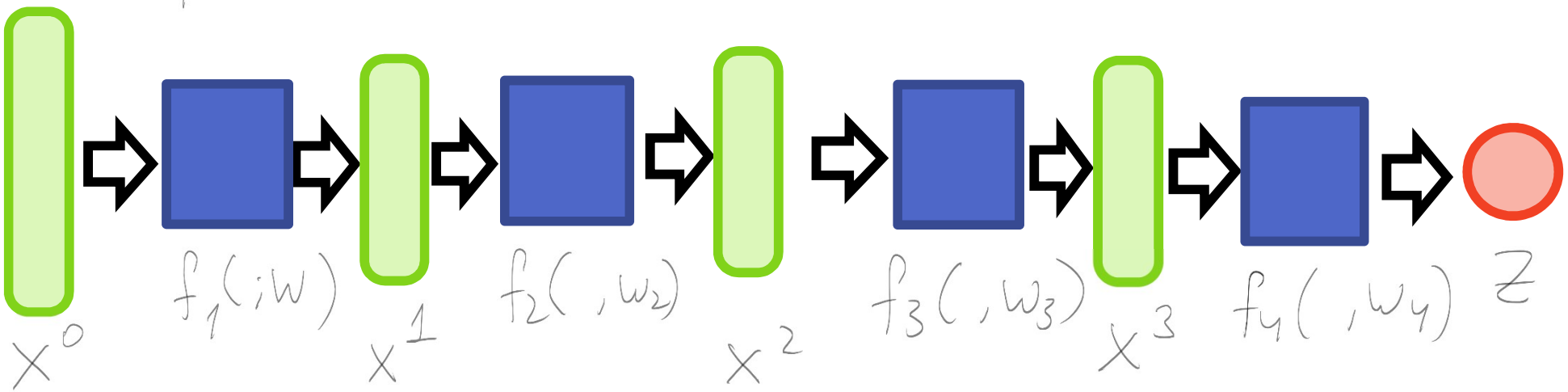
$$\frac{dz}{dx} = \left(\frac{dy}{dx}\right)^T \frac{dz}{dy}$$

Обратное распространение

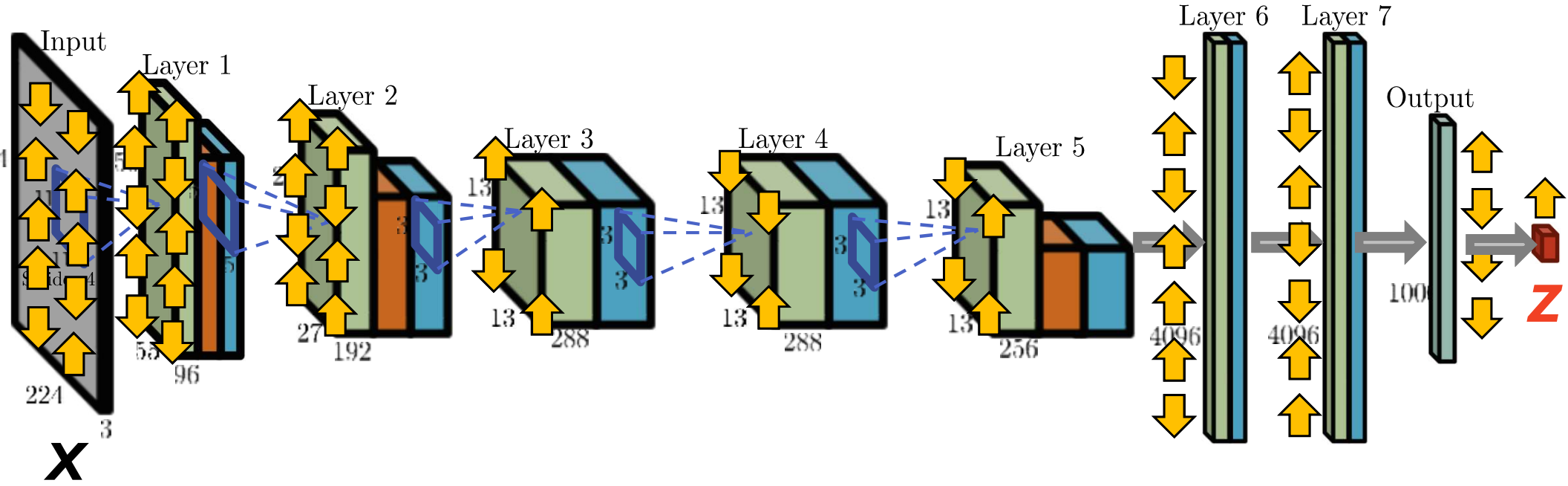
$$\frac{dz}{dw_3} = \frac{dx^3}{dw_3}^T \cdot \frac{dz}{dx^3} \qquad \frac{dz}{dx^2} = \frac{dx^3}{dx^2}^T \frac{dz}{dx^3}$$

$$\frac{dz}{dw_2} \qquad \frac{dx^2}{dw_2}^T \cdot \frac{dz}{dx_2} \qquad \frac{dz}{dx^1} = \frac{dx^2}{dx^1}^T \frac{dz}{dx^2}$$

$$\frac{dz}{dw_1} \qquad \frac{dx^1}{dw_1}^T \cdot \frac{dz}{dx_1} \qquad \frac{dz}{dx^0} = \frac{dx^1}{dx^0}^T \frac{dz}{dx^1}$$



Обучение обратным распространением



[Leibnitz;
L'Hopital 1696]

$$(f \circ g)'(x) = g'(x) \cdot f'(g(x))$$

Операции (*слои*):

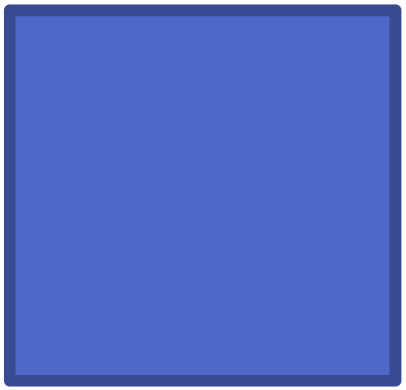
обобщенные свертки

макс-пулинг

поэлементные нелинейности

умножение на матрицу

Определение новых слоев



Каждый слой опредлеяется:

- прямым распространением: $y = f(x)$
- обратным распространением:

$z(x) \quad z(f(x;w)) \quad y = f(x;w)$

$$\frac{dz}{dx} = \frac{dy}{dx}^T \cdot \frac{dz}{dy} \qquad \frac{dz}{dw} = \frac{dy}{dw}^T \cdot \frac{dz}{dy}$$

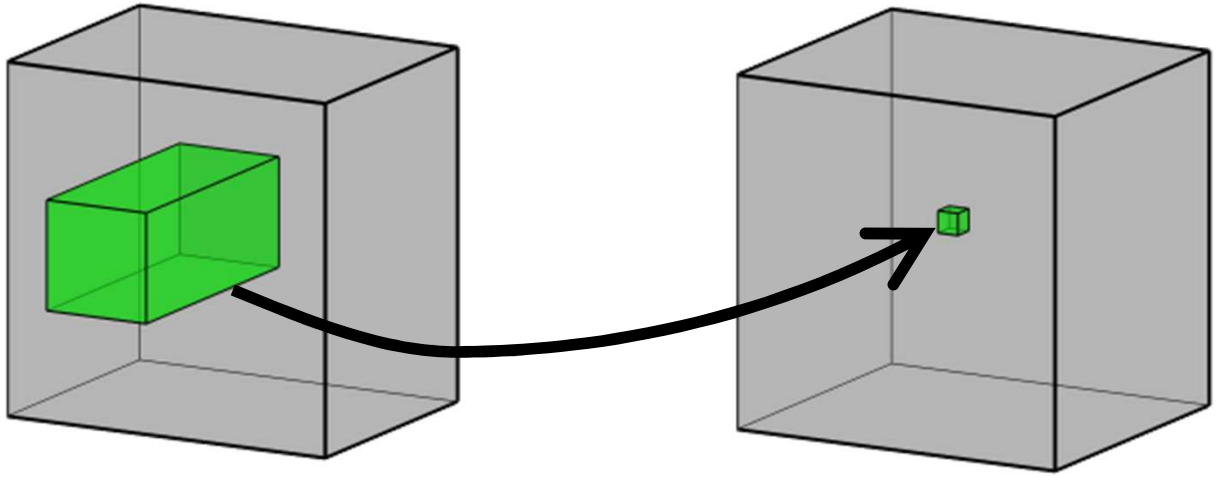
ООП и глубокое обучение

```
abstract class Layer {
        params w,dzdw;
        virtual y = forward(x);
        virtual dzdx = backward(dzdy,x,y);
        // should compute dzdw as well

        void update (tau) {
                w = w+tau*dzdw;
        }
};
```

На практике нужно использовать векторные и матричные операции

Обобщенная свертка

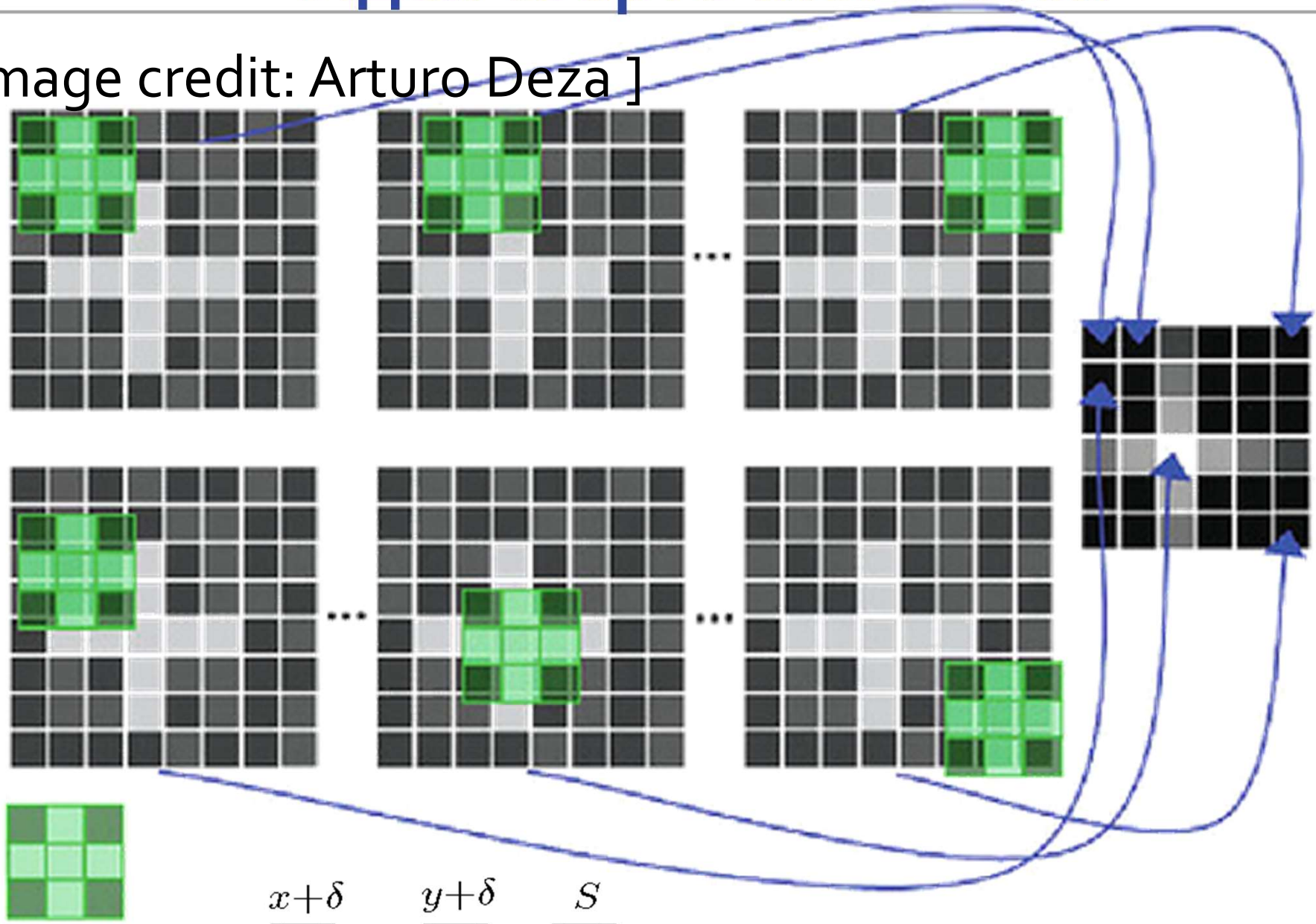


$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K(i-x+\delta, j-y+\delta, s, t)\, U(i,j,s)$$

- Локально-линейное преобразование
- «Самый важный» слой в компьютерном зрении

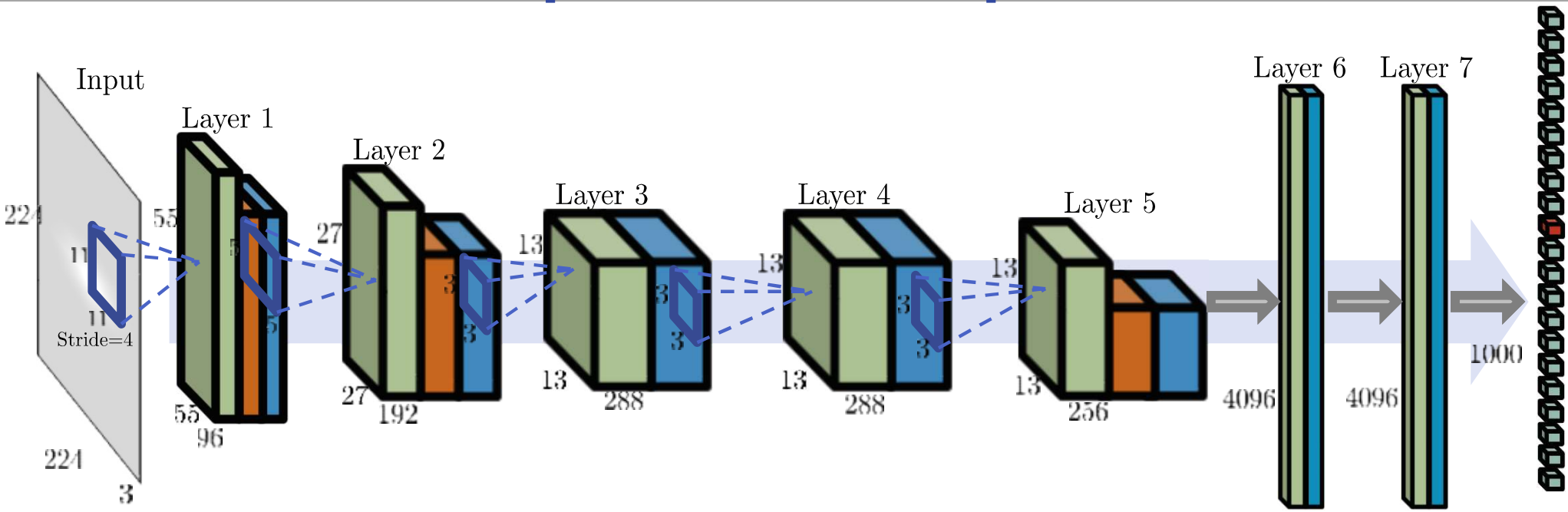
Идея сверточного слоя

[Image credit: Arturo Deza]



$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K(i-x+\delta, j-y+\delta, s, t)\, U(i,j,s)$$

Сверточная нейросеть



Операции (*слои*):

обобщенные свертки

макс-пулинг

поэлементные нелинейности

умножение на матрицу

2012: Image-net

IMAGENET

www.image-net.org

14,197,122 изображений, 21841 классов

(большей частью работа идет с подмножеством 1000x1000)

Statistics of high level categories

High level category	# synset (subcategories)	Avg # images per synset	Total # images
amphibian	94	591	56K
animal	3822	732	2799K
appliance	51	1164	59K
bird	856	949	812K
covering	946	819	774K
device	2385	675	1610K
fabric	262	690	181K
fish	566	494	280K
flower	462	735	339K
food	1495	670	1001K
fruit	309	607	188K

Image-net

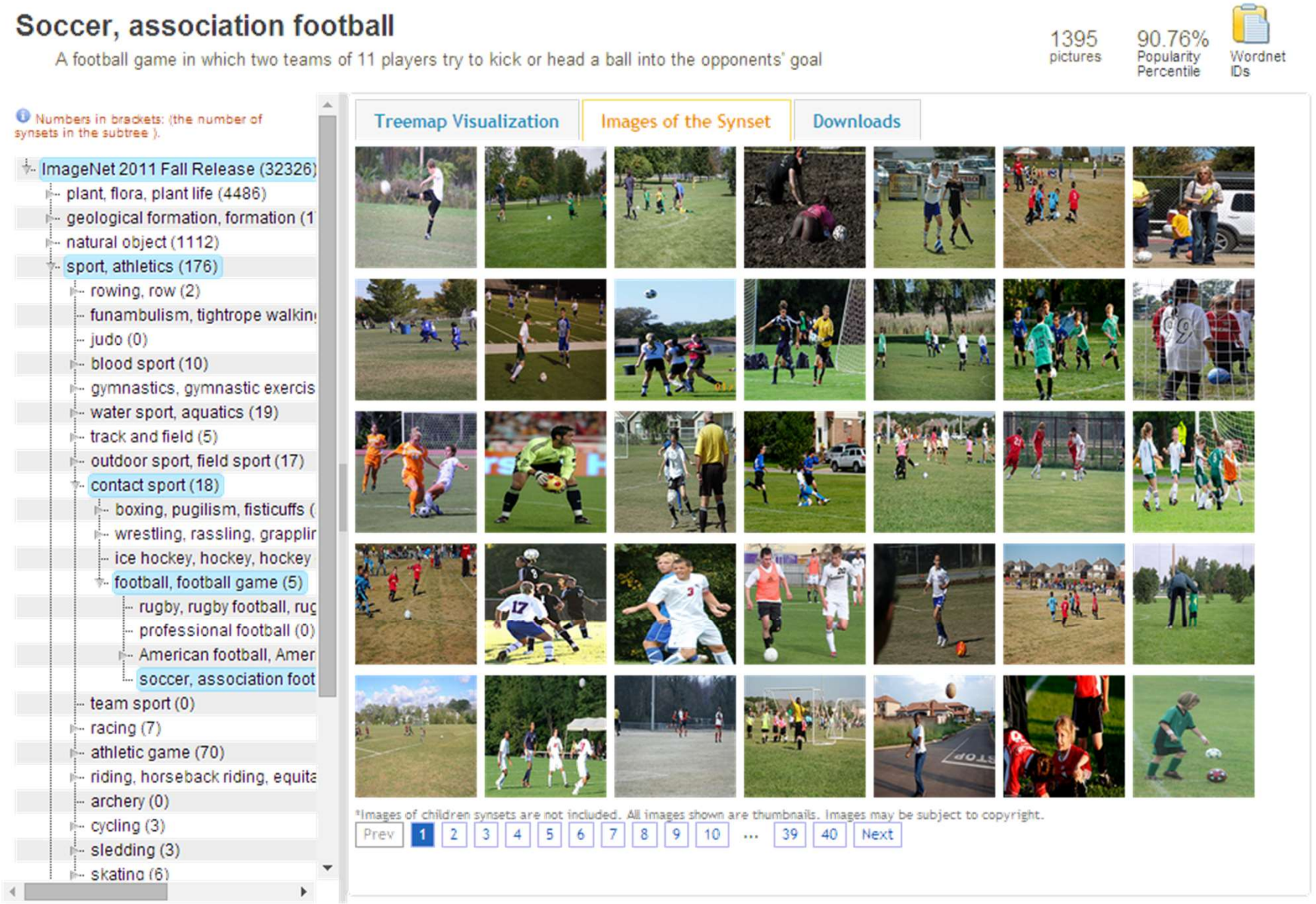
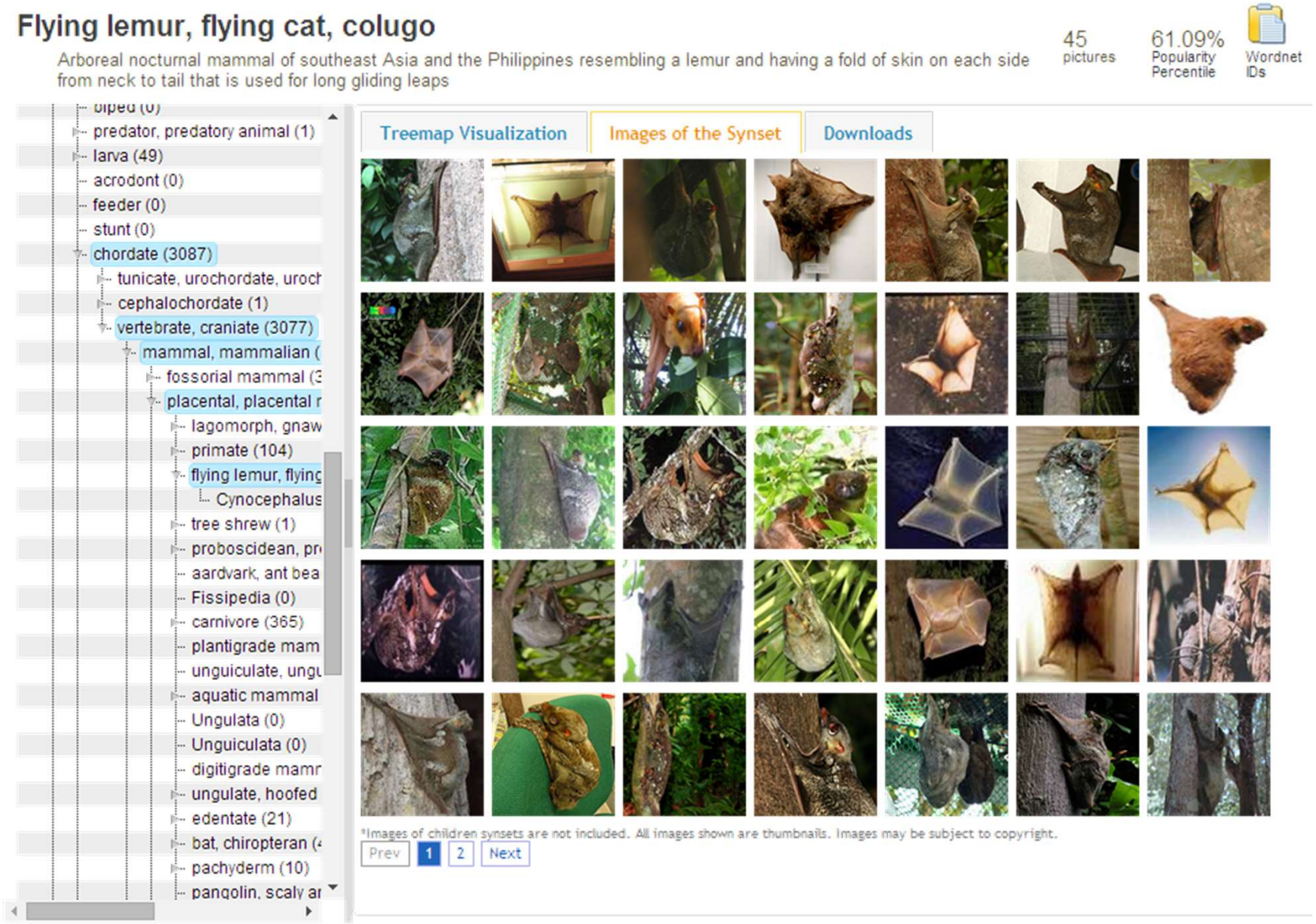
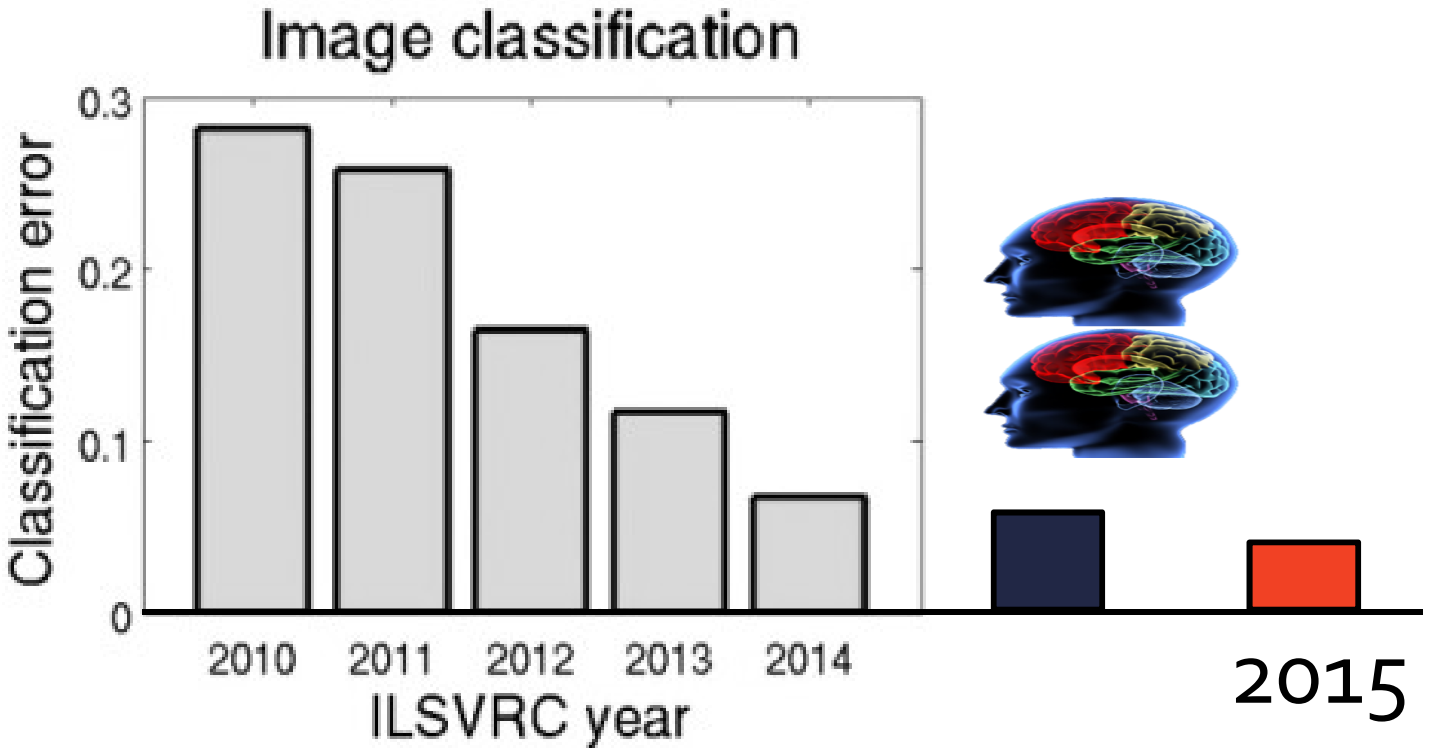

Soccer, association football
A football game in which two teams of 11 players try to kick or head a ball into the opponents' goal
1395 pictures
90.76% Popularity Percentile
Wordnet IDs
Numbers in brackets: (the number of synsets in the subtree).
ImageNet 2011 Fall Release (32326)
plant, flora, plant life (4486)
geological formation, formation (1
natural object (1112)
sport, athletics (176)
rowing, row (2)
funambulism, tightrope walkin
judo (0)
blood sport (10)
gymnastics, gymnastic exercis
water sport, aquatics (19)
track and field (5)
outdoor sport, field sport (17)
contact sport (18)
boxing, pugilism, fisticuffs (
wrestling, rassling, grapplir
ice hockey, hockey, hockey
football, football game (5)
rugby, rugby football, rug
professional football (0)
American football, Amer
soccer, association foot
team sport (0)
racing (7)
athletic game (70)
riding, horseback riding, equita
archery (0)
cycling (3)
sledding (3)
skating (6)
Treemap Visualization
Images of the Synset
Downloads
*Images of children synsets are not included. All images shown are thumbnails. Images may be subject to copyright.
Prev 1 2 3 4 5 6 7 8 9 10 ... 39 40 Next

Image-net

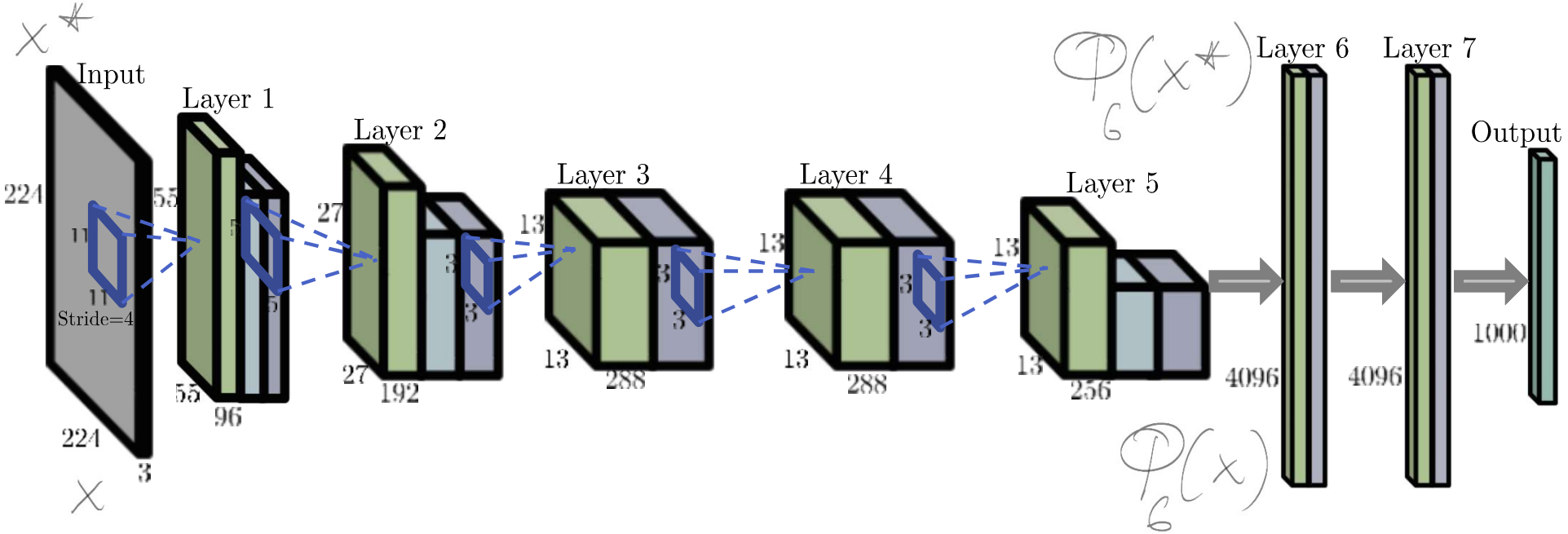

Flying lemur, flying cat, colugo
Arboreal nocturnal mammal of southeast Asia and the Philippines resembling a lemur and having a fold of skin on each side from neck to tail that is used for long gliding leaps
45 pictures
61.09% Popularity Percentile
Wordnet IDs
predator, predatory animal (1)
larva (49)
acrodont (0)
feeder (0)
stunt (0)
chordate (3087)
tunicate, urochordate, uroch
cephalochordate (1)
vertebrate, craniate (3077)
mammal, mammalian (
fossorial mammal (3
placental, placental r
lagomorph, gnaw
primate (104)
flying lemur, flying
Cynocephalus
tree shrew (1)
proboscidean, pr
aardvark, ant bea
Fissipedia (0)
carnivore (365)
plantigrade mam
unguiculate, ungu
aquatic mammal
Ungulata (0)
Unguiculata (0)
digitigrade mamr
ungulate, hoofed
edentate (21)
bat, chiropteran (
pachyderm (10)
pangolin, scaly ar
Treemap Visualization
Images of the Synset
Downloads
*Images of children synsets are not included. All images shown are thumbnails. Images may be subject to copyright.
Prev
1
2
Next

Текущее состояние



- Практически достигнут уровень мозга в задаче классификации (изображение -> метка класса)
- Метка класса -> изображение = ?

Синтез через подсчет прообразов



$$\hat{x} = \arg\min_x Z(x) = \arg\min_x \|\Phi_6(x^*) - \Phi_6(x)\|^2 + \lambda R(x)$$

«Стандартная» регуляризация:

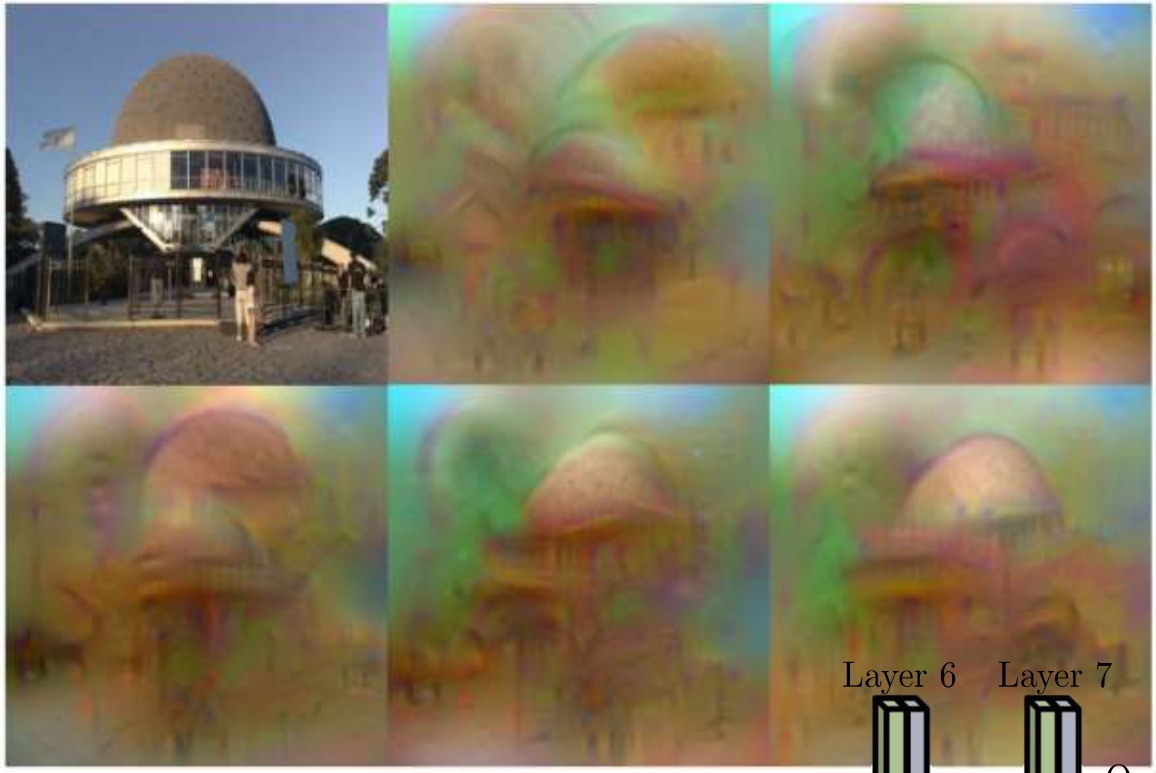
$$\sum \left((x_{i,j-1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2 \right)^{\beta/2}$$

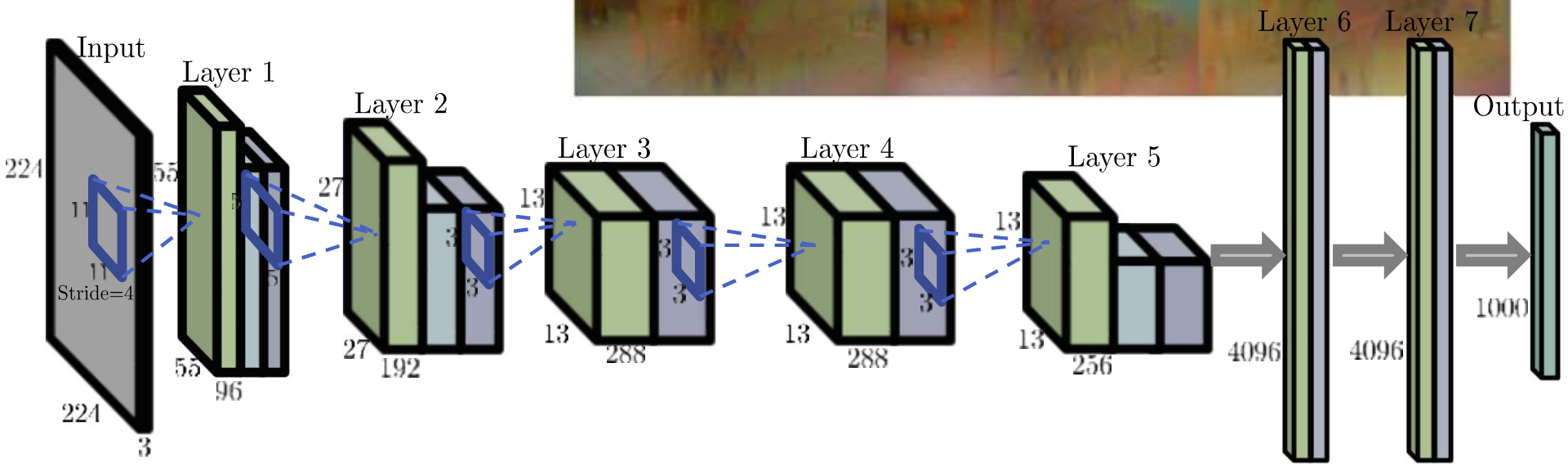
[Mahendrahan & Vedaldi CVPR15]

Синтез через подсчет прообразов

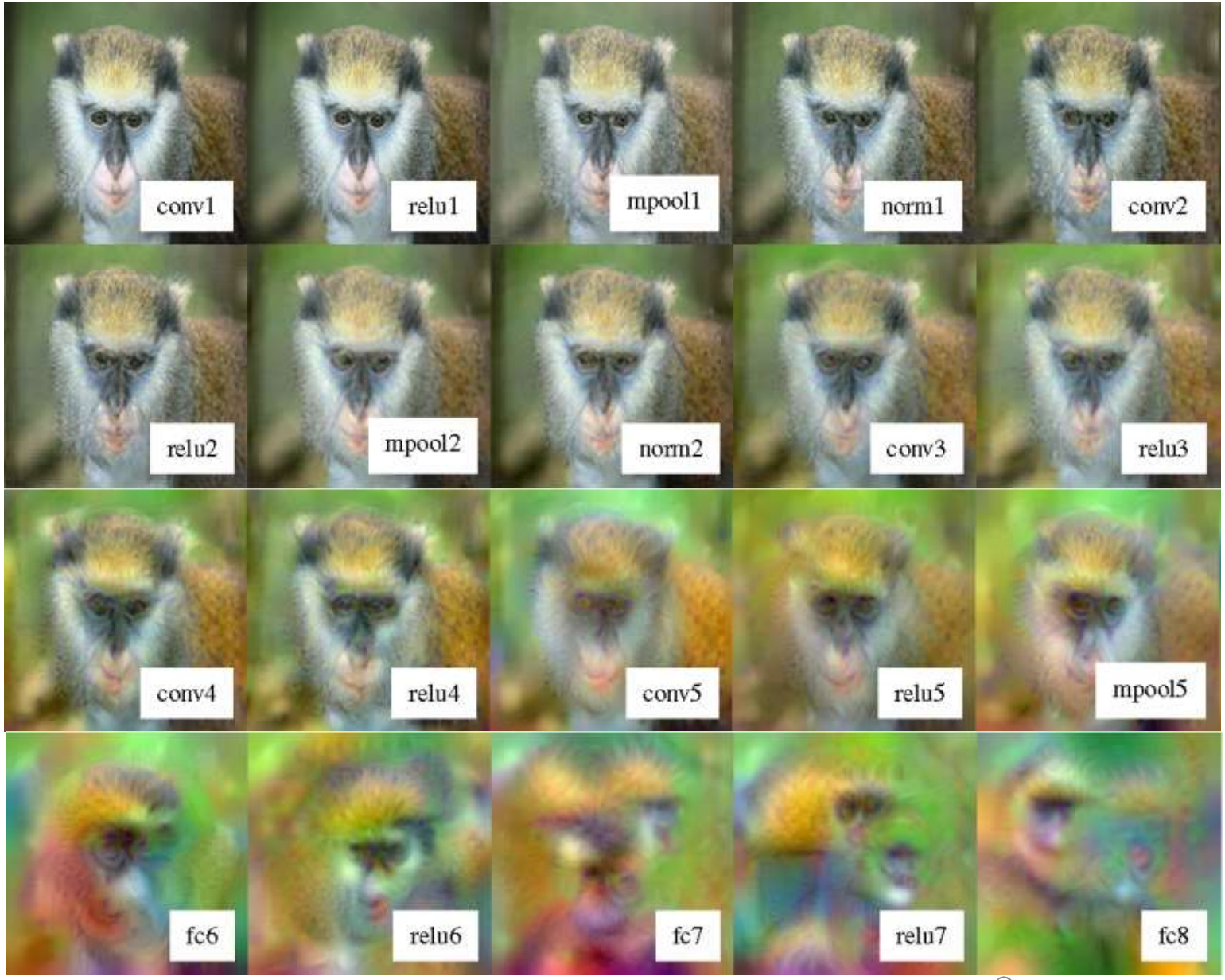
$$\hat{x} = \arg\min \|\Phi(x^*) - \Phi(x)\|^2 + \lambda R(x)$$

[Mahendrahan & Vedaldi CVPR15]





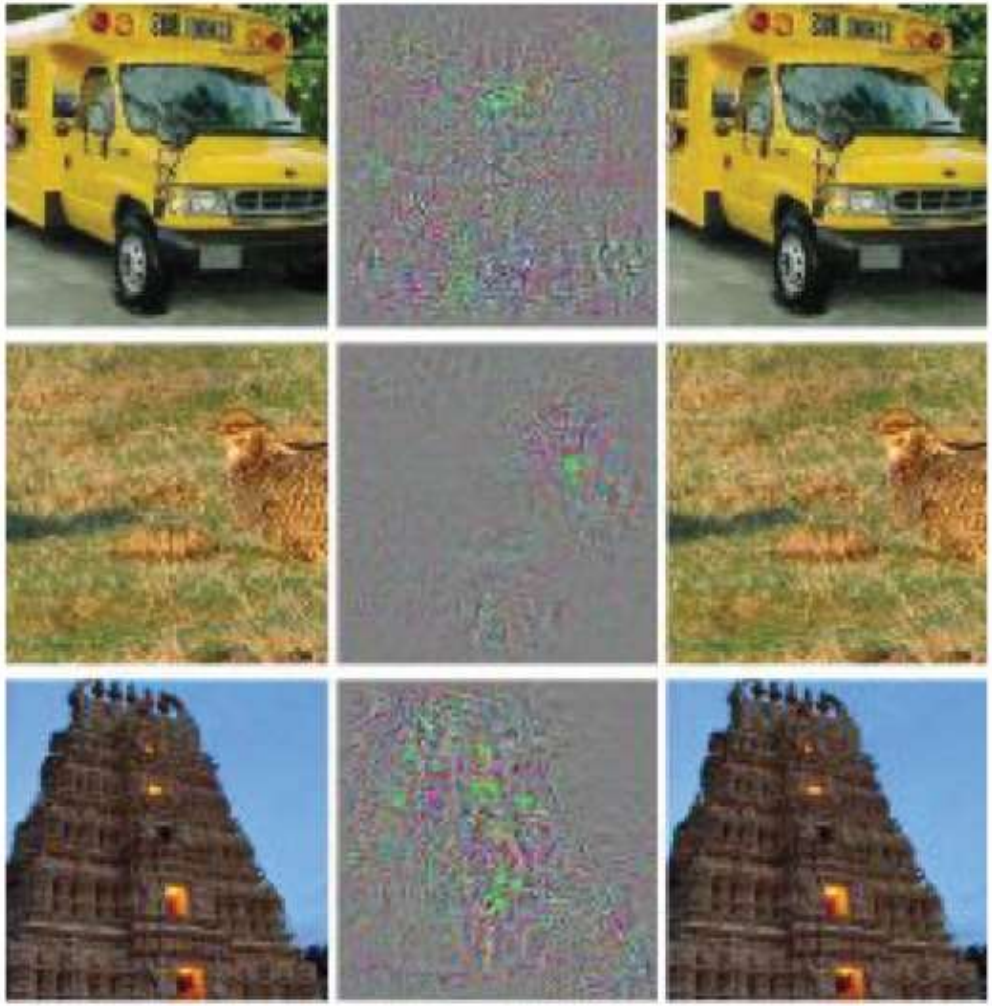
Generating images by Inverting CNNs

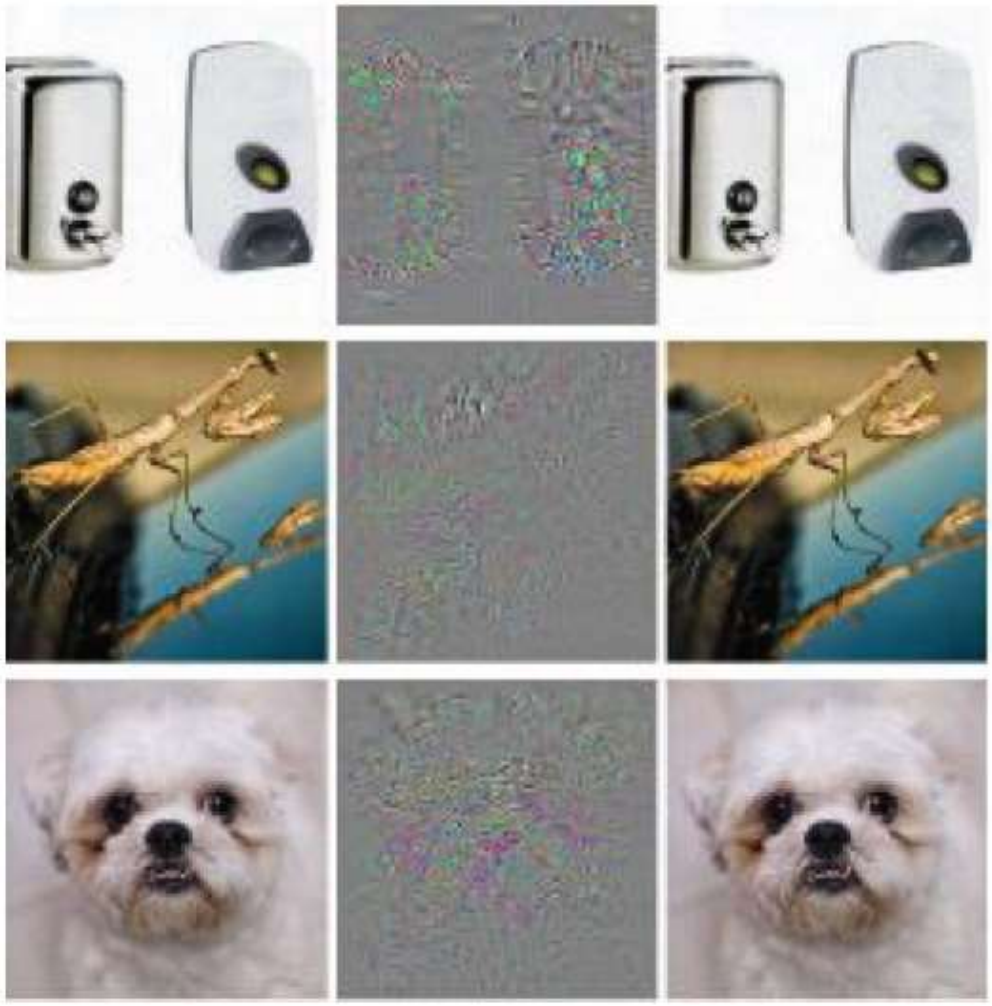


[Mahendrahan & Vedaldi CVPR15]

$$x^* = \arg\min \|\Phi(x^*) - \Phi(x)\|^2 + \lambda R(x)$$

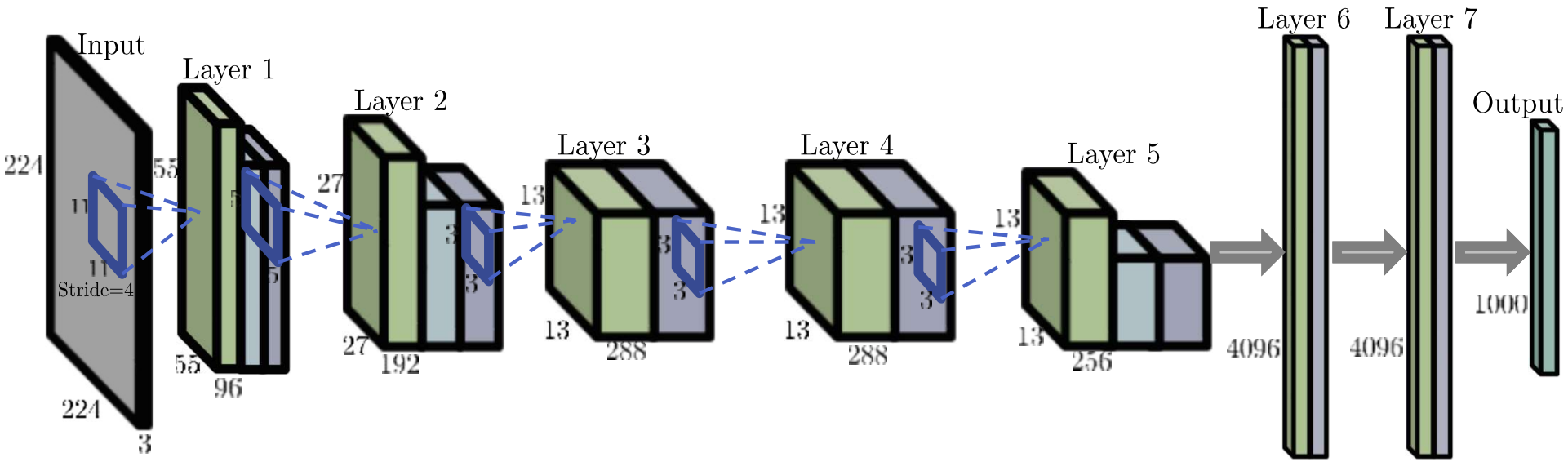
Синтез иллюзий для нейросети





[Szegedy, Zaremba, Sutskever, Bruna, Erhan, Goodfellow, Fergus 2014]

Синтез иллюзий для нейросети



$$\hat{r} = \arg\min \; c|r| + L(x+r, k)$$

Вывод: негладкость $\Phi(x)$

[Szegedy, Zaremba, Sutskever, Bruna, Erhan, Goodfellow, Fergus 2014]

Иллюзии биологических нейросетей

План

1. «Стандартные» свёрточные сети
2. **«Практическая» оптимизация**
3. Синтезирующие сети с простыми функциями потери
4. Синтез через повторение статистик
5. Играющие сети (adversarial networks)

Стохастический градиентный спуск

$$v[t] = -\alpha[t]\,\nabla(E, w[t])$$
$$w[t+1] = w[t] + v[t]$$

где $\nabla(E, w[t]) = \frac{dE^{c(t)}}{dw}\Big|_{w[t]}$

- i(t) обычно соответствует случайной перестановке данных
- Один проход по всем данным обычно называется *эпохой*

Stochastic gradient descent (SGD)

$$v[t] = -\alpha[t] \nabla(E, w[t])$$
$$w[t+1] = w[t] + v[t]$$

- Один проход по всем данным обычно называется *эпохой*
- Популярный выбор для α[t]:
  - константа, напр. α[t] = 0.0001
  - Кусочно-постоянный, напр. α[t] снижается в 10 раз каждые N эпох
  - гармоническое, напр. α[t] = 0.001 / ([t/N]+10)

Пакетный градиентный спуск

Градиент:

$$\frac{dE}{dw} = \frac{1}{N} \sum_{i=1}^{N} \frac{d\ell(x_i, y_i, w)}{dw}$$

Пакет (батч, минибатч):

$$\{b_1, b_2 .. b_{N_b}\} \subset 1 .. N$$

Пакетный градиентный спуск:

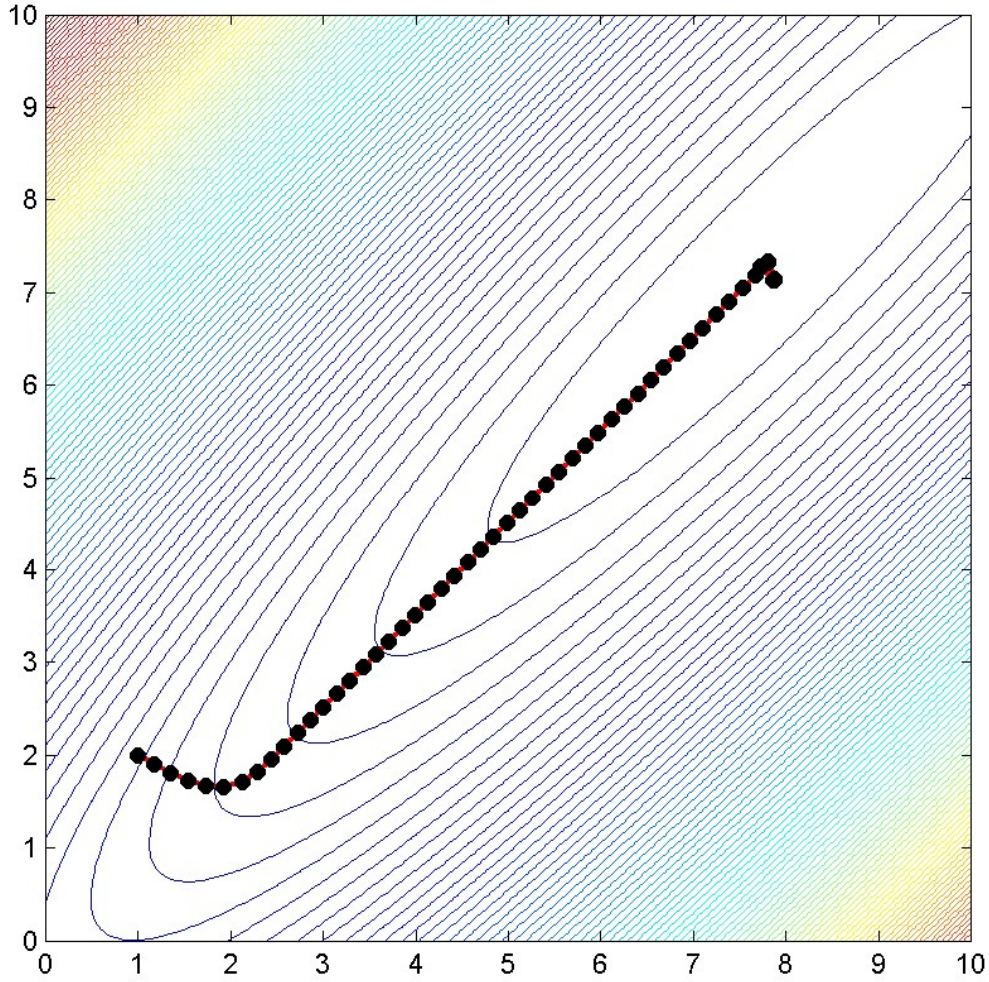
$$\frac{dE}{dw} = \frac{1}{N_b} \sum_{i=1}^{N_b} \frac{d\ell(x_{b(i)}, y_{b(i)}, w)}{dw}$$

Причины для пакетирования?

$$\frac{dE}{dw} = \frac{1}{N_b} \sum_{i=1}^{N_b} \frac{d\ell(x_{b(i)}, y_{b(i)}, w)}{dw}$$

- Менее шумная аппроксимация, «лучше» сходимость
- **Главная причина:** параллелизм современных архитектур

Проблемы стохастического спуска



- ..примерно те же, что у градиентного спуска

Улучшение СГС с помощью момента

- …напоминает сопряженные градиенты (комбинируем градиент и предыдущее направление):

$$v[t] = -\alpha[t]\,\nabla(E, w[t])$$
$$w[t+1] = w[t] + v[t]$$

$$v[t+1] = \mu\, v[t] - \alpha[t]\,\nabla(E, w[t])$$
$$w[t+1] = w[t] + v[t+1]$$

Типичное значение $\mu = 0.9$

Скользящее экспоненициальное окно

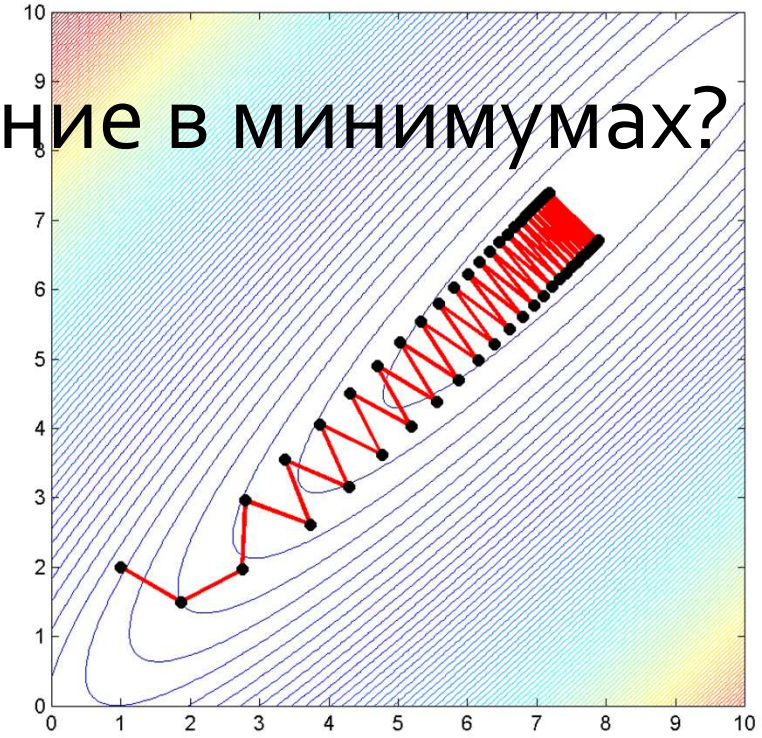
$$v[t+1] = \mu\, v[t] - \alpha[t]\, \nabla\,(E, w[t])$$
$$w[t+1] = w[t] + v[t+1]$$

$$v[t+1] = \mu\, v[t] - \alpha[t]\, \nabla\,(E, w[t]) =$$
$$= \mu^2\, v[t-1] - \mu\alpha[t-1]\, \nabla\,(E, w[t-1])$$
$$- \alpha[t]\, \nabla\,(f, w[t]) =$$
$$= \mu^3\, v[t-2] - \mu^2\, \alpha[t-2]\, \nabla\,(E, w[t-2])$$
$$- \mu\alpha[t-1]\, \nabla\,(E, w[t-1]) - \alpha[t]\, \nabla\,(E, w[t]) =$$
$$= \mu^{k+1}\, v[t-k] + \sum_{i=0}^{k} \mu^i\, \alpha[t-i]\, \nabla\,(E, w[t-i])$$

Причины улучшения

$$v[t+1] = \sum_{i=0}^{k} \mu^i \alpha[t-i] \nabla(E, w[t-i])$$

- Сглаживание шума СГС (~увеличение пакетов)
- **Сглаживание осцилляций, присущих ГС**
- Предотвращает застревание в минимумах?



Метод Нестерова

$$v[t+1] = \mu\, v[t] - \alpha[t]\, \nabla(E, w[t])$$
$$w[t+1] = w[t] + v[t+1]$$

Еще до подсчета градиента знаем, что: $w[t+1] \approx w[t] + \mu\, v[t]$

Считаем градиент в более «актуальном» месте:

$$v[t+1] = \mu\, v[t] - \alpha[t]\, \nabla(E, w[t] + \mu\, v[t])$$
$$w[t+1] = w[t] + v[t+1]$$

Adagrad method [Duchi et al. 2011]

Идея: корректировать движение по градиенту

$$g[t+1] = g[t] + \nabla(E, w[t]) \odot \nabla(E, w[t])$$

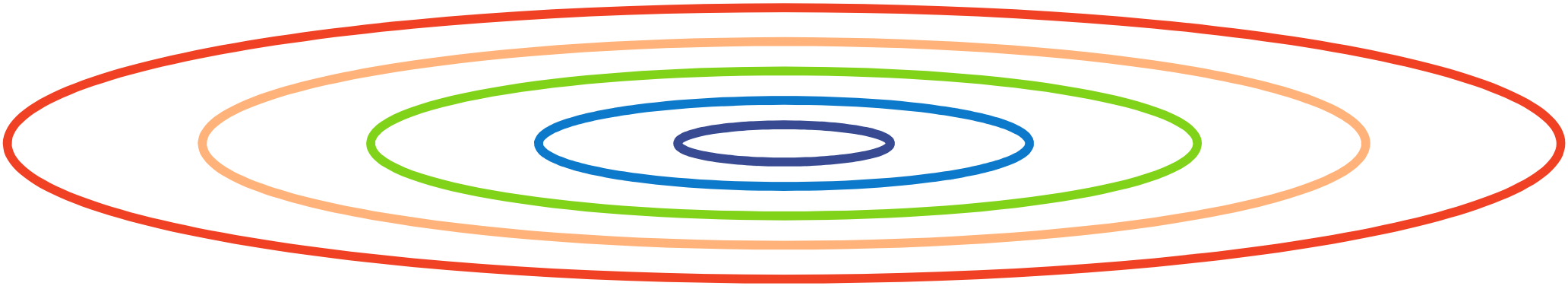
$$w[t+1] = w[t] - \frac{\alpha}{\sqrt{g[t] + \varepsilon}} \odot \nabla(E, w[t])$$

(Полу)-автоматическая подстройка скорости (уменьшение со временем)

Adagrad method [Duchi et al. 2011]

$$g[t+1] = g[t] + \nabla(E, w[t]) \odot \nabla(E, w[t])$$

$$w[t+1] = w[t] - \frac{\alpha}{\sqrt{g[t] + \varepsilon}} \odot \nabla(E, w[t])$$



RMSPROP [Hinton 2012]

Схоже с Adagrad, но накопление заменяется экспоненциальным окном:

$$g[t+1] = \mu\, g[t] + (1-\mu)\, \nabla(E, w[t]) \odot \nabla(E, w[t])$$

$$w[t+1] = w[t] - \frac{\alpha[t]}{\sqrt{g[t] + \varepsilon}} \odot \nabla(E, w[t])$$

Adadelta [Zeiler 2012]

Идея: ускорение вдоль «монотонно» улучшающихся измерений

$$g[t+1] = \mu\, g[t] + (1-\mu)\, \nabla(E, w[t]) \odot \nabla(E, w[t])$$

$$w[t+1] = w[t] - \frac{\sqrt{d[t] + \varepsilon}}{\sqrt{g[t] + \varepsilon}} \odot \nabla(E, w[t])$$

$$d[t+1] = \mu\, d[t] + (1-\mu)\, (w[t+1]-w[t]) \odot (w[t+1]-w[t])$$

Adadelta [Zeiler 2012]

$$g[t+1] = \mu\, g[t] + (1-\mu)\, \nabla(E, w[t]) \odot \nabla(E, w[t])$$

$$w[t+1] = w[t] - \frac{\sqrt{d[t] + \varepsilon}}{\sqrt{g[t] + \varepsilon}} \odot \nabla(E, w[t])$$

$$d[t+1] = \mu\, d[t] + (1-\mu)\, (w[t+1]-w[t]) \odot (w[t+1]-w[t])$$

- Полезное свойство: правильные размерности

ADAM [Kingma & Ba 2015]

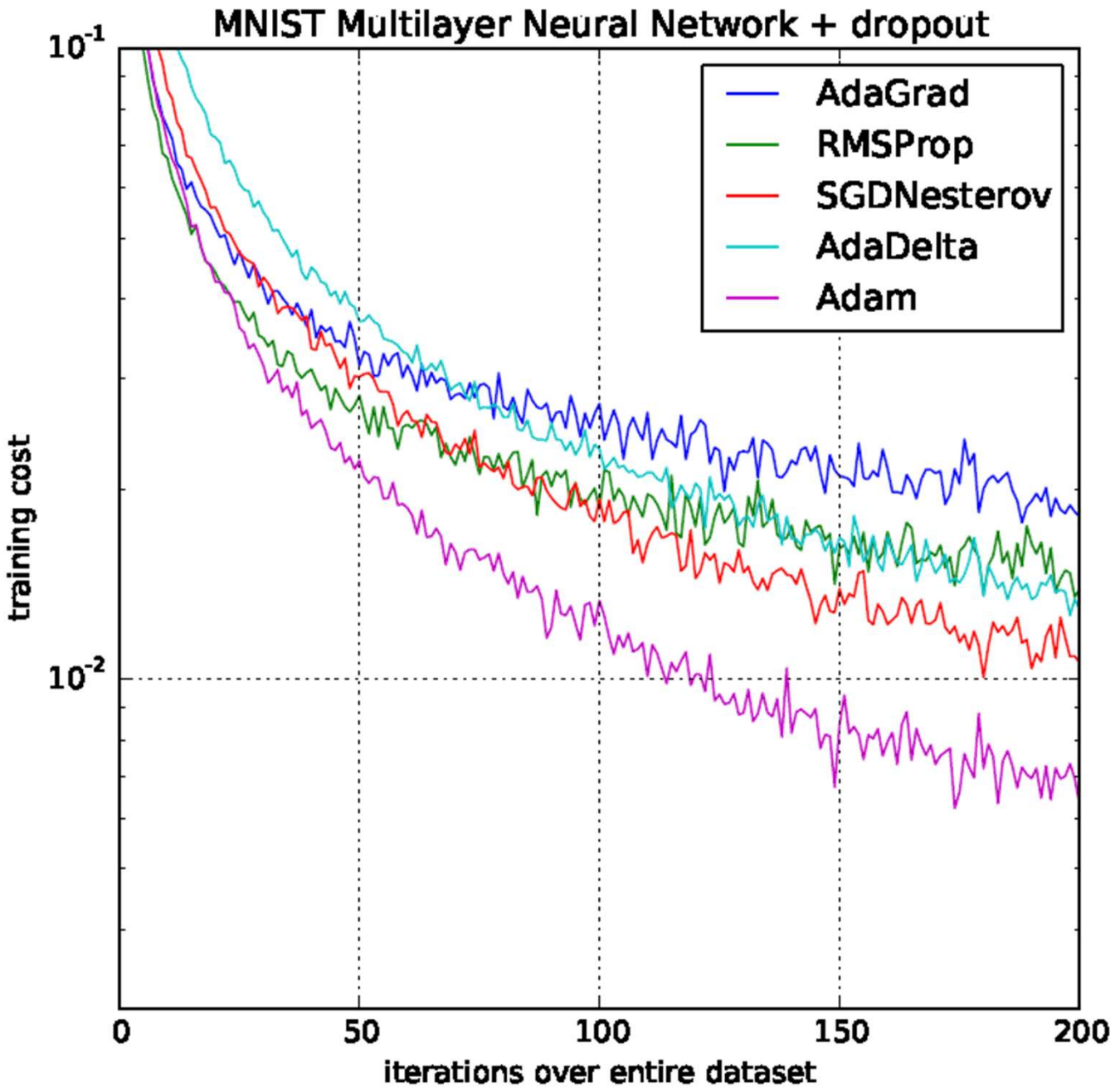
ADAM = "ADAptive Moment Estimation"

$$v[t+1] = \beta\, v[t] + (1-\beta)\, \nabla(E, w[t])$$

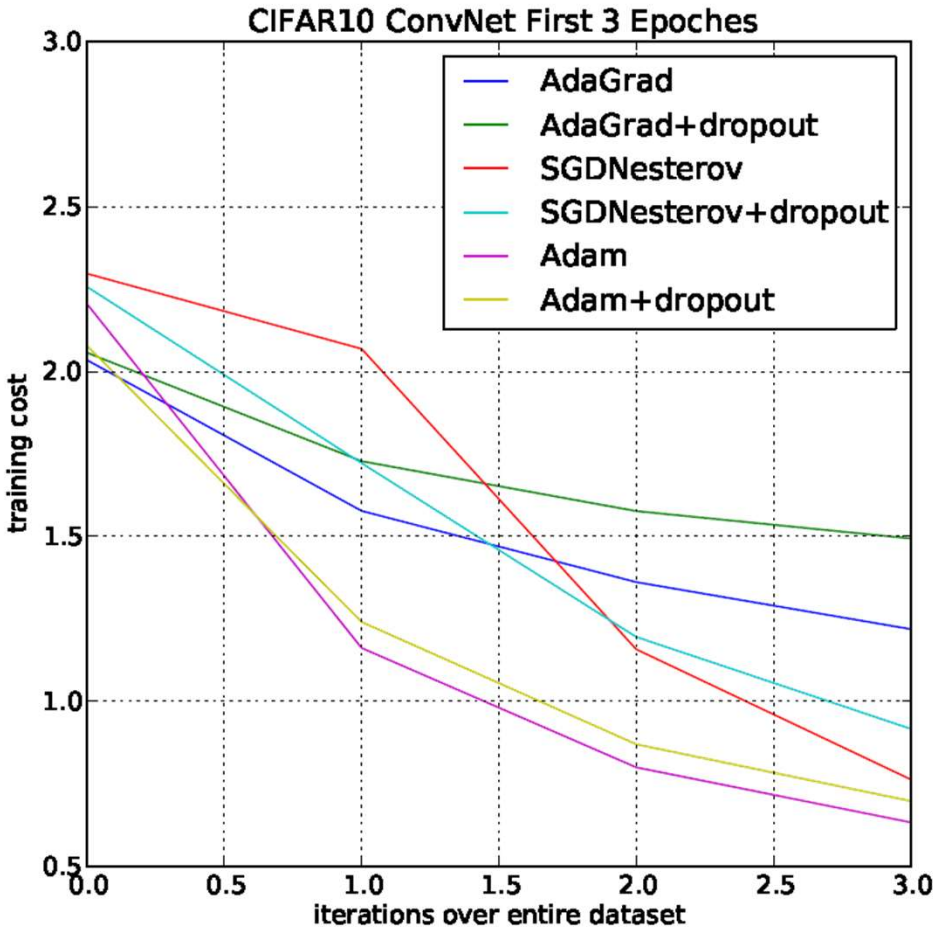
$$g[t+1] = \mu\, g[t] + (1-\mu)\nabla(E, w[t]) \odot \nabla(E, w[t])$$

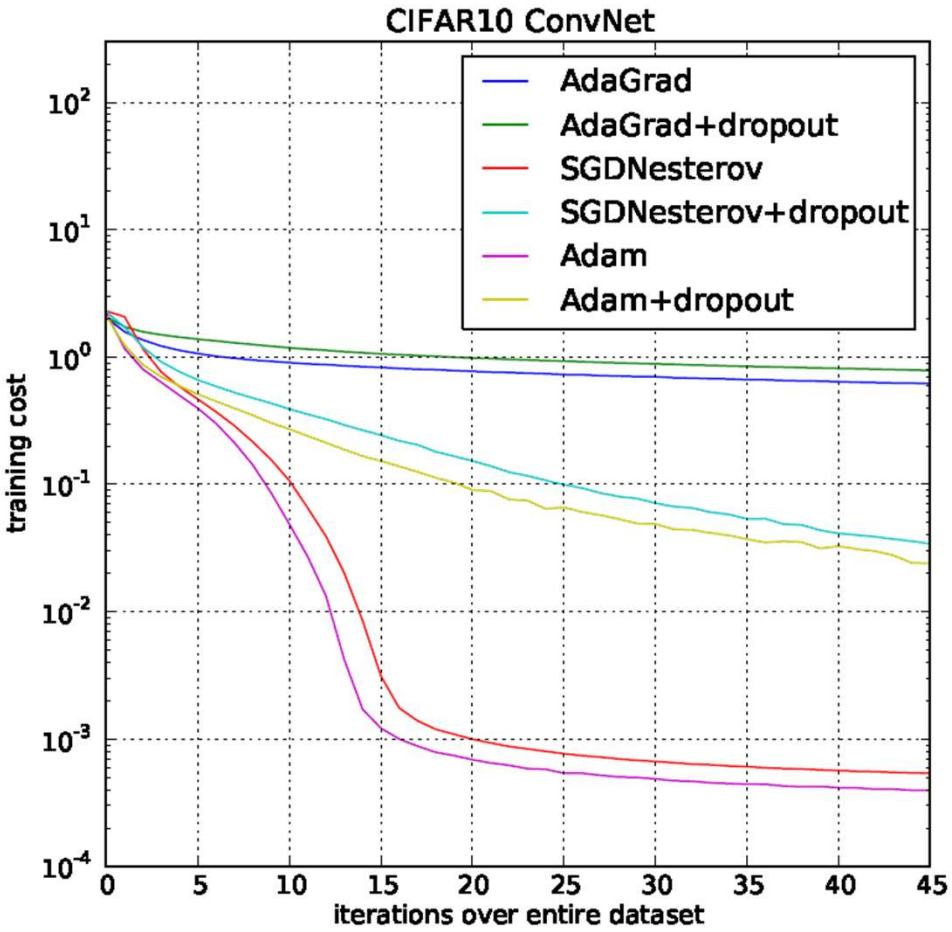
$$w[t+1] = w[t] - \alpha[t] \frac{1}{1-\beta} \frac{1}{\sqrt{g[t]} + \varepsilon} \odot v[t]$$

ADAM [Kingma & Ba 2015]



ADAM [Kingma & Ba 2015]





Сравнение: логистическая регрессия

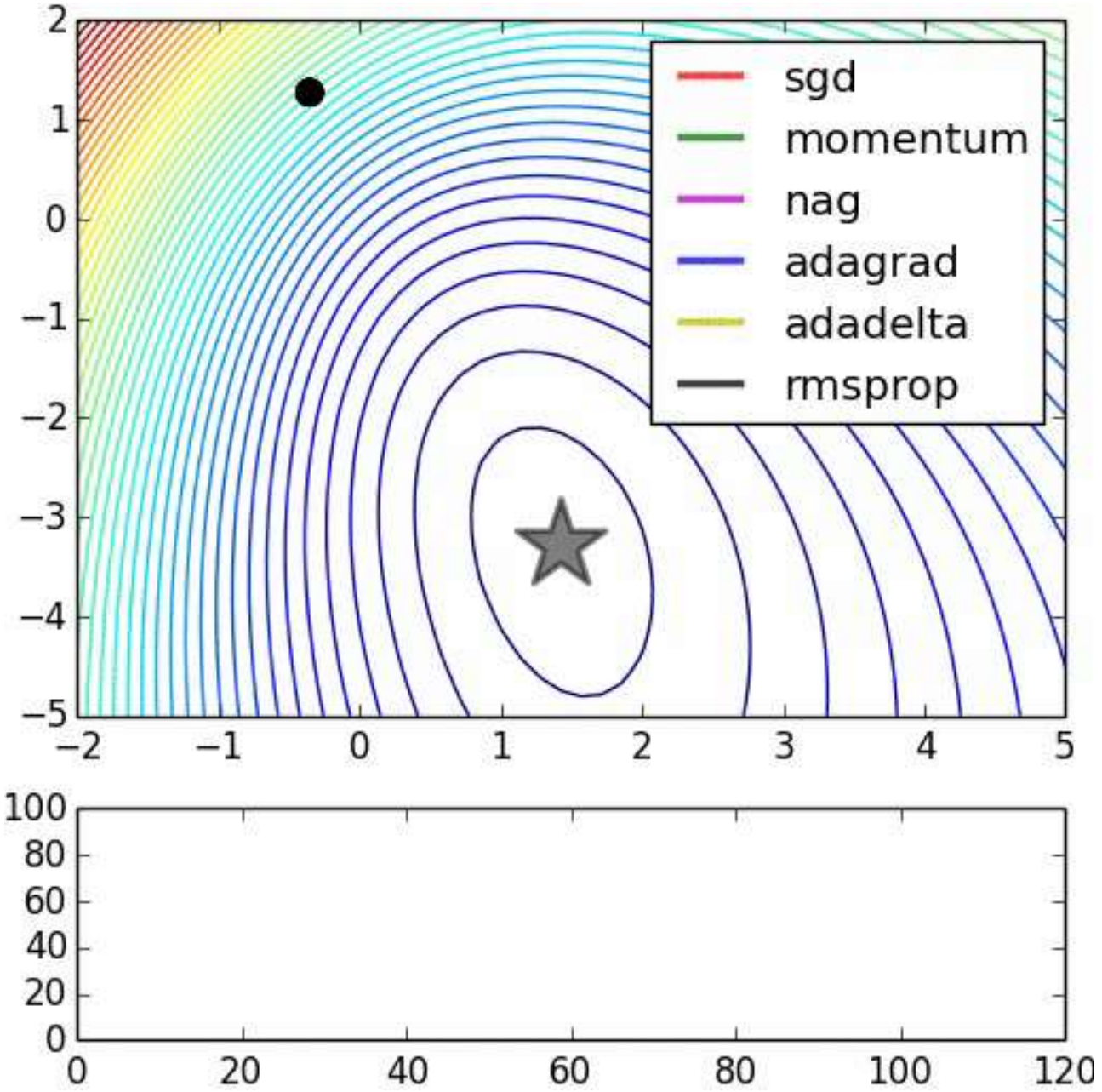


Image credit: Alec Redford

Функция Бигля

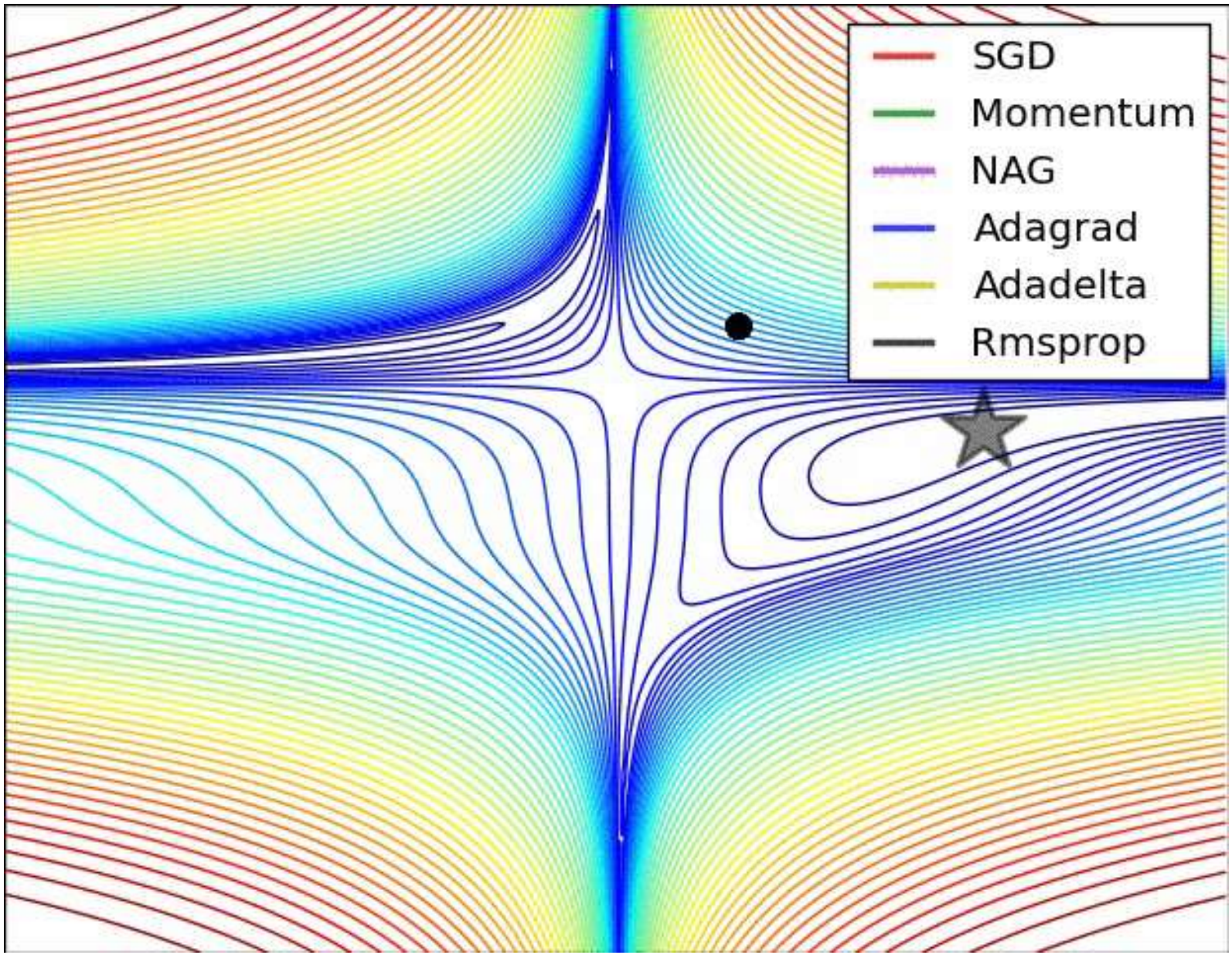


Image credit: Alec Redford

Уход из седловой точки

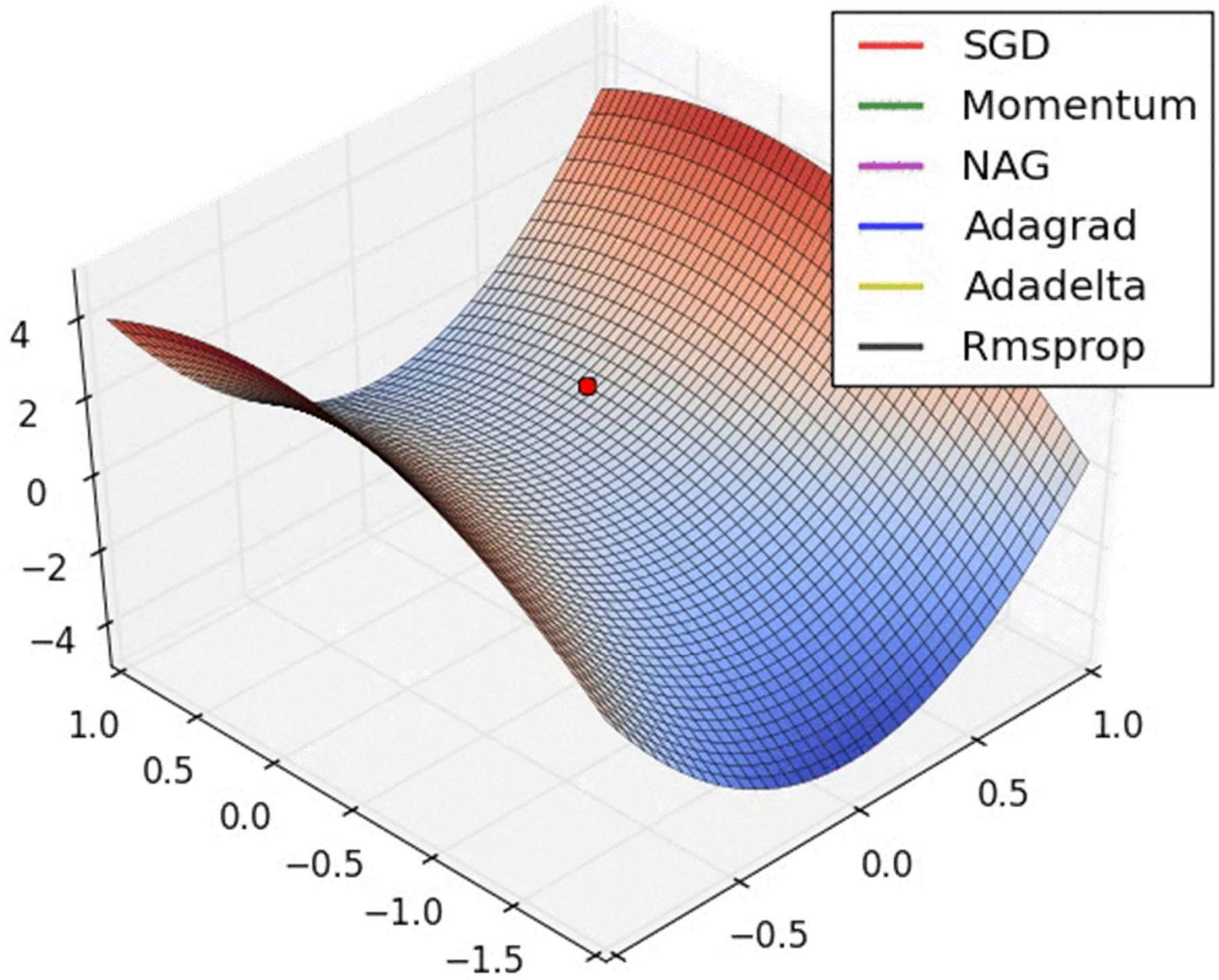


Image credit: Alec Redford

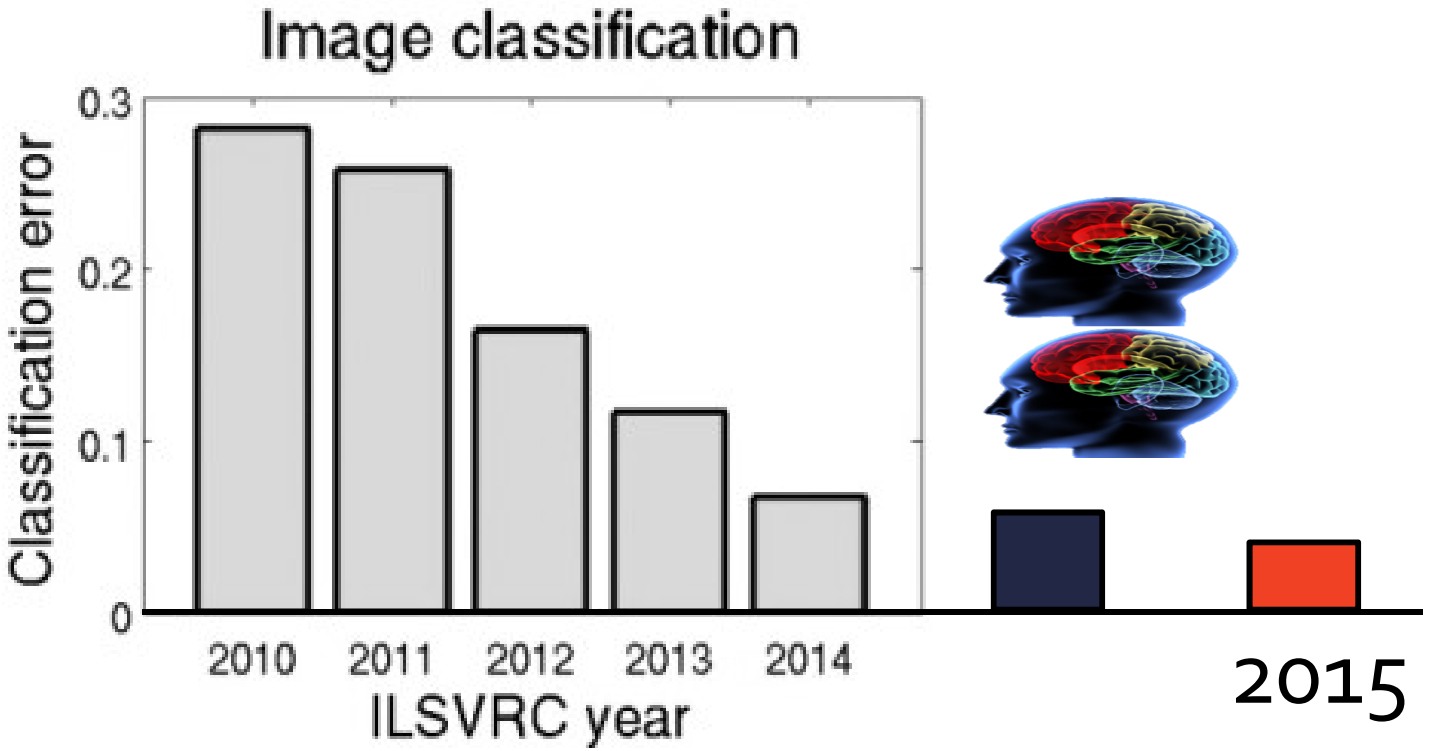
Current state

- Выбор оптимизатора и параметров по-прежнему нетривиально (обычно начинаем с ADAM)
- Существенное улучшение за счет слоёв, которые улучшают/обуславливают градиенты и активации (batch normalization, weight normalization, layer normalization…)

План

1. «Стандартные» свёрточные сети
2. «Практическая» оптимизация
3. **Синтезирующие сети с простыми функциями потери**
4. Синтез через повторение статистик
5. Играющие сети (adversarial networks)

Текущее состояние



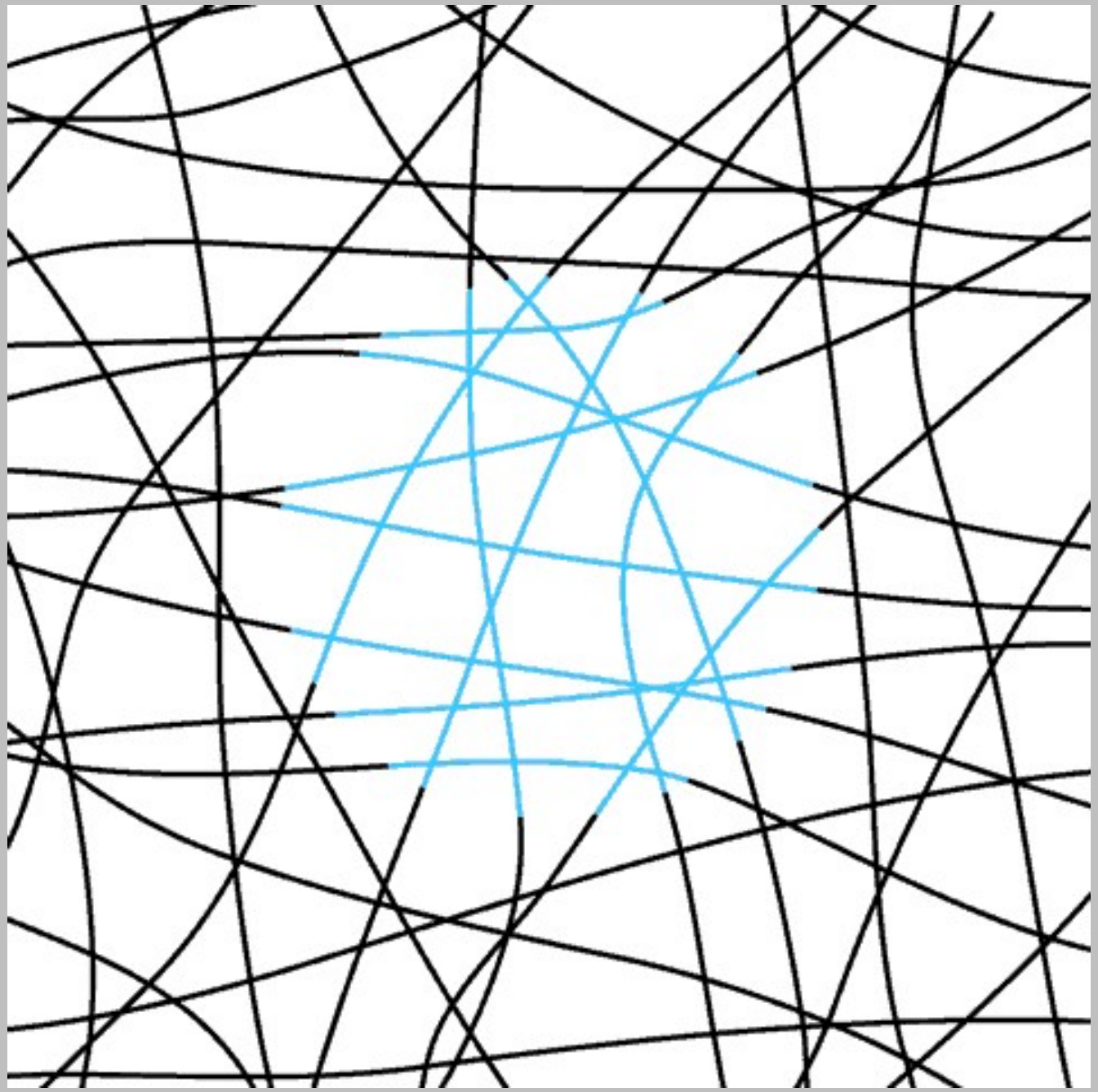
- Практически достигнут уровень мозга в задаче классификации (изображение -> метка класса)
- Метка класса -> изображение = ?

Зачем нужно синтезировать изображения?

Зачем нужно синтезировать изображения с помощью нейросетей:

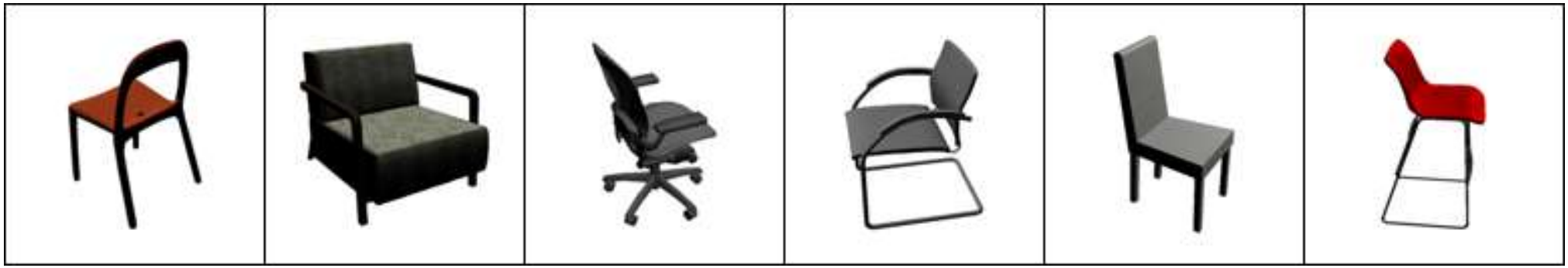
1. Fun
2. Понимание как (не) работают глубокие нейросети
3. Компьютерная графика
4. Редактирование изображений
5. Чтобы улучшить компьютерное зрение (через *порождающие модели*)

Порождающая модель в мозгу



«Разворот» сверточных сетей

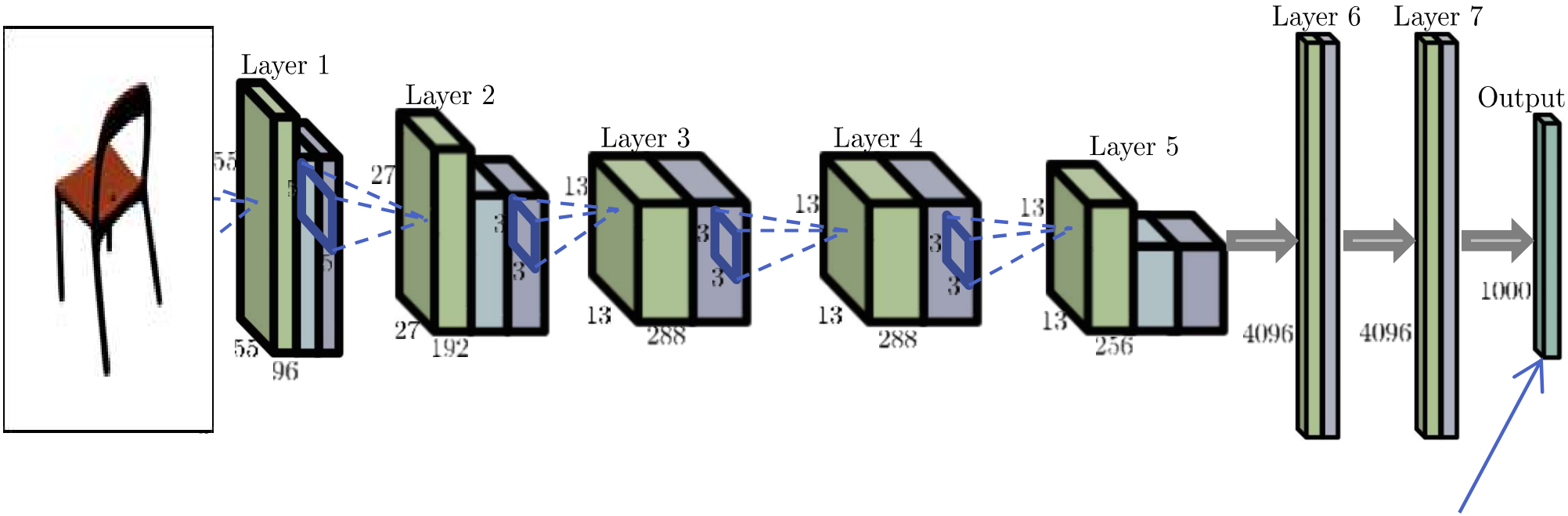
Исходные изображения:



Каждое изображение аннотировано типом и углом зрения

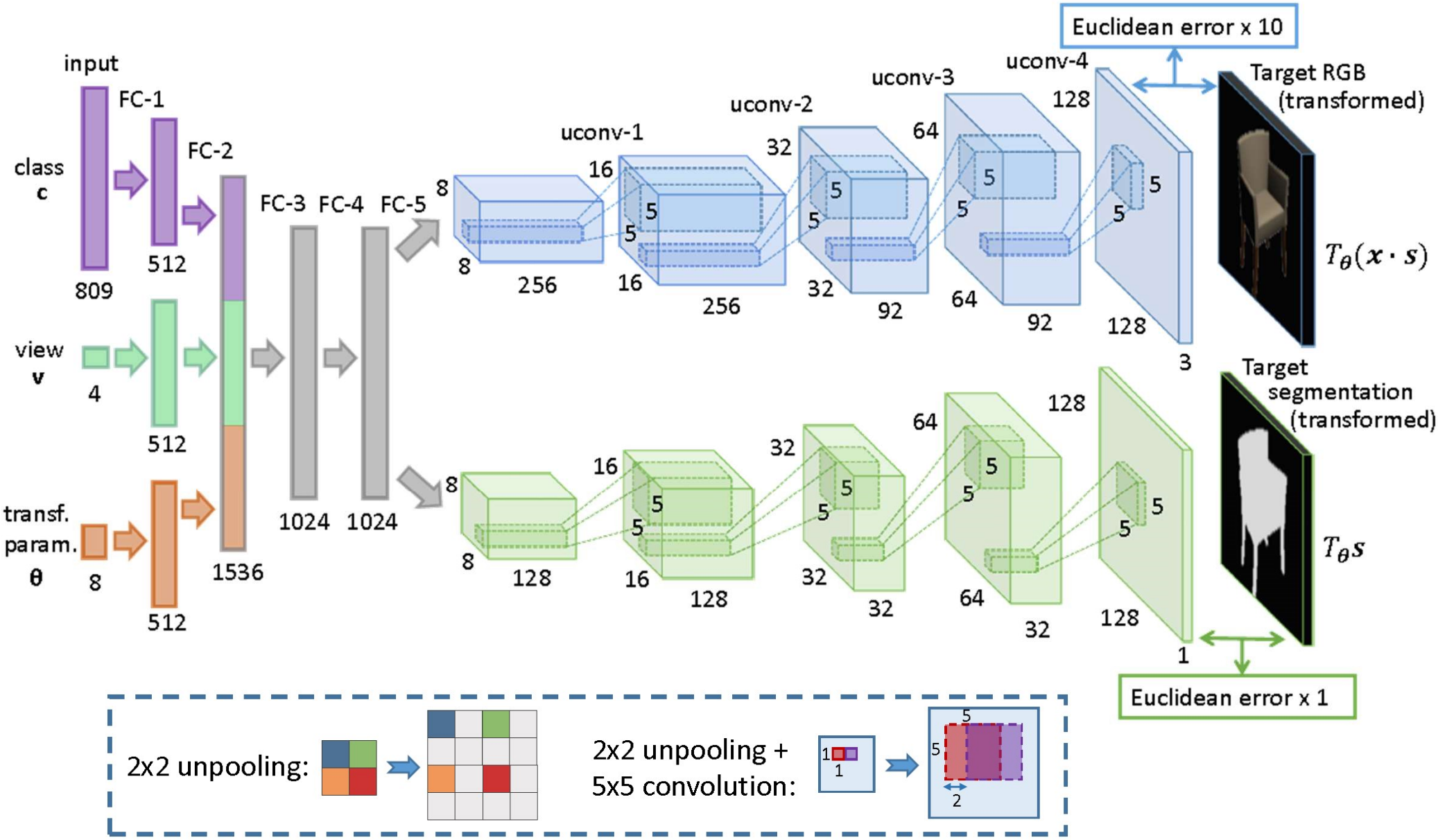
[Dosovitskiy, Springenberg, Brox CVPR 2015]

Сеть до разворота



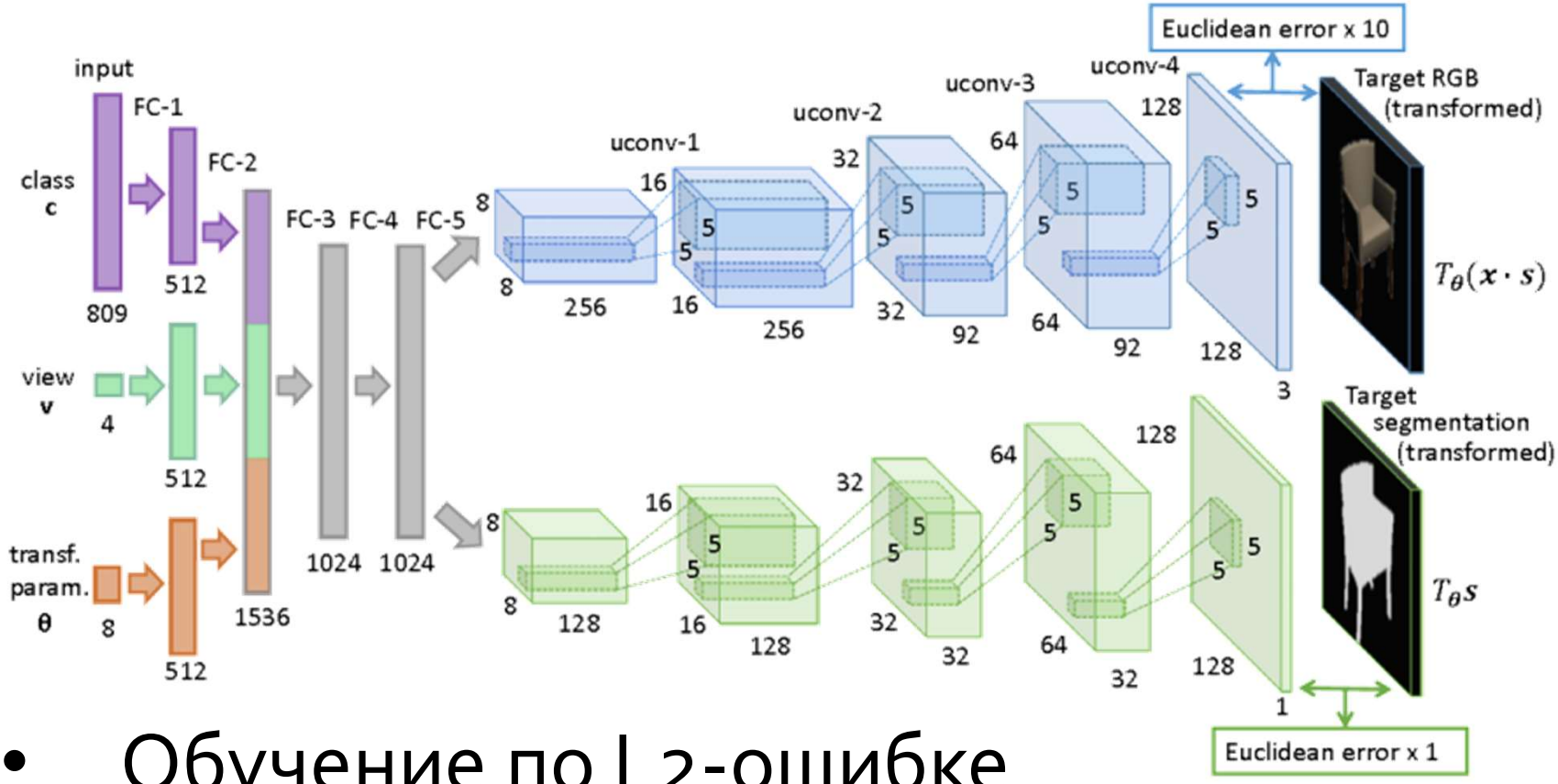
тип стула
угол зрения
и т.п.

Развернутая сеть



[Dosovitskiy, Springenberg, Brox CVPR 2015]

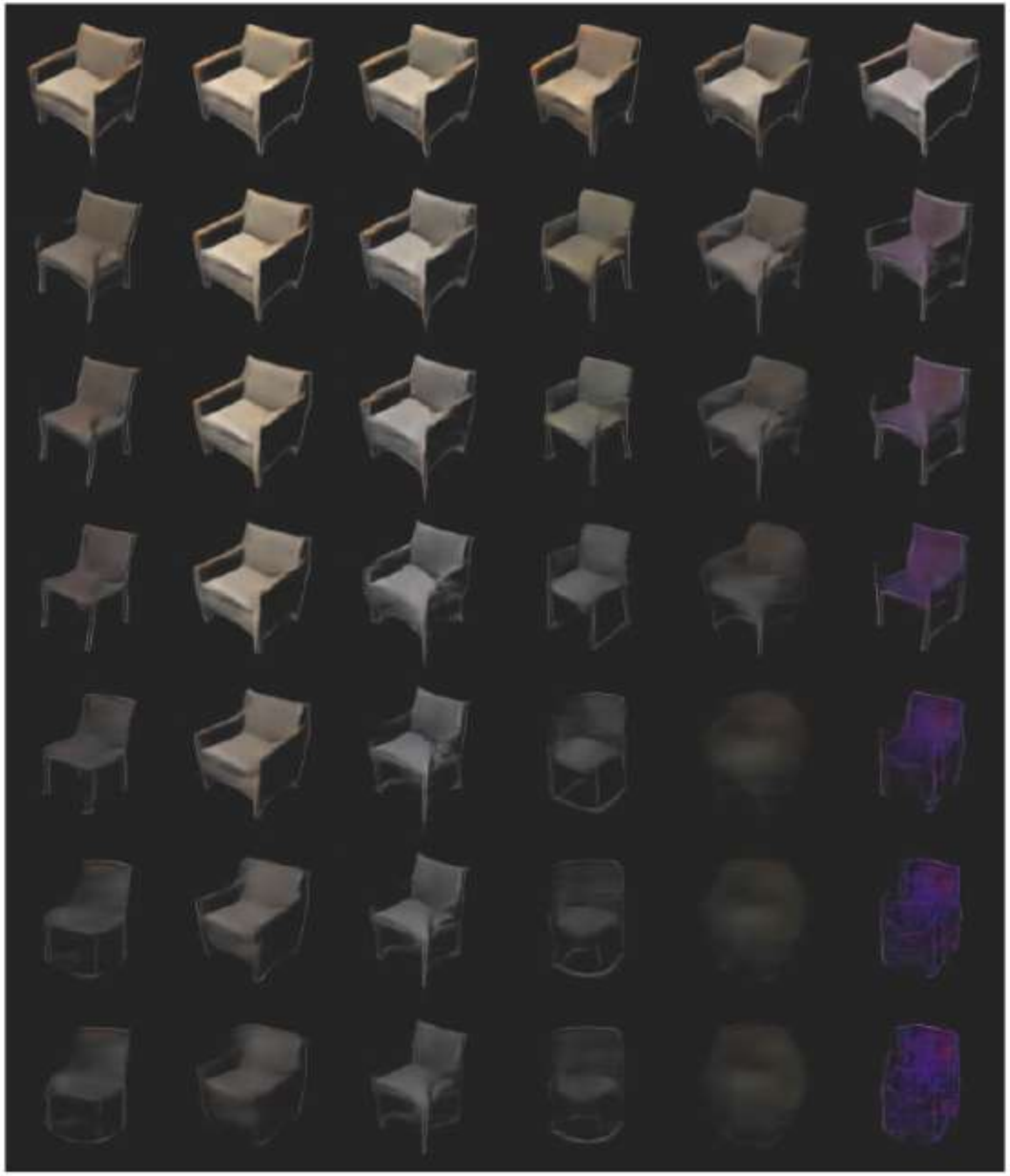
Развернутая сеть

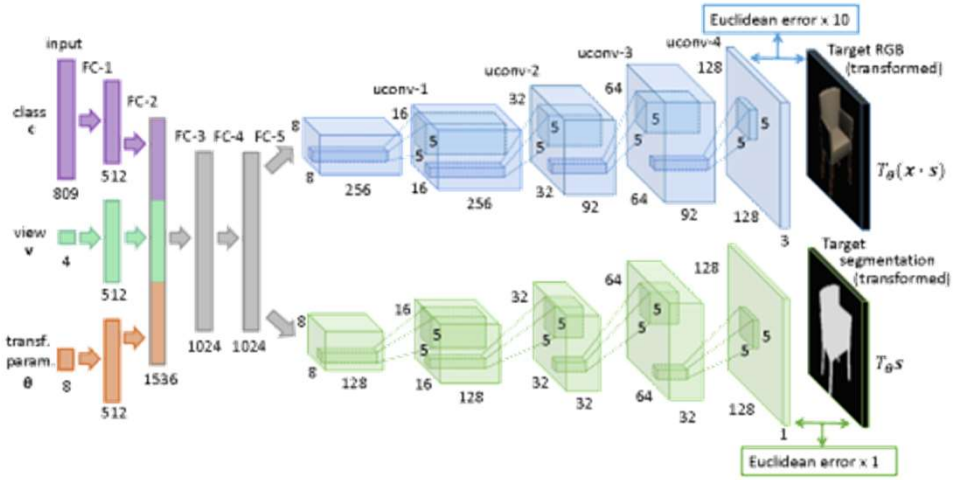


- Обучение по L2-ошибке
- «Размножение» тренировочных данных
- Обучение маски даёт дополнительную регуляризацию

[Dosovitskiy, Springenberg, Brox CVPR 2015]

Развернутая сеть

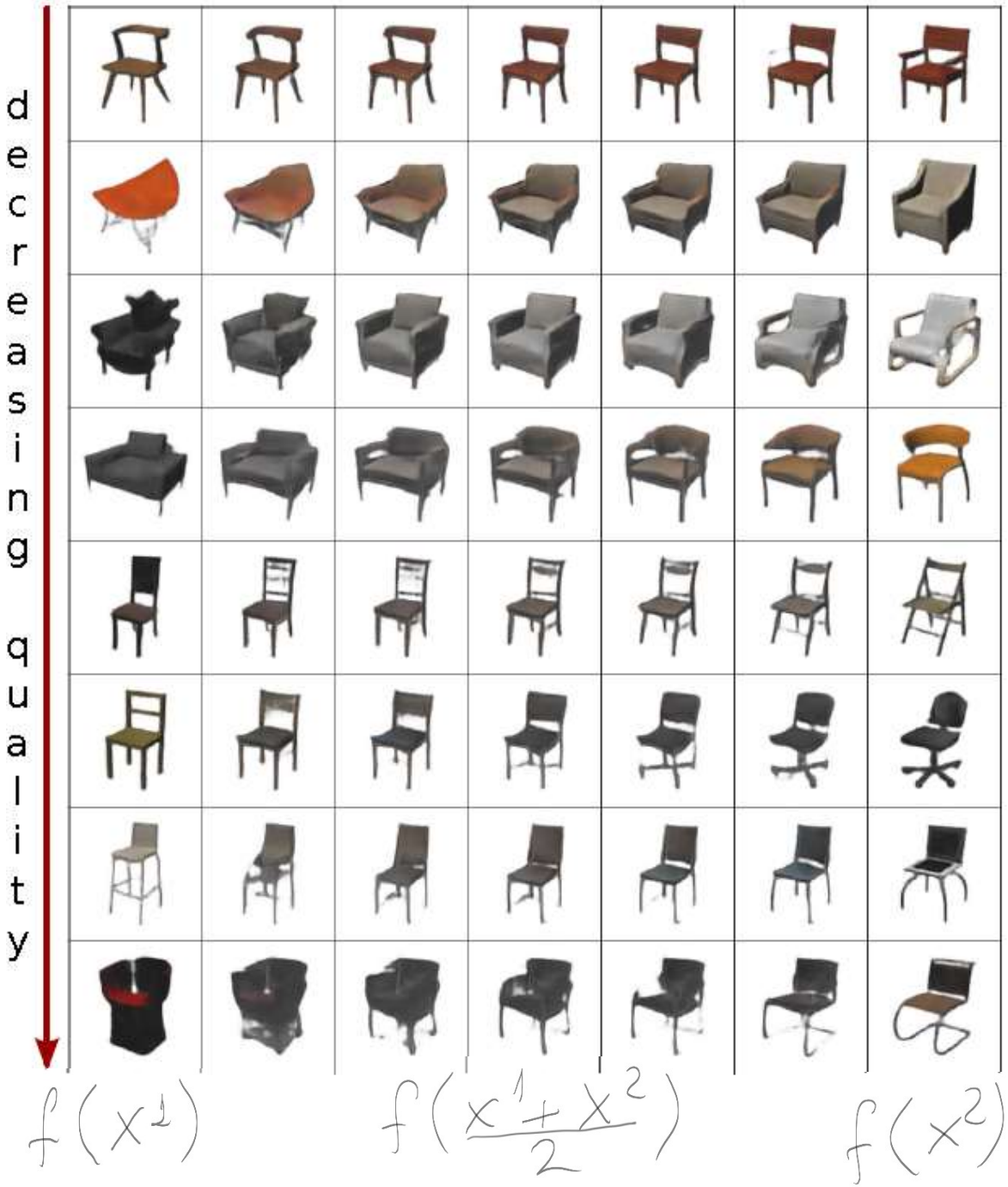


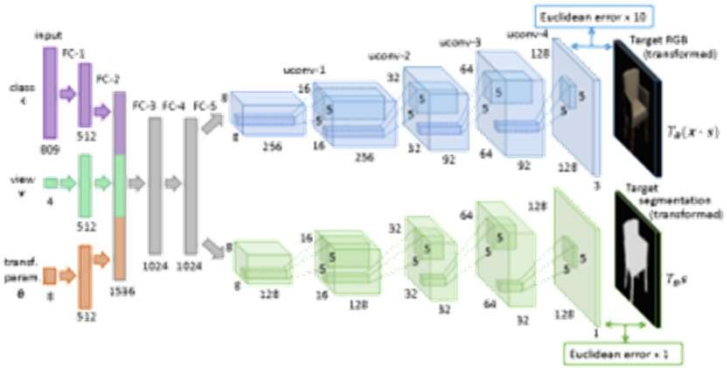


Изменение одного «нейрона»

[Dosovitskiy, Springenberg, Brox CVPR 2015]

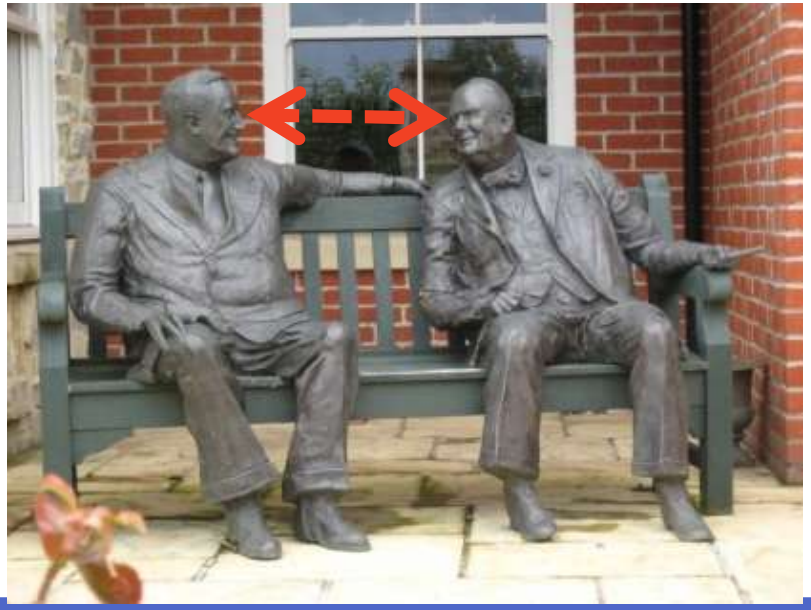
Развернутая сеть

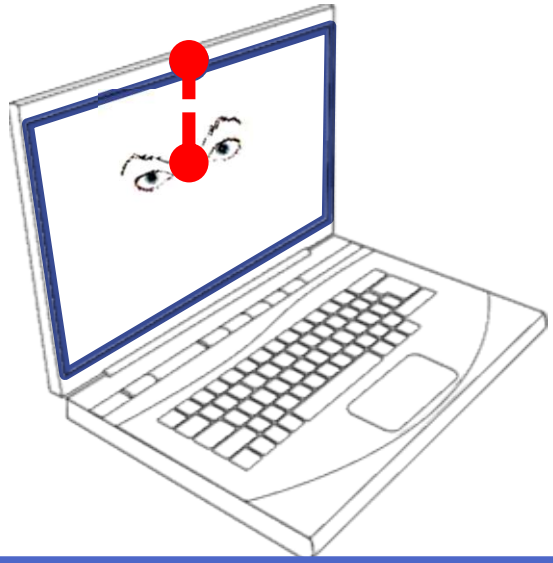


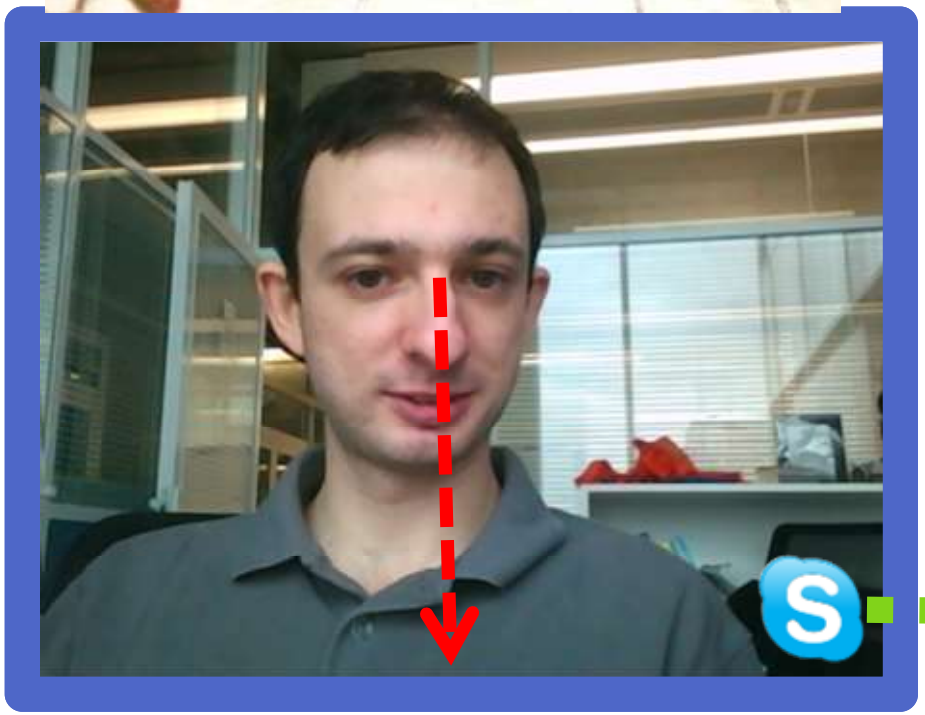


[Dosovitskiy, Springenberg, Brox CVPR 2015]

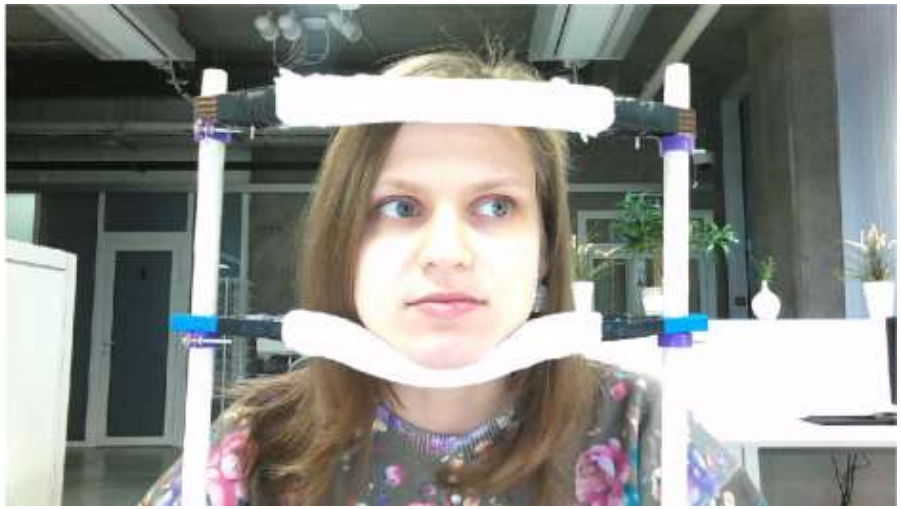
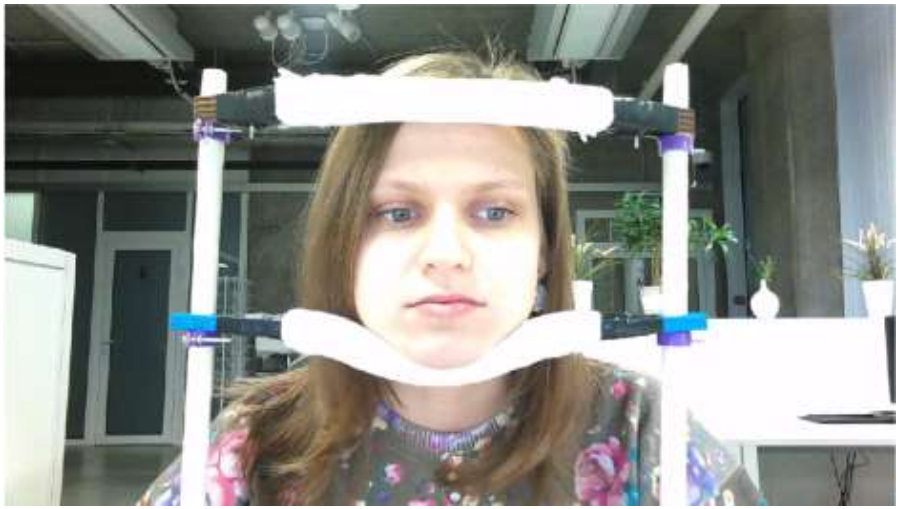
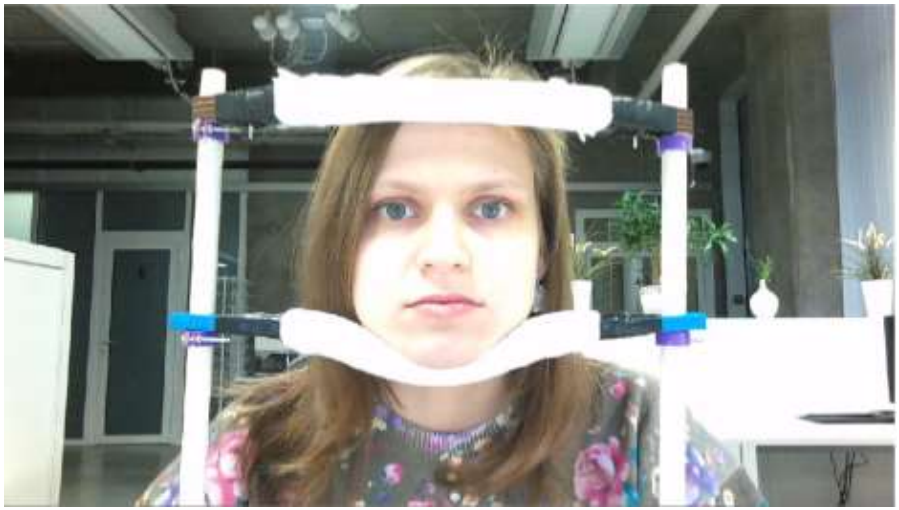
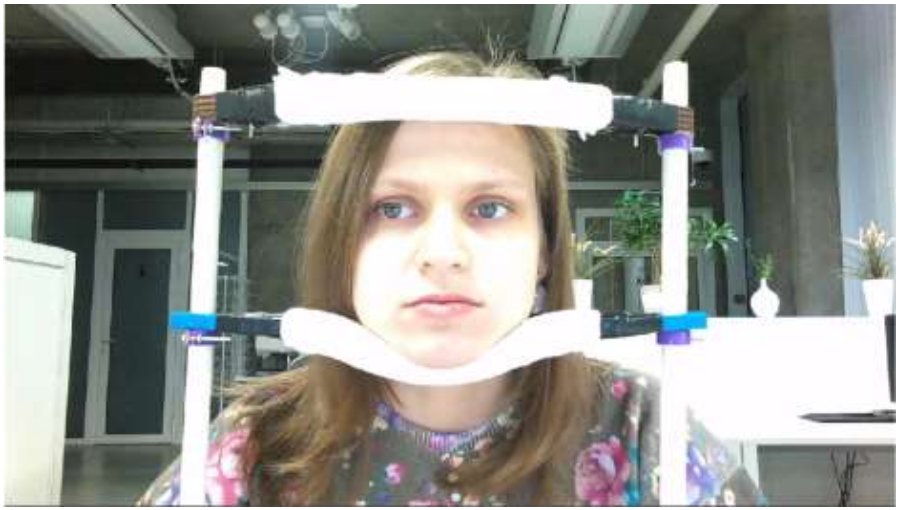
Практическое применение: gaze correction



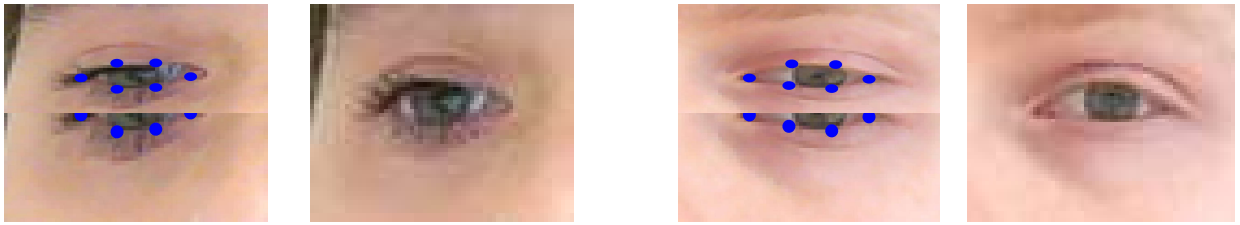




Обучение с учителем – вещь тяжелая

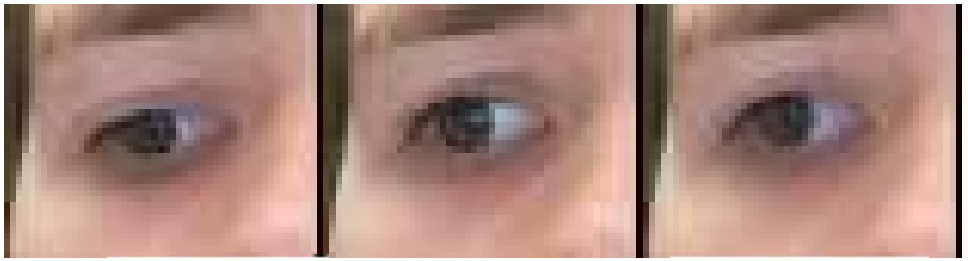





Редактирование через поля смещений



in out in out

Наблюдение: значительная часть преобразования может быть задана *полем смещений*:



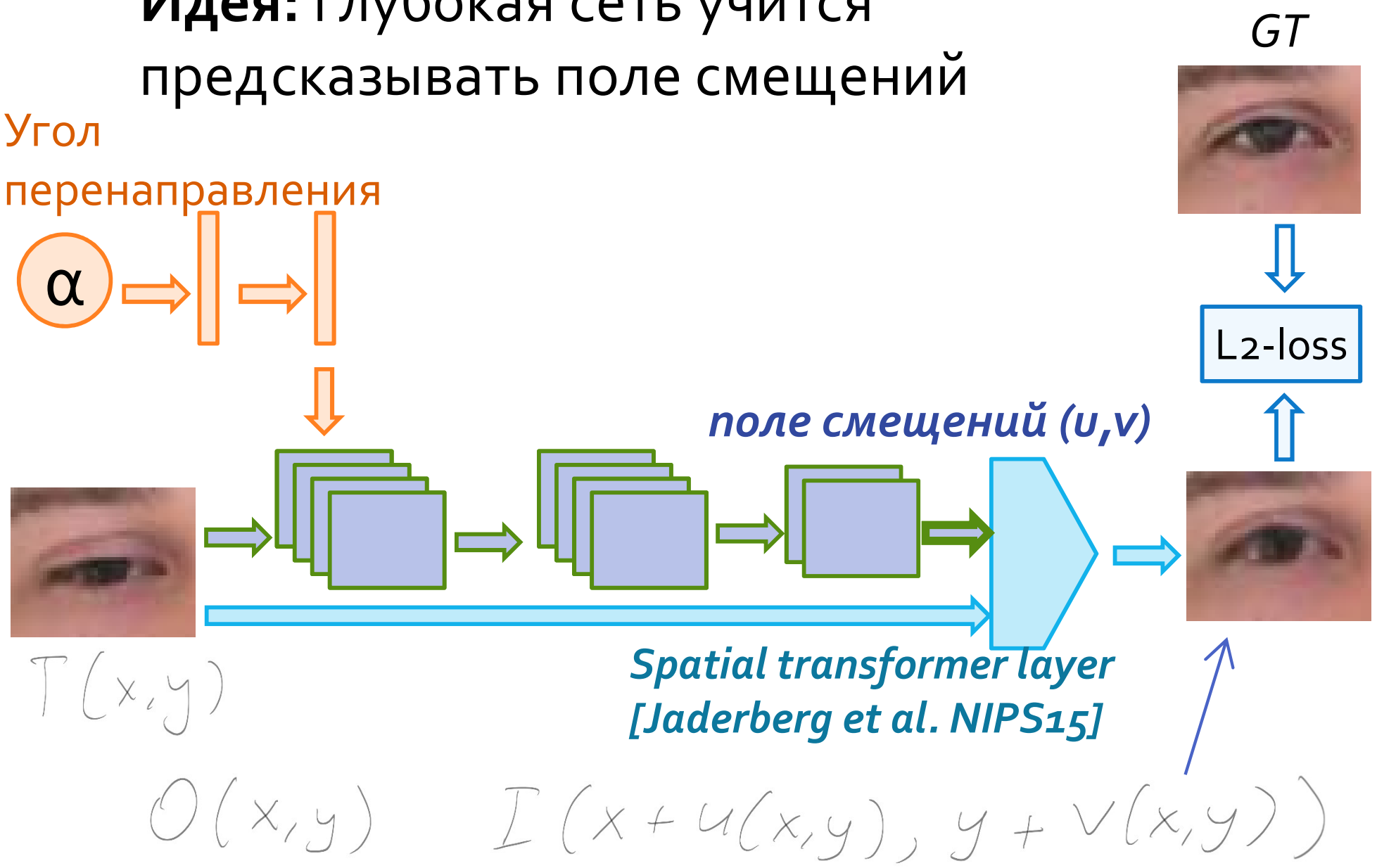
$I(x,y)$ GT $O(x,y)$ $[u(x,y)\ v(x,y)]$

$O(x,y) \quad I(x+u(x,y),\ y+v(x,y))$

Поля смещений хорошо обобщаются!

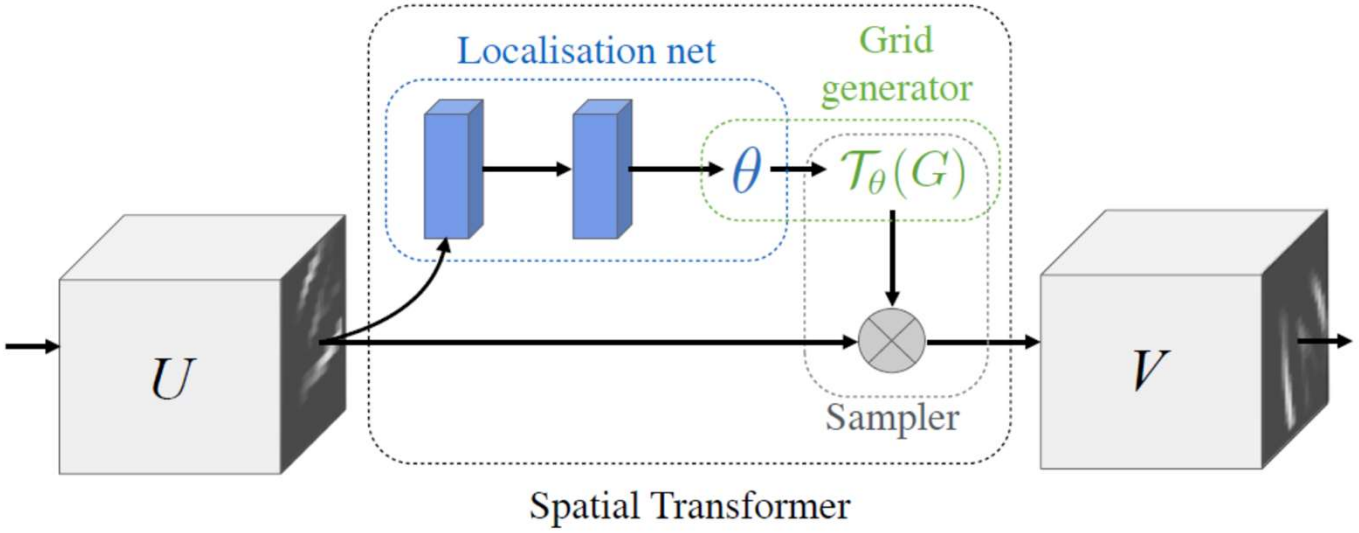
DeepWarp

Идея: глубокая сеть учится предсказывать поле смещений

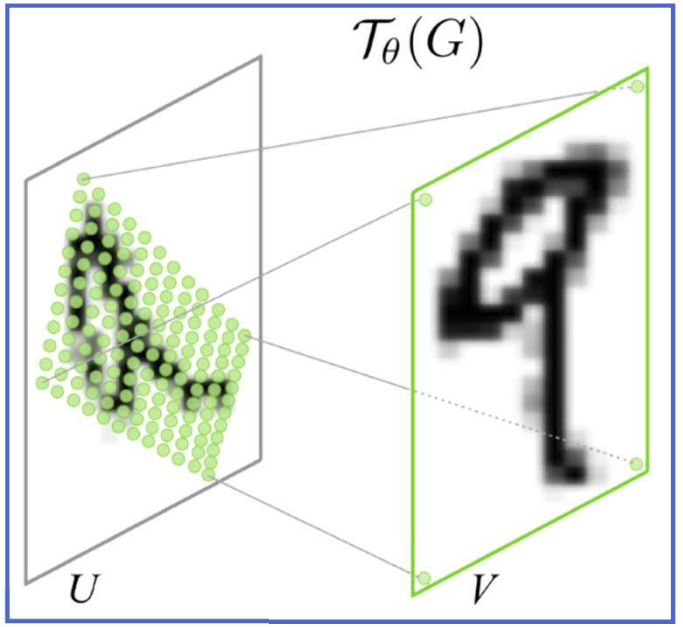


[Ganin,Kononenko,Sungatullina,Lempitsky, 2016]

Spatial transformer nets [Jaderberg et al.2015]



Spatial Transformer



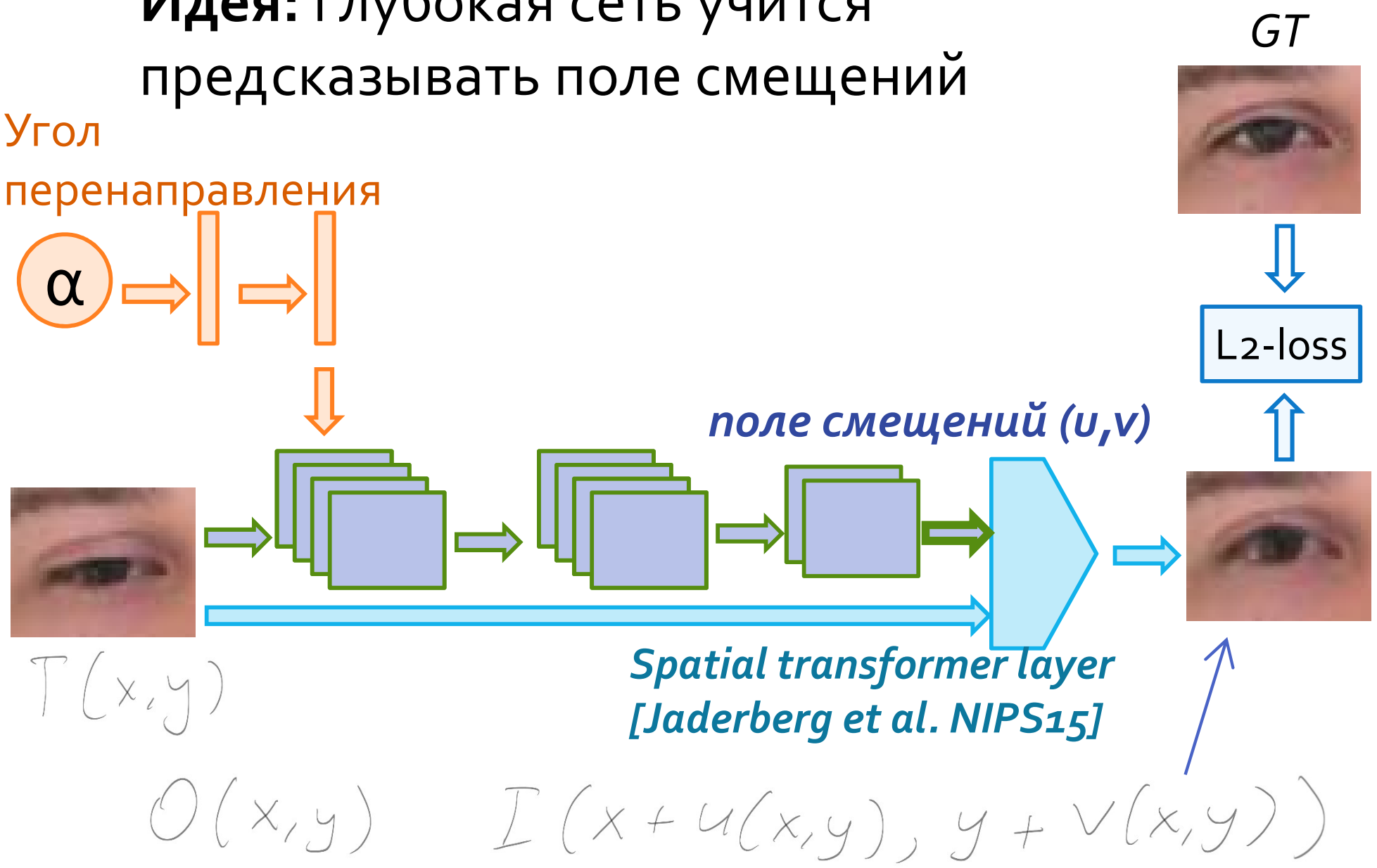
$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|)$$

$$\frac{\partial V_i^c}{\partial U_{nm}^c} = \sum_n^H \sum_m^W \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|)$$

$$\frac{\partial V_i^c}{\partial x_i^s} = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |y_i^s - n|) \begin{cases} 0 & \text{if } |m - x_i^s| \geq 1 \\ 1 & \text{if } m \geq x_i^s \\ -1 & \text{if } m < x_i^s \end{cases}$$

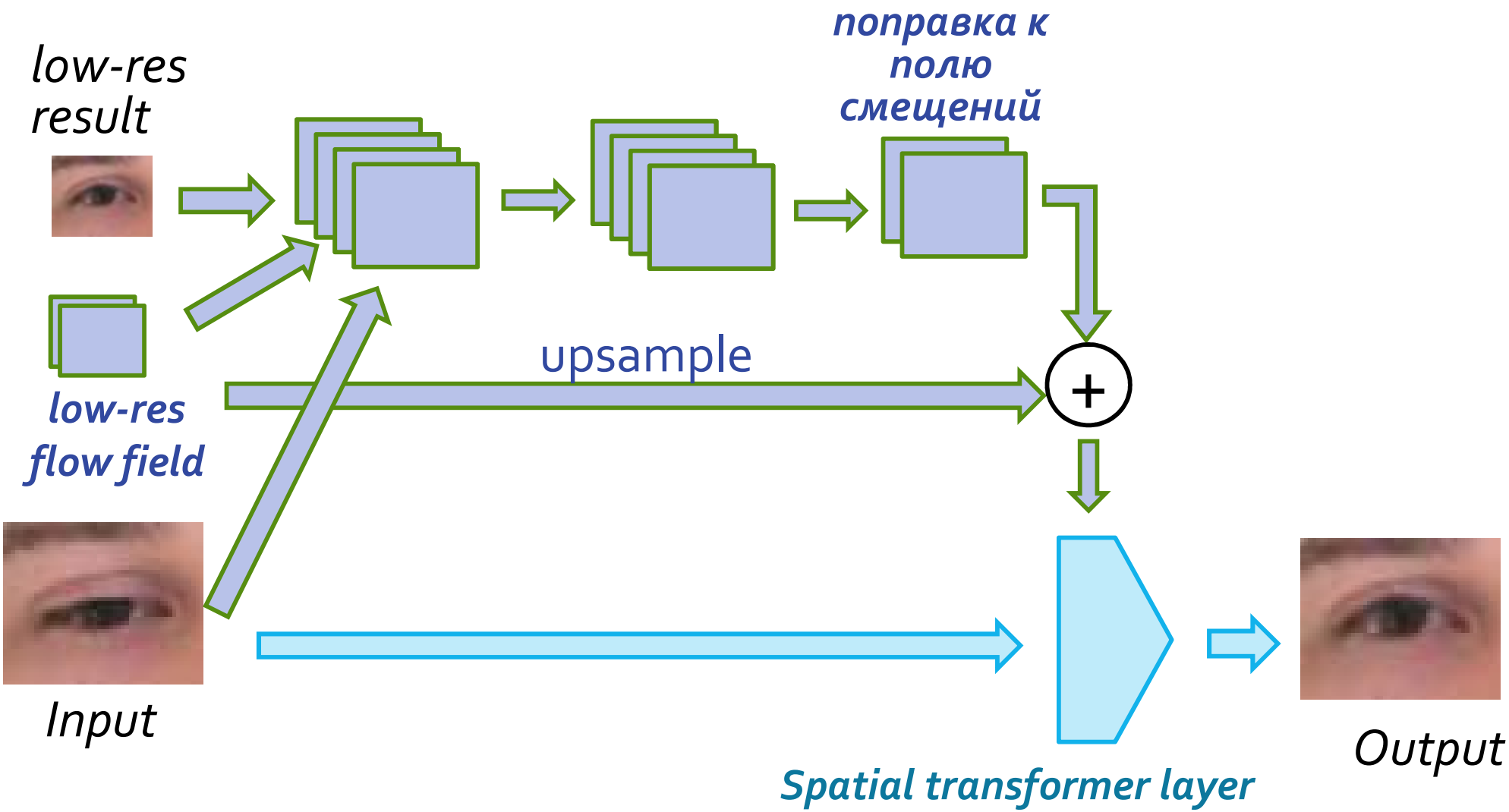
DeepWarp

Идея: глубокая сеть учится предсказывать поле смещений



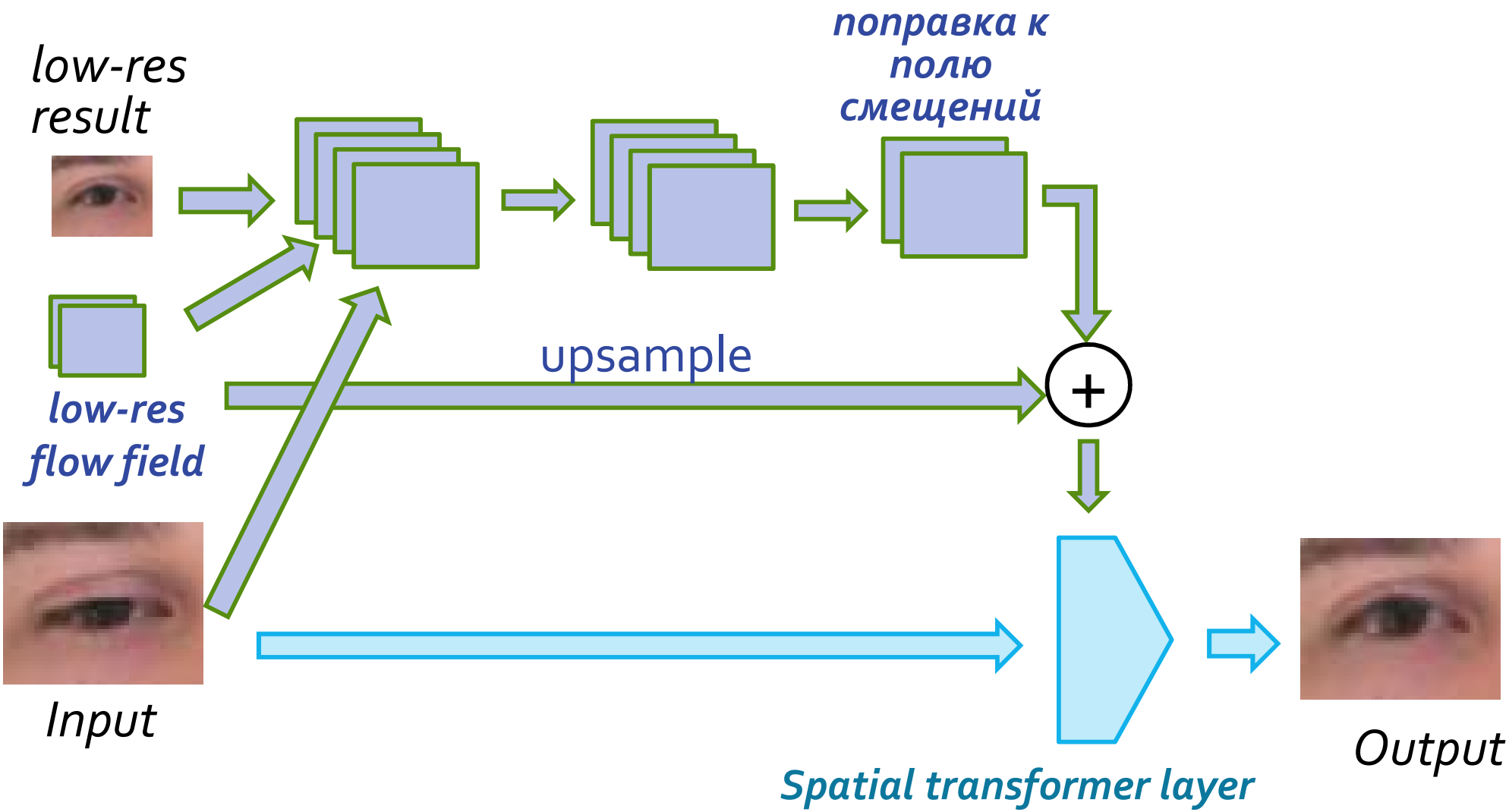
[Ganin,Kononenko,Sungatullina,Lempitsky, 2016]

Второй «уровень» сети



[Ganin,Kononenko,Sungatullina,Lempitsky, 2016]

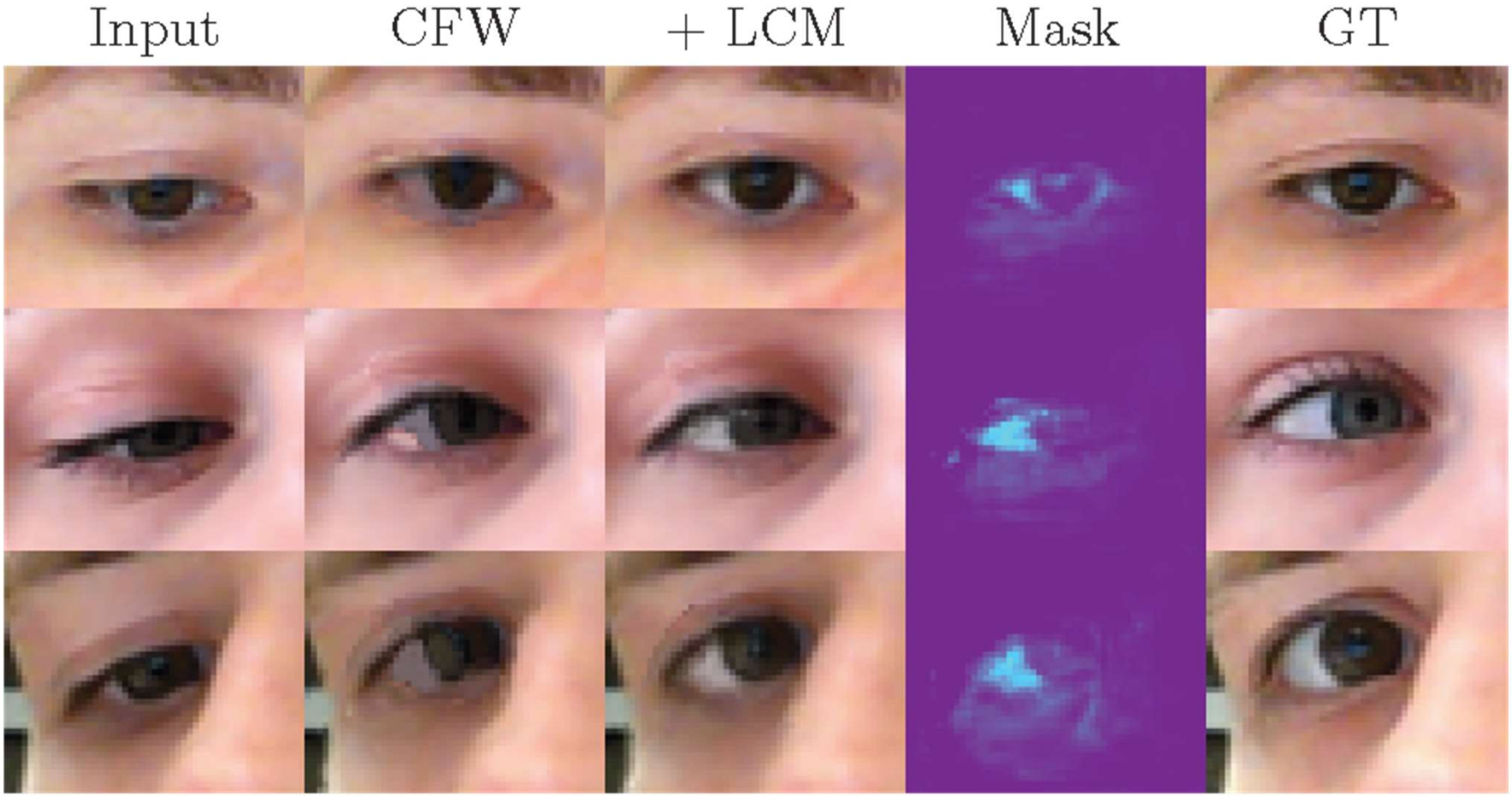
Второй «уровень» сети



[Ganin,Kononenko,Sungatullina,Lempitsky, 2016]

Третий «уровень» сети

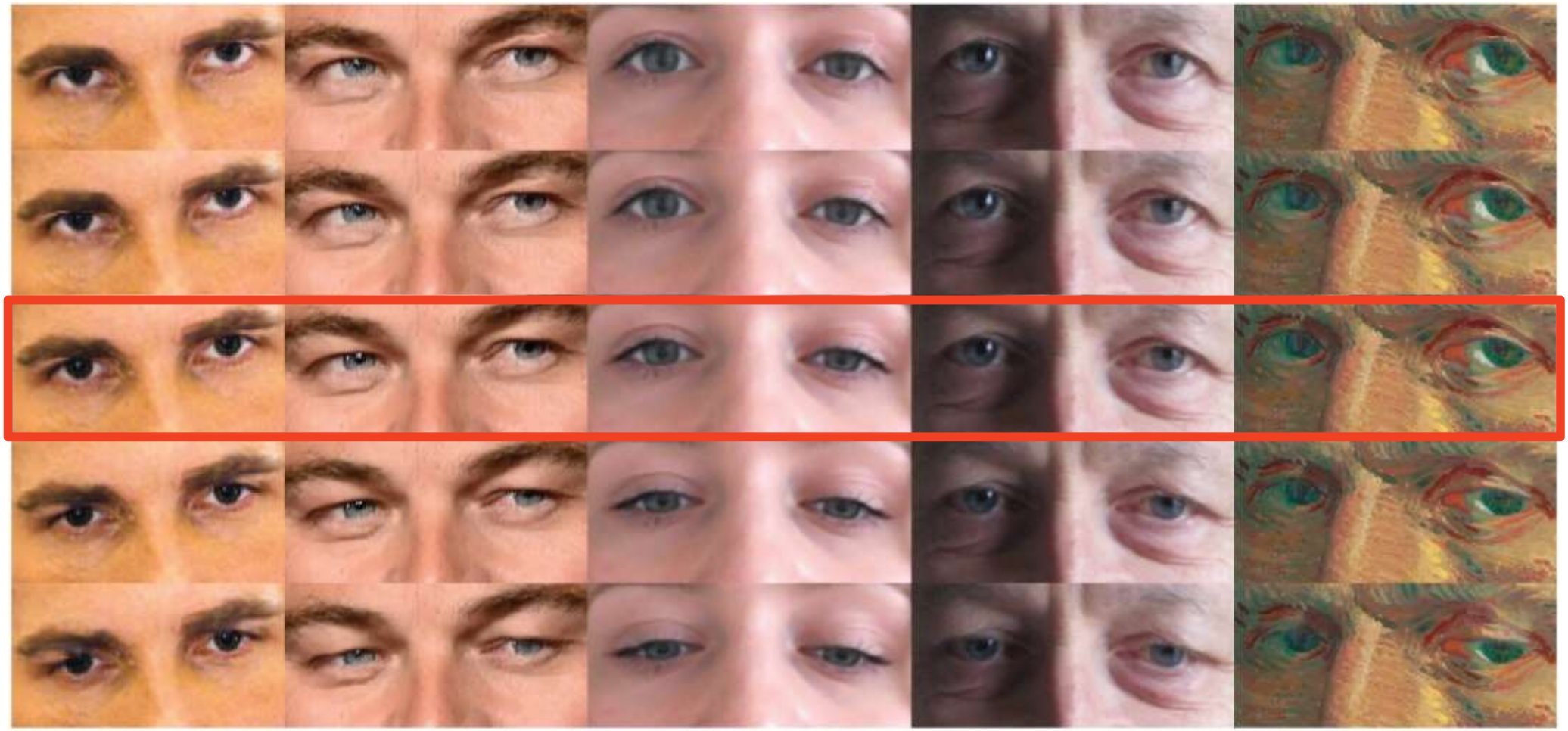
Третий уровень принимает на вход карты признаков и результат предыдущих уровней и предсказывает мультипликативную поправку:



Все три этапа обучаются вместе

Вертикальное перенаправление

15 градусов вверх



15 градусов вниз

[Ganin,Kononenko,Sungatullina,Lempitsky, 2016]

Еще примеры






[Ganin,Kononenko,Sungatullina,Lempitsky, 2016]

План

1. «Стандартные» свёрточные сети
2. «Практическая» оптимизация
3. Синтезирующие сети с простыми функциями потери
4. **Синтез через повторение статистик**
5. Играющие сети (adversarial networks)

Задача «безусловной» генерации

$x \sim P(x)$

- **Научиться получать новые примеры**
- (Научиться оценивать вероятности)

Moment-matching nets [Li et al. 2015]

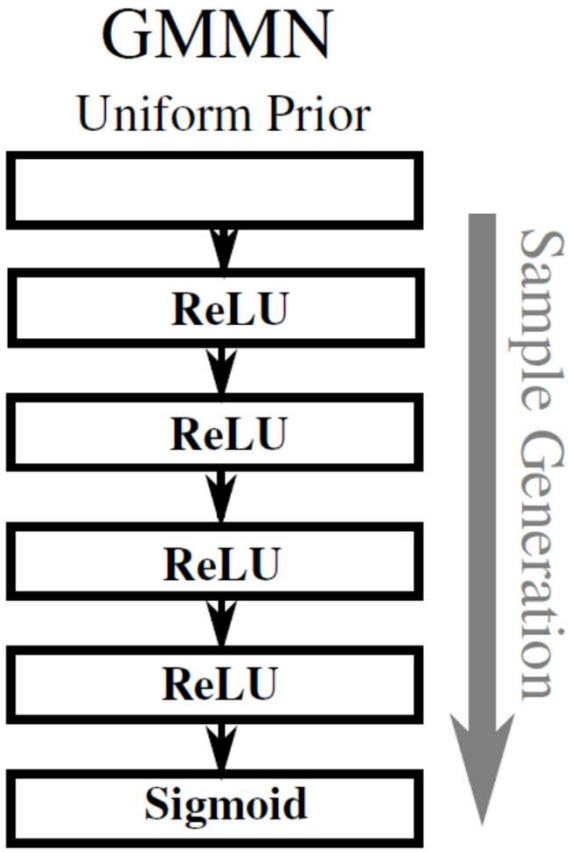
$$\mathcal{L}_{\text{MMD}^2} = \left\| \frac{1}{N}\sum_{i=1}^{N}\phi(x_i) - \frac{1}{M}\sum_{j=1}^{M}\phi(y_j) \right\|^2 \quad (1)$$

$$= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\phi(x_i)^\top\phi(x_{i'}) - \frac{2}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\phi(x_i)^\top\phi(y_j)$$

$$+ \frac{1}{M^2}\sum_{j=1}^{M}\sum_{j'=1}^{M}\phi(y_j)^\top\phi(y_{j'}) \quad (2)$$

$$\mathcal{L}_{\text{MMD}^2} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}k(x_i, x_{i'}) - \frac{2}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}k(x_i, y_j)$$

$$+ \frac{1}{M^2}\sum_{j=1}^{M}\sum_{j'=1}^{M}k(y_j, y_{j'}) \quad (3)$$



Moment-matching nets [Li et al. 2015]

Algorithm 1: GMMN minibatch training

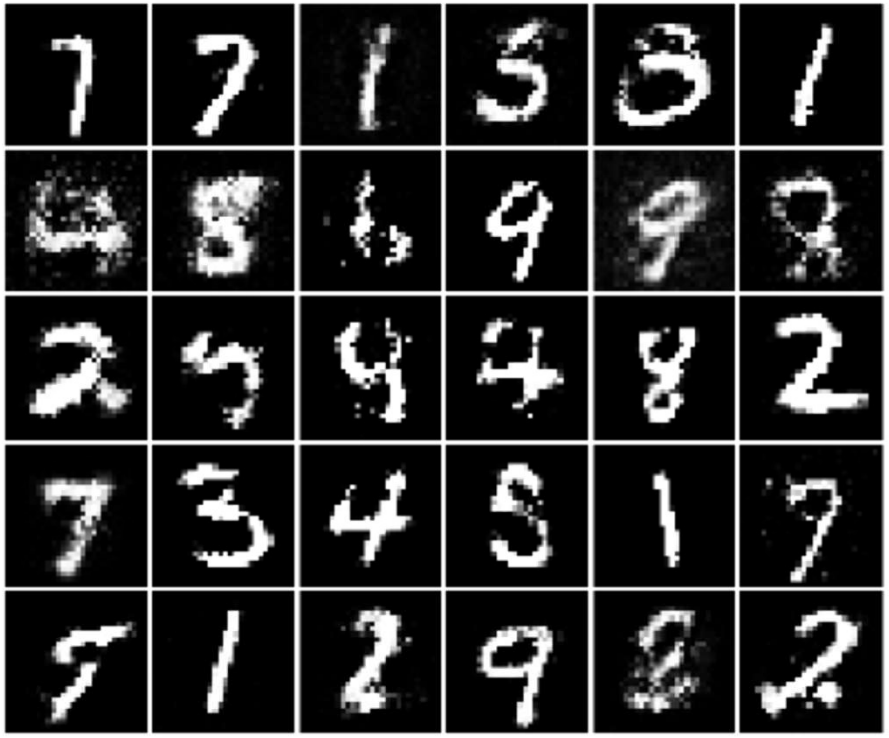
Input : Dataset $\{\mathbf{x}_1^d, ..., \mathbf{x}_N^d\}$, prior $p(\mathbf{h})$, network $f(\mathbf{h}; \mathbf{w})$ with initial parameter $\mathbf{w}^{(0)}$

Output: Learned parameter $\mathbf{w}^*$

while *Stopping criterion not met* **do**
  Get a minibatch of data $\mathbf{X}^d \leftarrow \{\mathbf{x}_{i_1}^d, ..., \mathbf{x}_{i_b}^d\}$
  Get a new set of samples $\mathbf{X}^s \leftarrow \{\mathbf{x}_1^s, ..., \mathbf{x}_b^s\}$
  Compute gradient $\frac{\partial \mathcal{L}_{\text{MMD}}}{\partial \mathbf{w}}$ on $\mathbf{X}^d$ and $\mathbf{X}^s$
  Take a gradient step to update $\mathbf{w}$
end

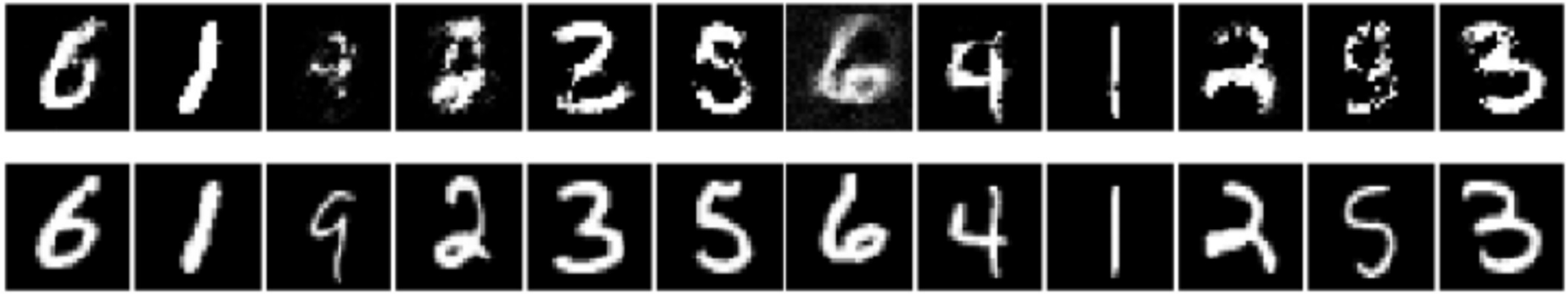
Use mixture of several Gaussian kernels with different bandwidths

Moment-matching nets [Li et al. 2015]



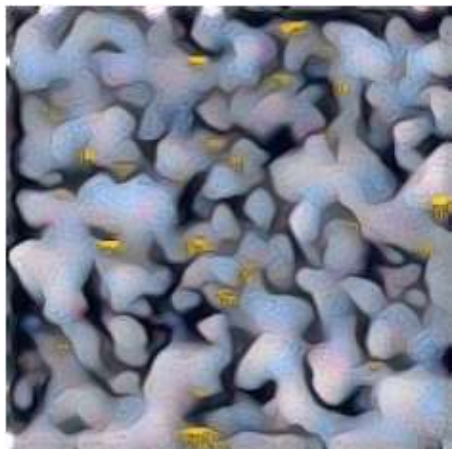
(a) GMMN MNIST samples

(b) GMMN TFD samples



Синтез текстур

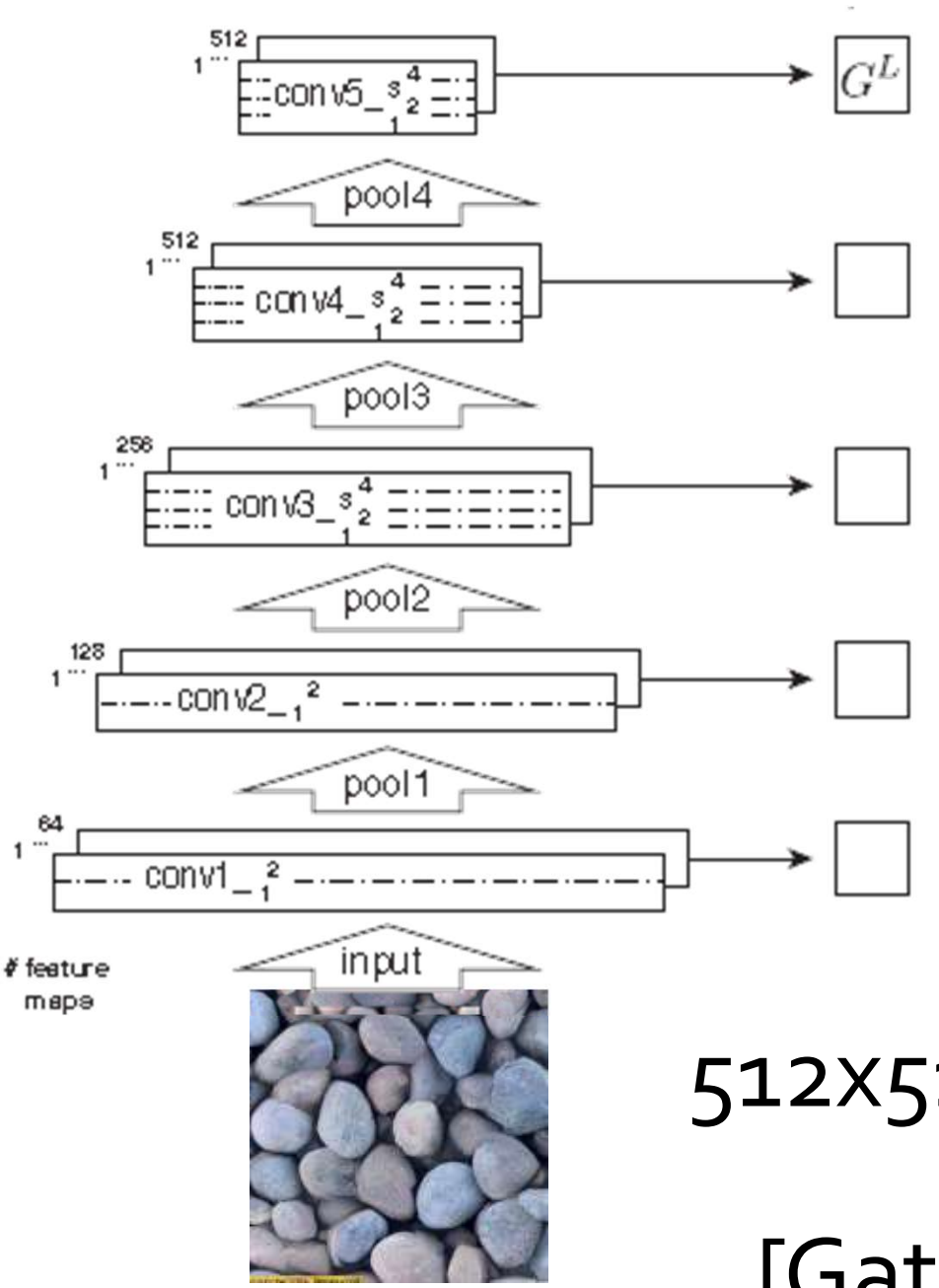
Ключевой вопрос: какие статистики описывают текстуру?

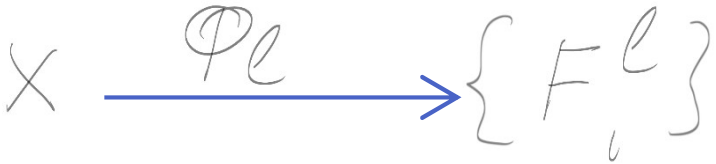



похожи различны

- Попиксельная разность?
- Гистограммы цветов?
- Нужны статистики более высоких порядков!

Статистики, порождаемые нейросетью





Матрица Грамма:

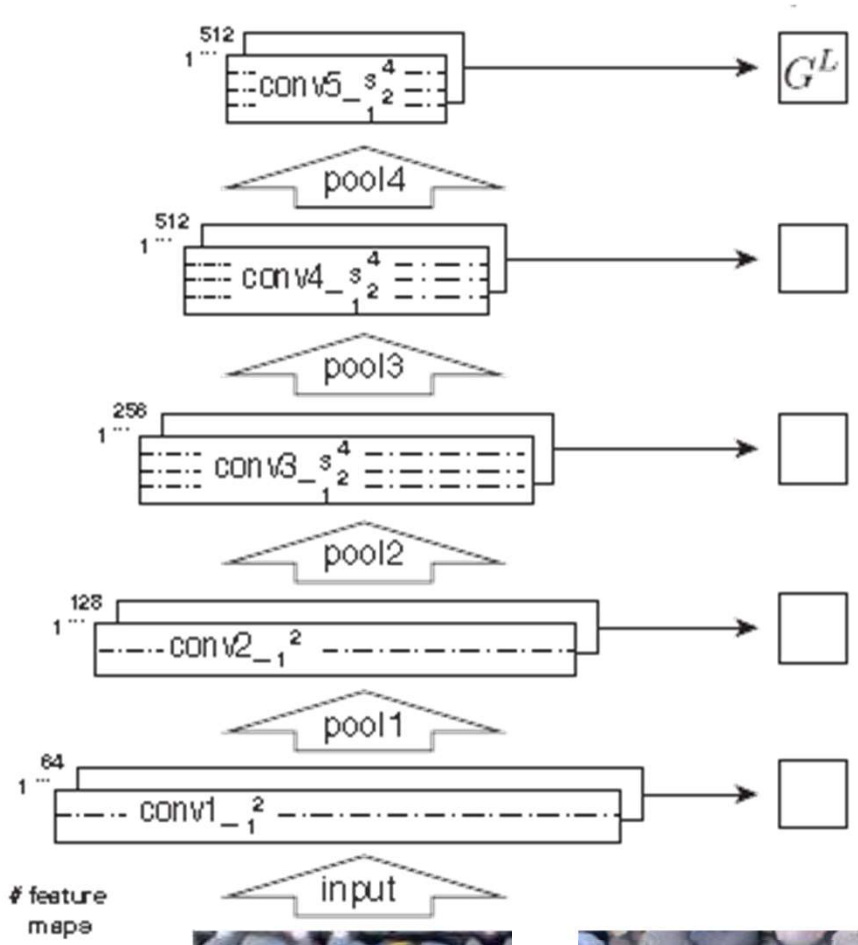
$$\left[G^l \right]$$

$$G^l_{ij} = \sum_k F^l_{i,k} \cdot F^l_{j,k}$$

512x512-мерный дескриптор

[Gatys, Ecker, Bethge 2015]

Синтез текстур методом прообразов

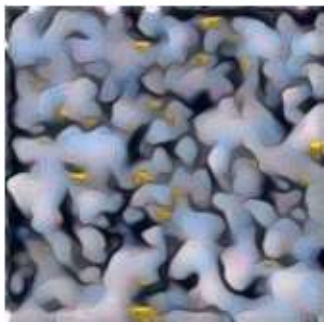


$$x^* \longrightarrow \phi_\ell(x^*) \longrightarrow G(x^*)$$

$$x \longrightarrow \phi_\ell(x) \longrightarrow G(x)$$

$$L(x) = \| G(x) - G(x^*) \|_F^2$$

$$x^{t+} \leftarrow x^t - \alpha_t \frac{dL(x^t)}{dx}$$



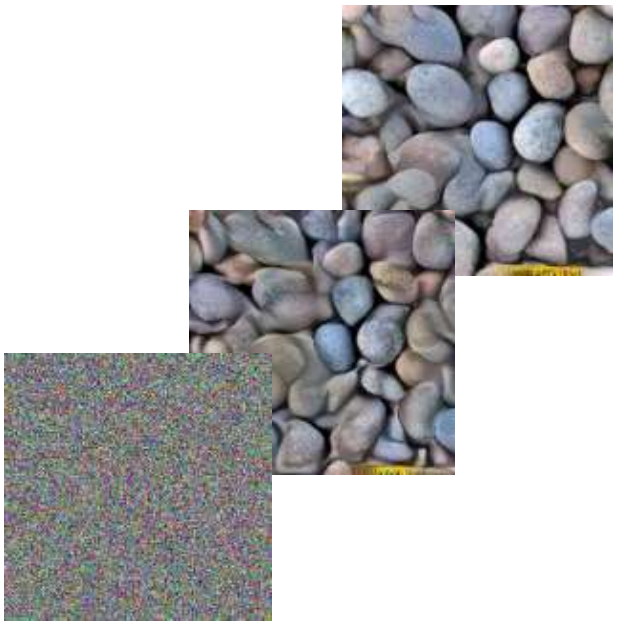
x

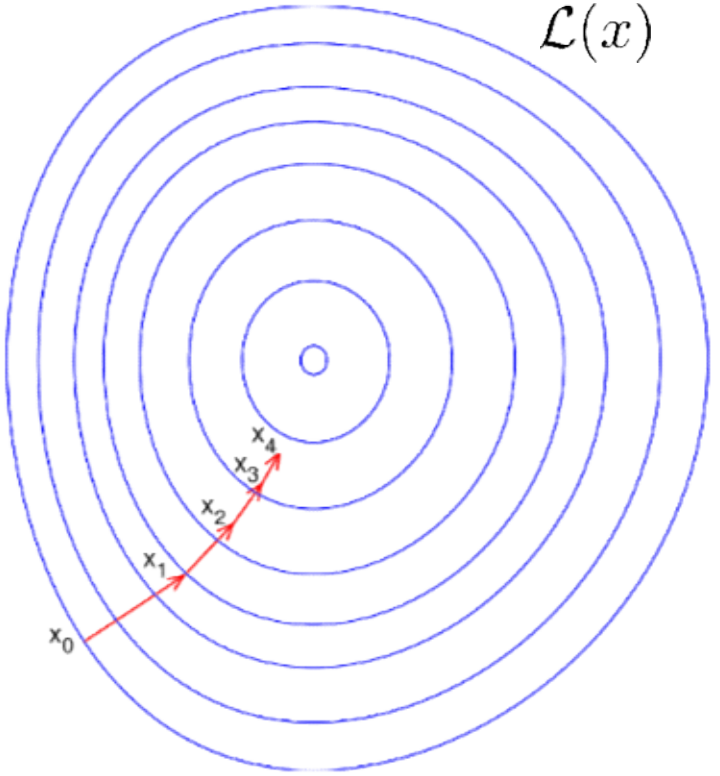
x^*

[Gatys, Ecker, Bethge 2015]

Image generation by optimization

$$x^* = \arg\min_x \mathcal{L}(x)$$





[Gatys, Ecker, Bethge 2015]

Текстуры, созданные методом прообразов

Synthesised

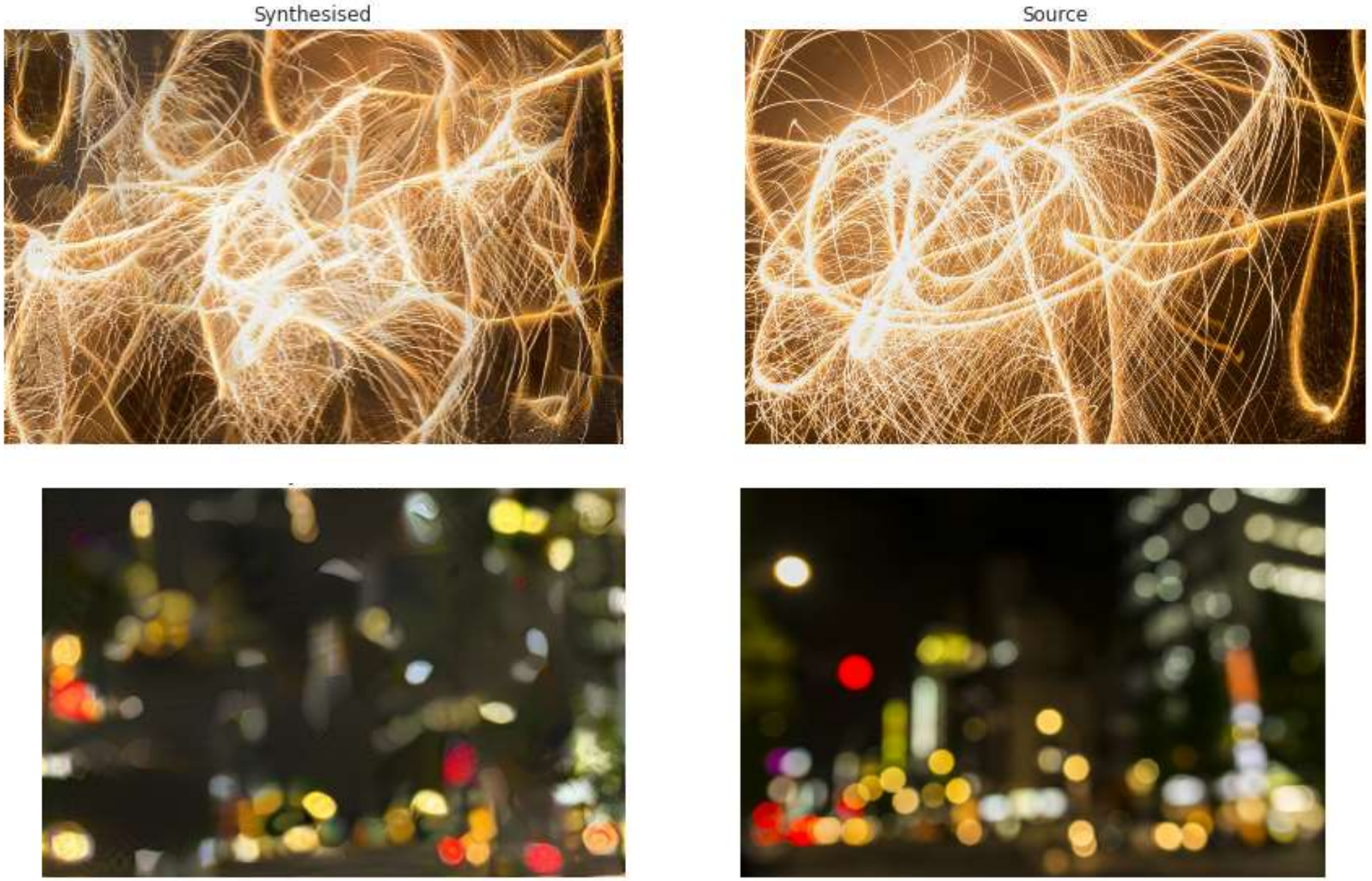
Source

Synthesised

Source

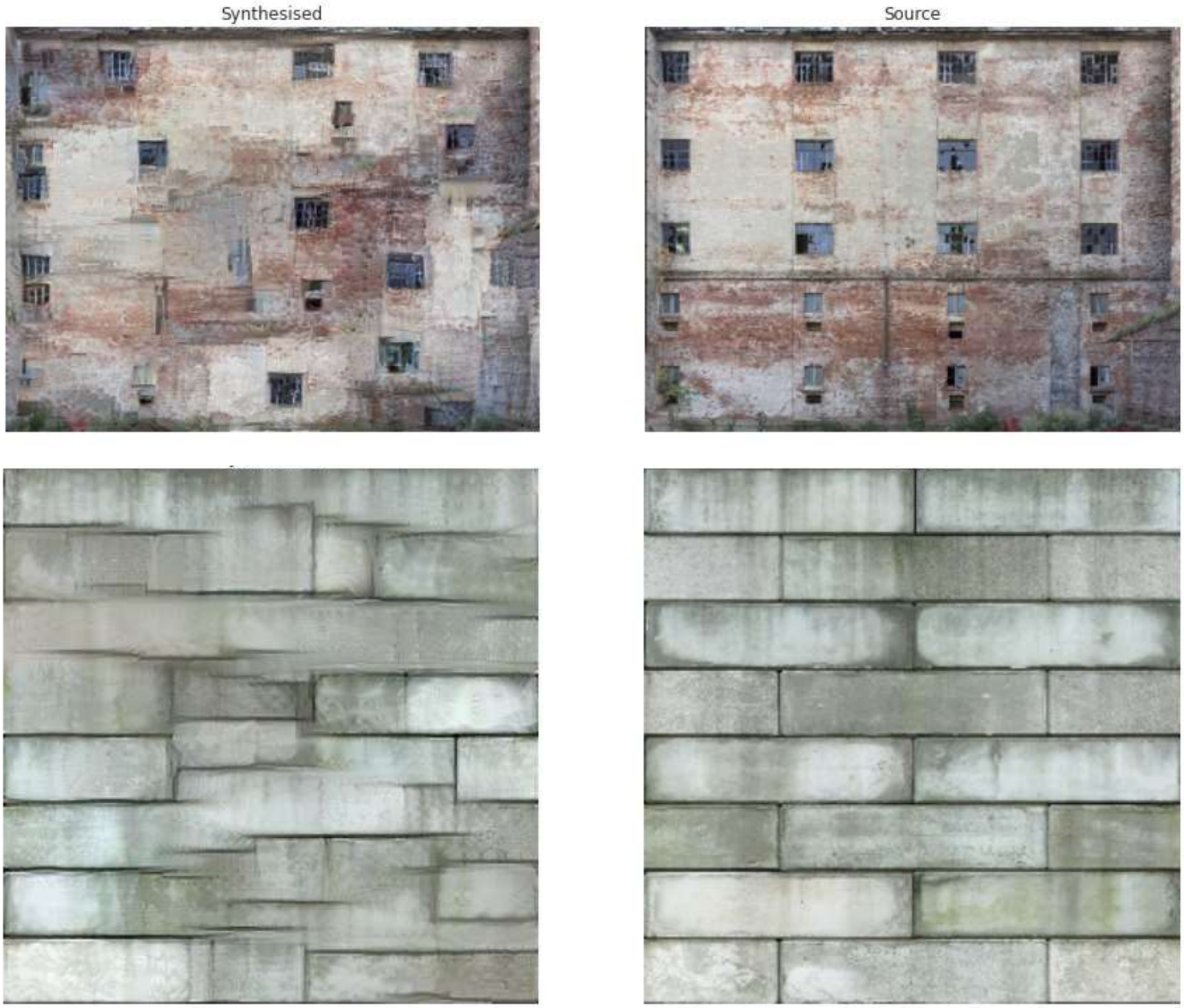
[Gatys, Ecker, Bethge 2015]

Текстуры, созданные методом прообразов



[Gatys, Ecker, Bethge 2015]

Текстуры, созданные методом прообразов



[Gatys, Ecker, Bethge 2015]

Texture network

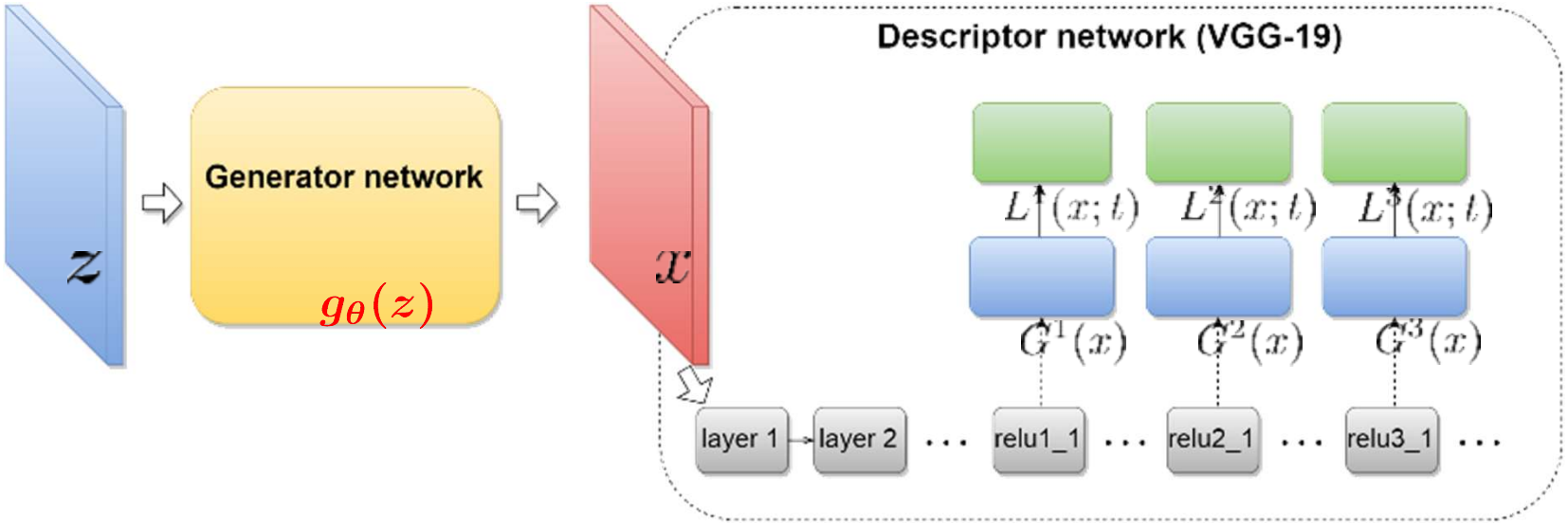


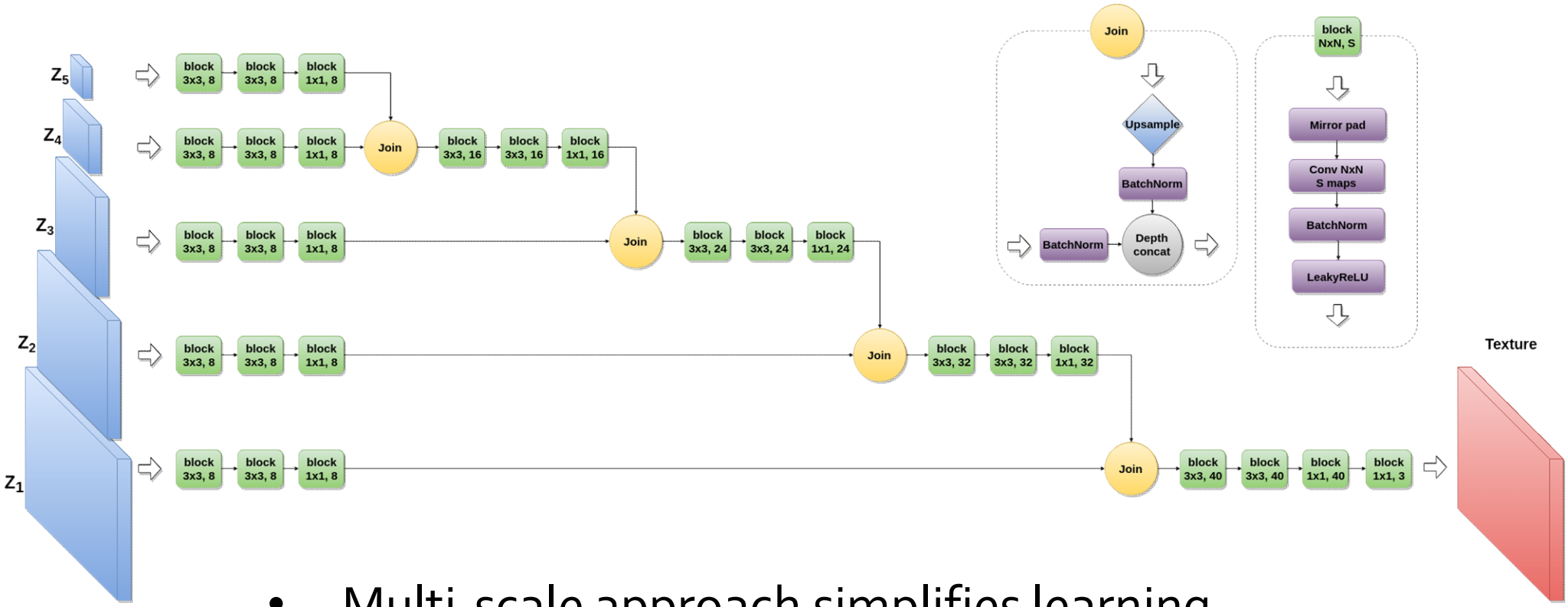
Image generation: $$x = g_\theta(z), \quad z \sim U(0,1)$$

Optimization task becomes:

$$\min_\theta \mathbb{E}\mathcal{L}_{texture}(g_\theta(z);t), \quad z \sim U(0,1)$$

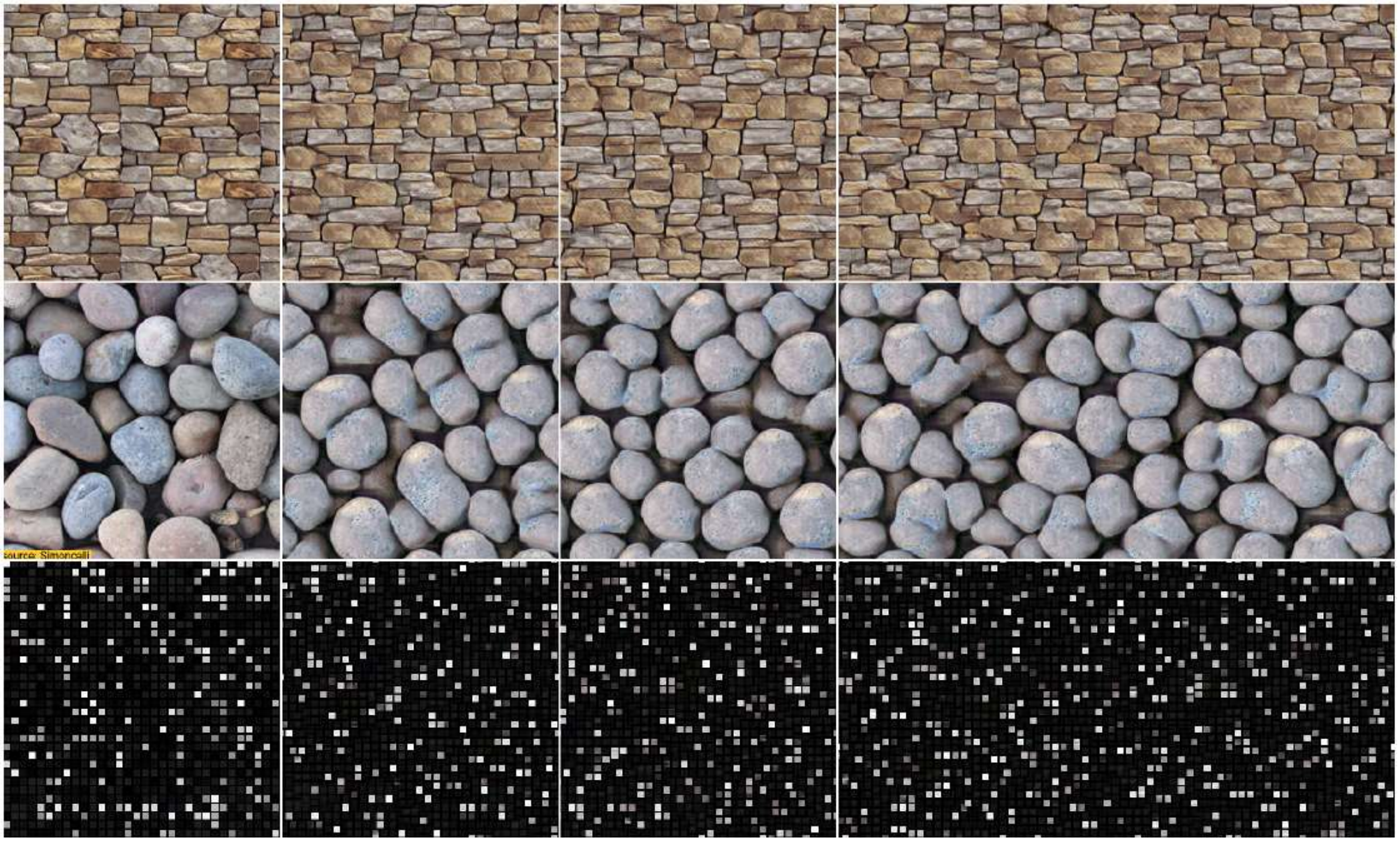
$$\theta^{k+1} = \theta^k - \alpha \frac{\partial \mathcal{L}(g_\theta(z);t)}{\partial \theta}$$

Generator networks details



- Multi-scale approach simplifies learning
- Looks scary, but still feedforward and fully convolutional

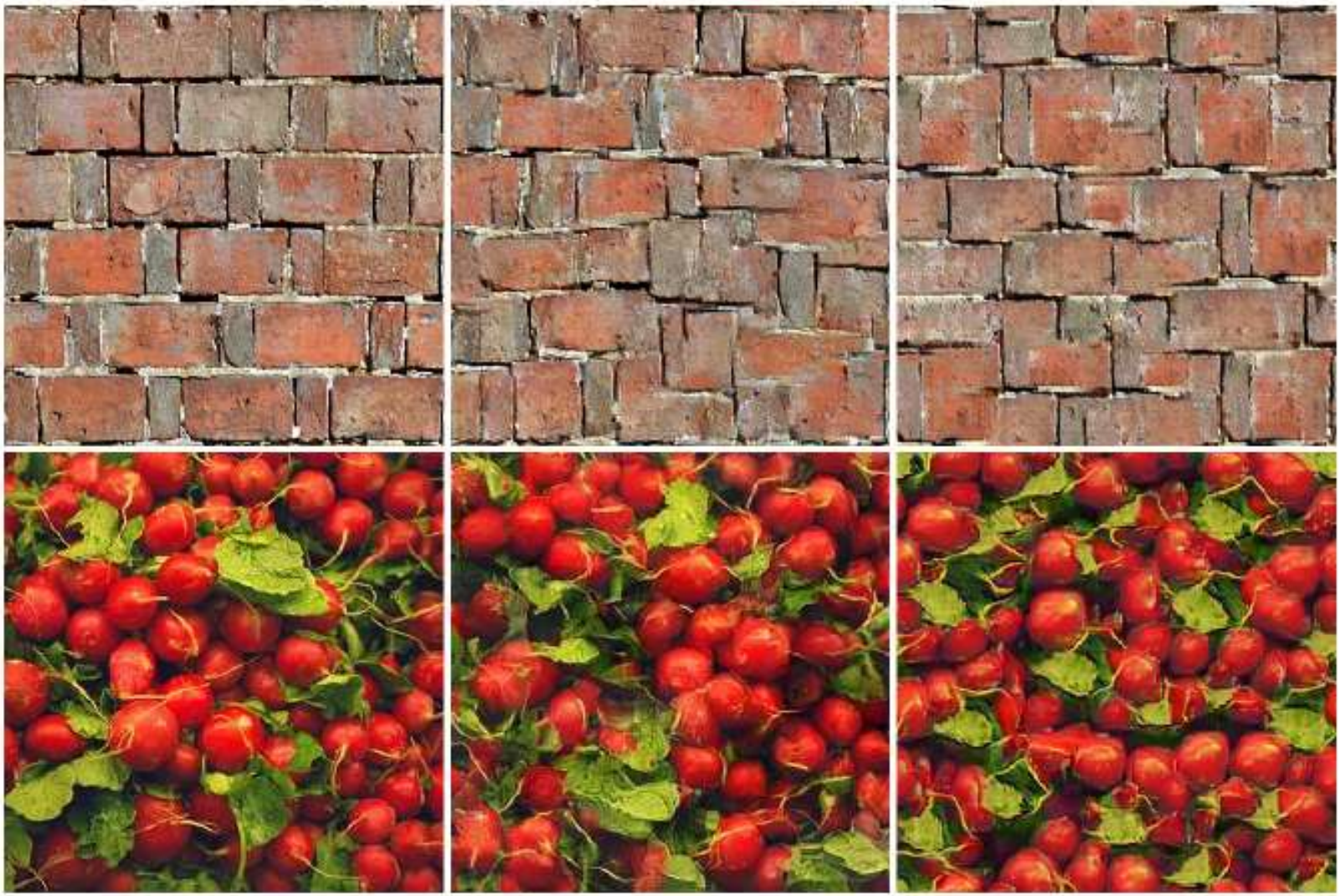
Примеры прямого синтеза



образец

результаты

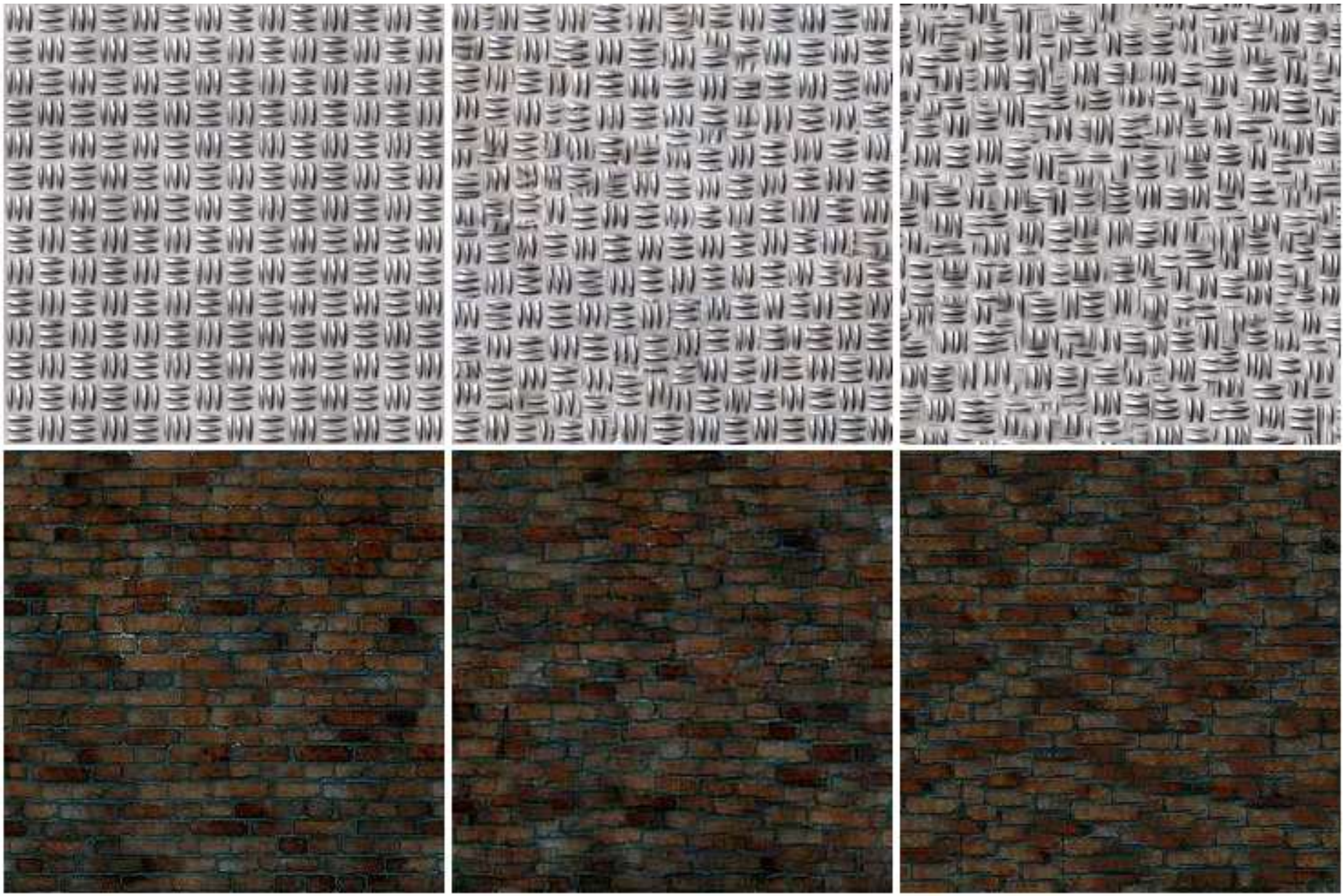
Сравнение синтеза текстур



образец оптим. TextureNet

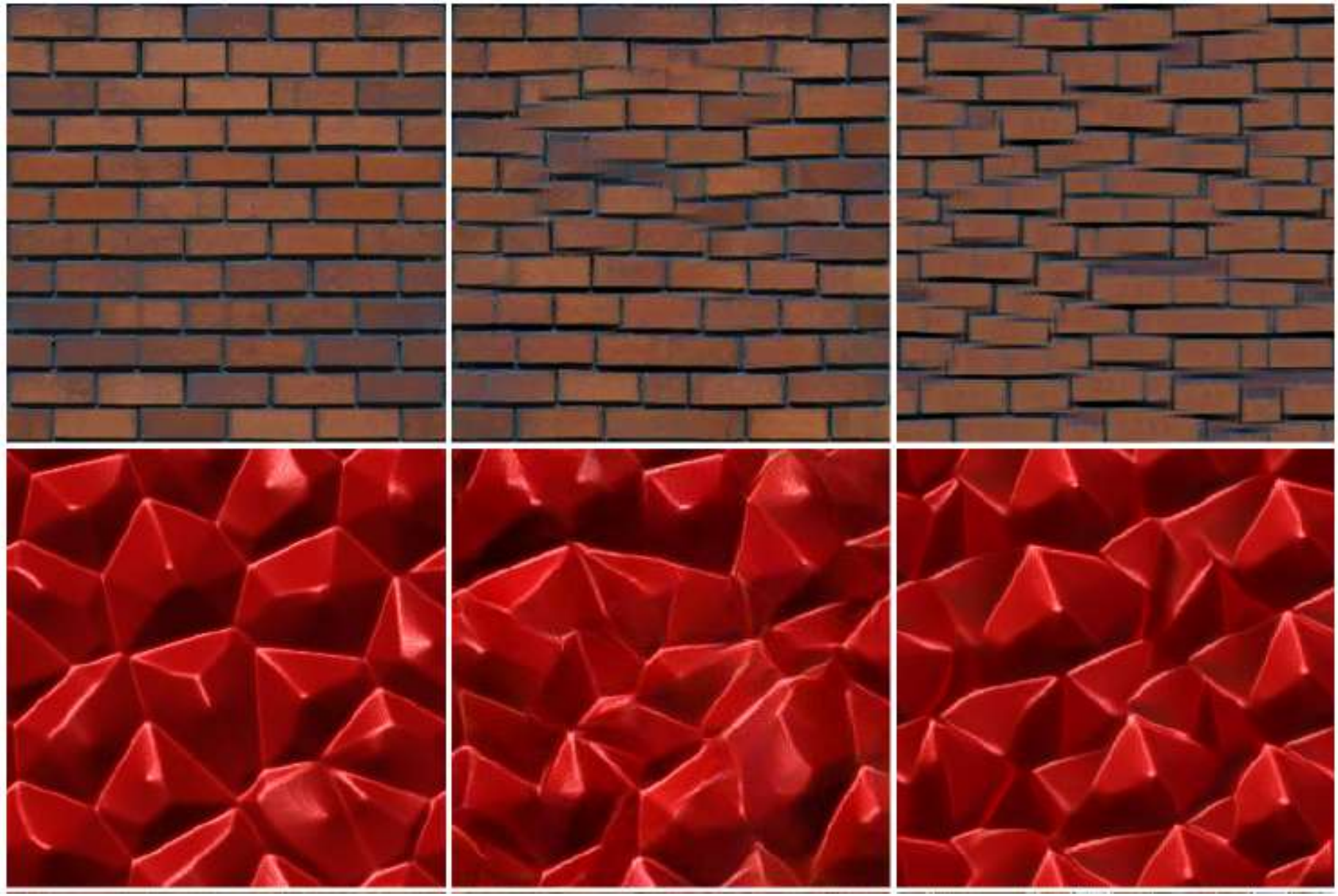
[Ulyanov, Lebedev, Vedaldi, Lempitsky ICML16]

Сравнение синтеза текстур



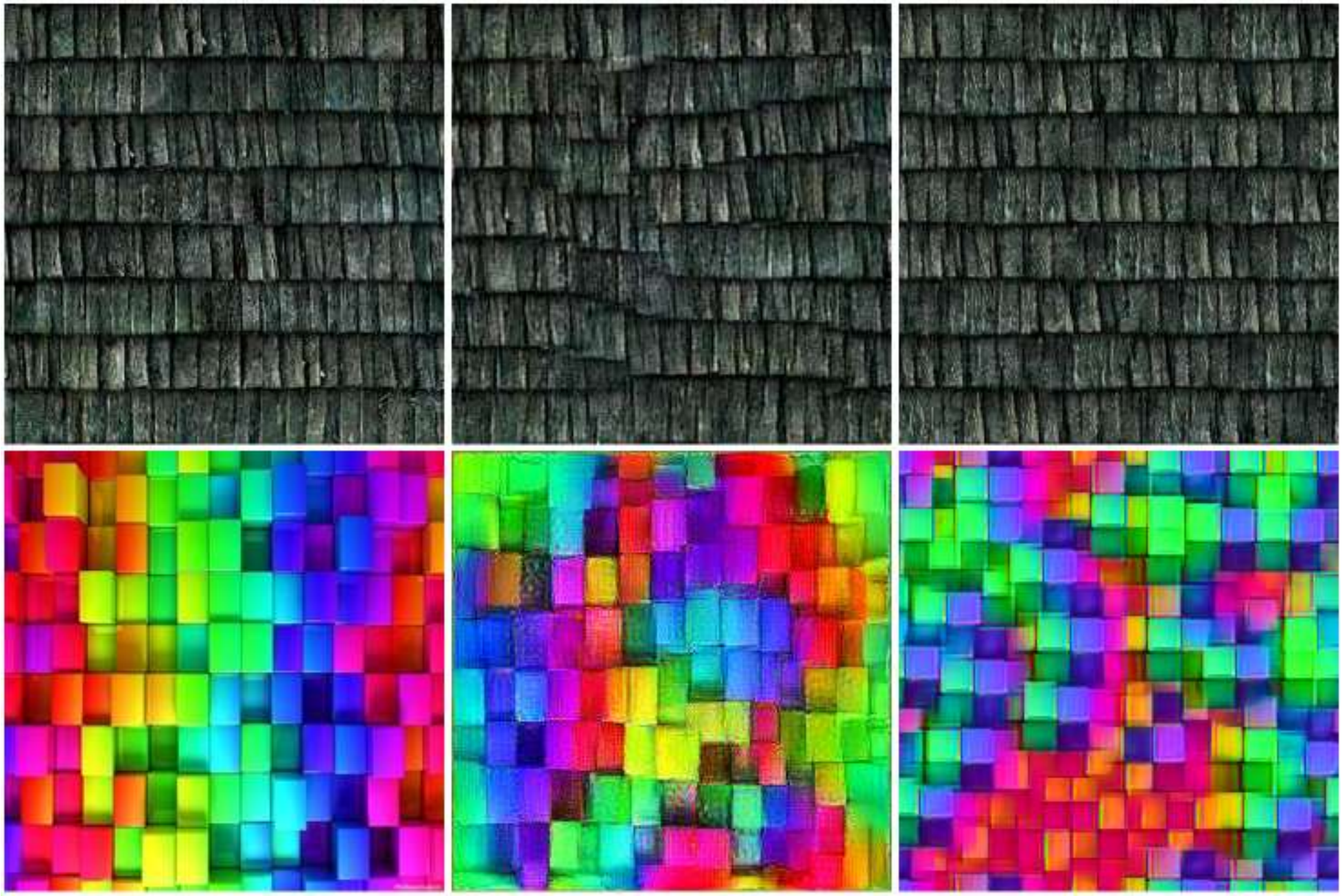
образец оптим. TextureNet

Сравнение синтеза текстур



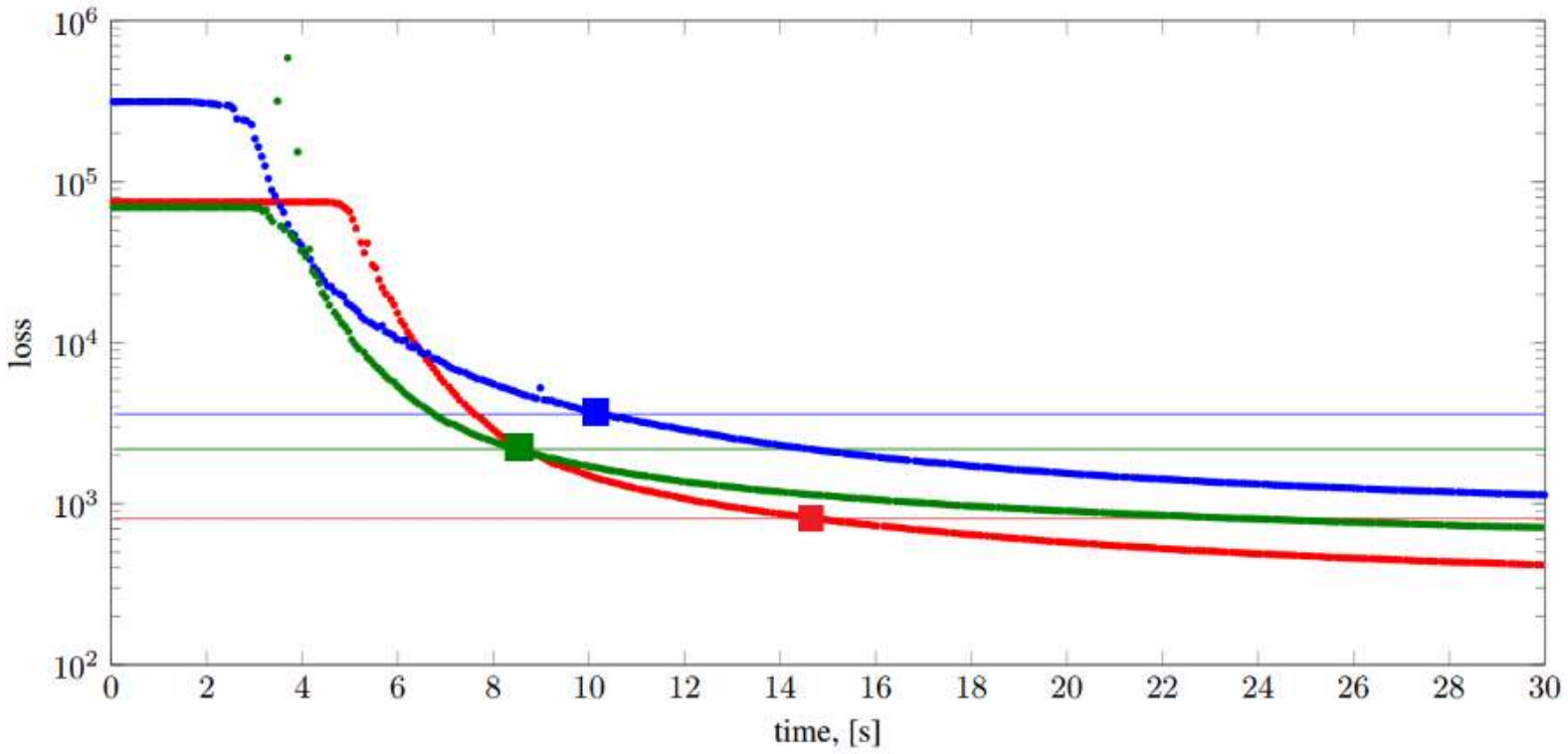
образец оптим. TextureNet

Сравнение синтеза текстур



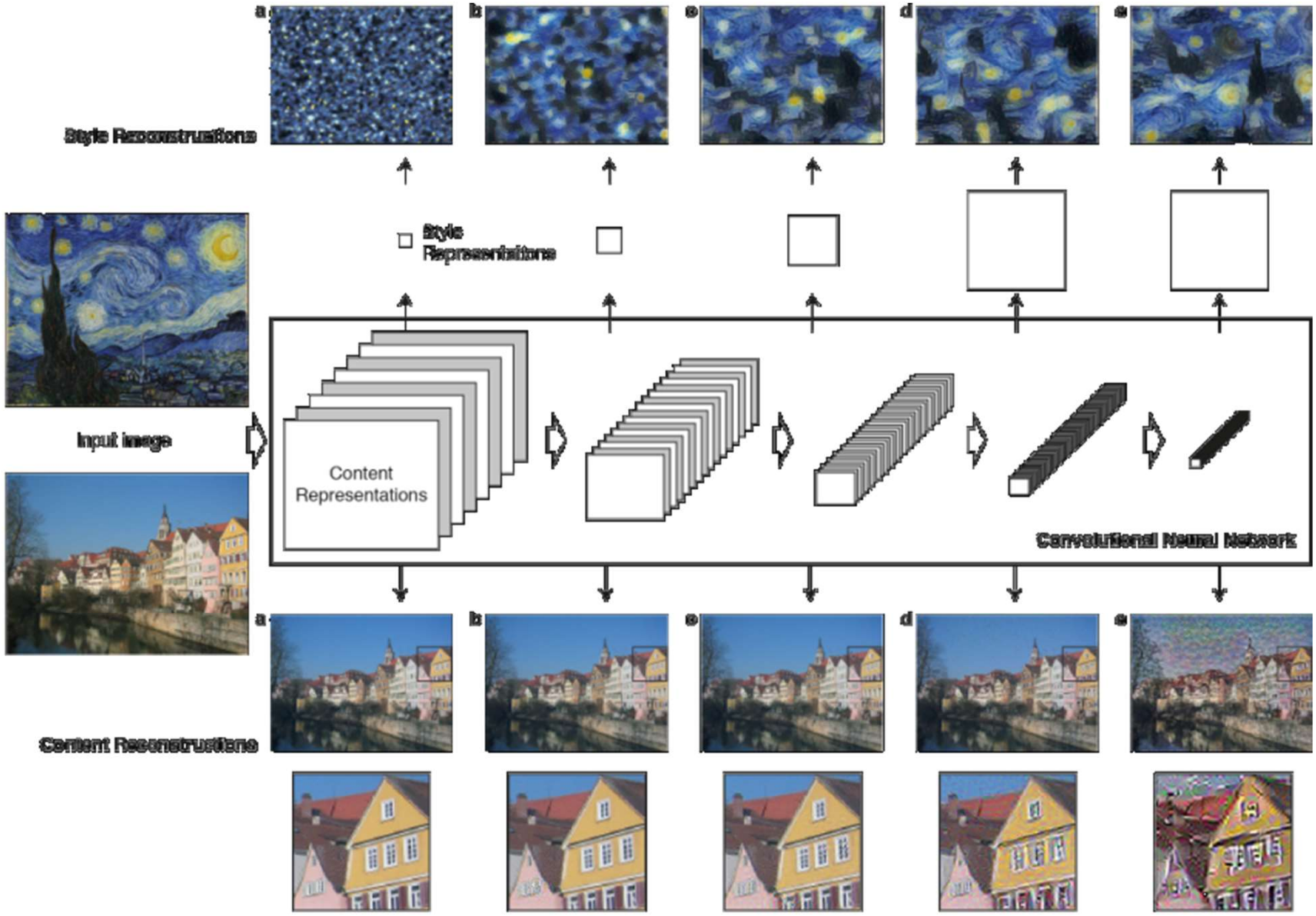
образец оптим. TextureNet

Quantitative evaluation



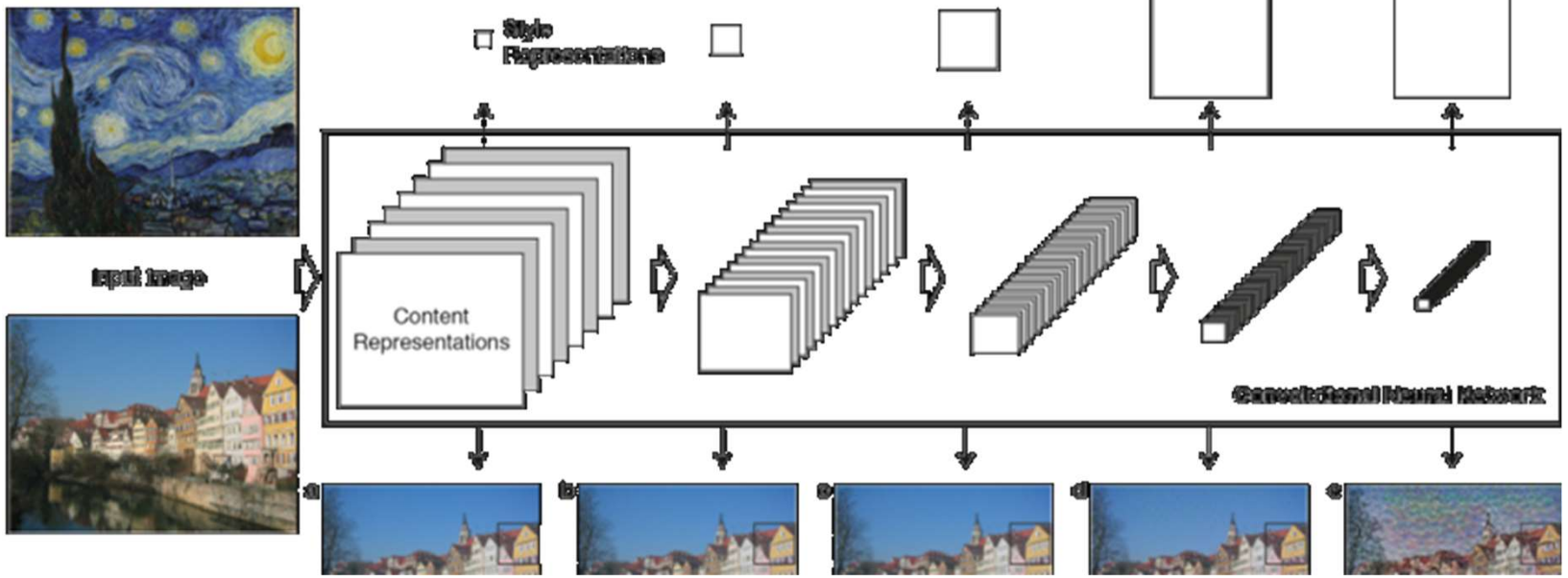
- Horizontal lines: average sample loss
  - **0.06 second** to generate sample
- Dotted lines: *optimization-based* loss as a function of time
  - **10 seconds** to achieve the same loss values

Стилизация изображений



[Gatys, Ecker, Bethge 2015]

Stylization with texture loss



$$|G^{\ell} - G_A^{\ell}|^2$$

$$\|F^{\ell'} - F_B^{\ell'}\|^2$$

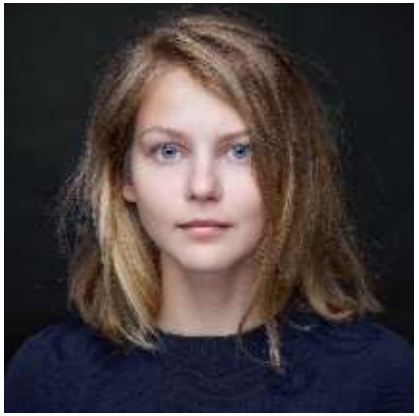
$$L = |G^{\ell} - G_A^{\ell}|^2 + \|F^{\ell'} - F_B^{\ell'}\|^2$$

[Gatys, Ecker, Bethge 2015]

Neural algorithm for artistic style

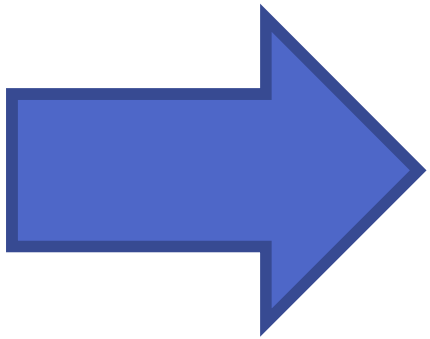
[Gatys et al. 2015]

Feed-forward stylization



photo

style



result

“Style” network

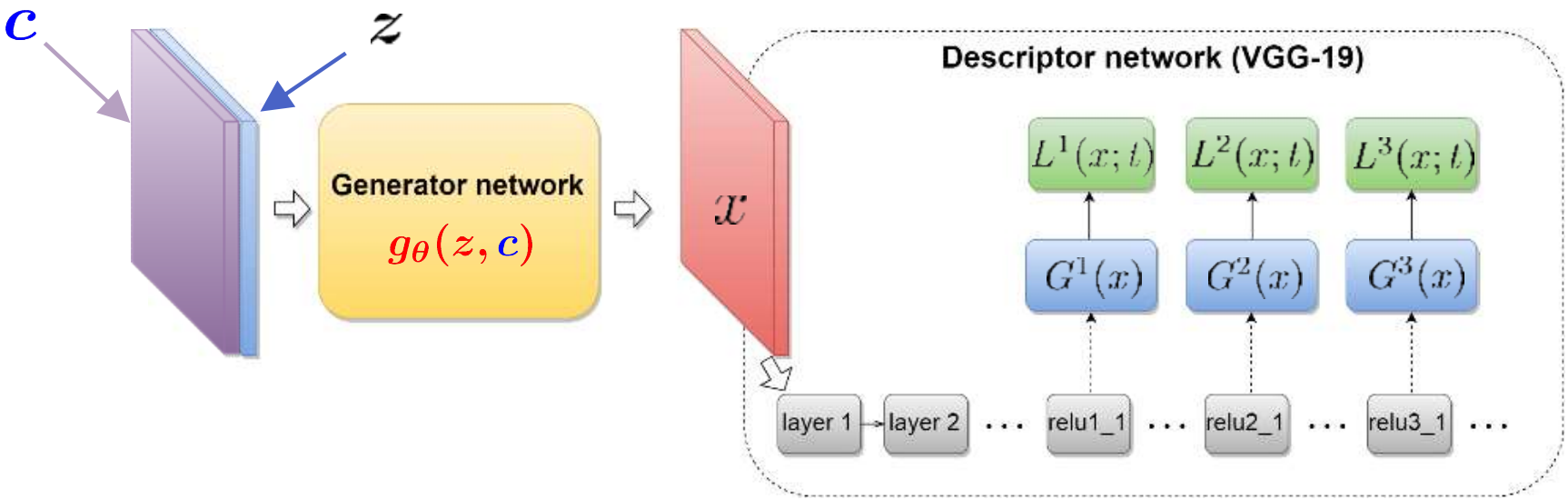


Image generation: $$x = g_\theta(\boldsymbol{z}, \boldsymbol{c}), \quad \boldsymbol{z} \sim U(0,1)$$

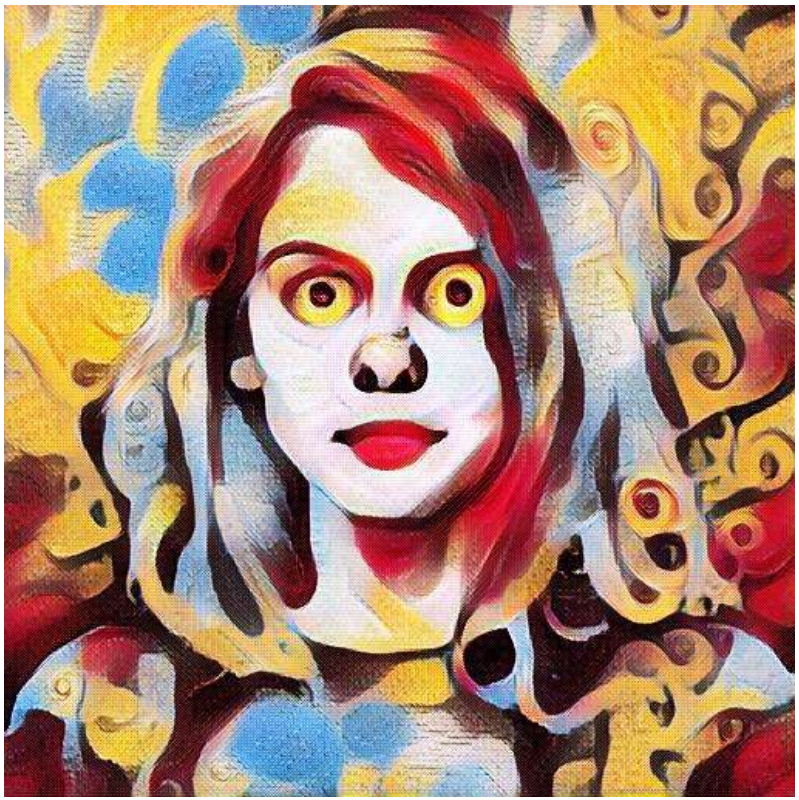
Optimization task: $$\min_\theta \mathbb{E}\mathcal{L}(g_\theta(\boldsymbol{z}, \boldsymbol{c}); c, t), \quad \boldsymbol{z} \sim U(0,1)$$

Feed-forward stylization

feedforward

optimization

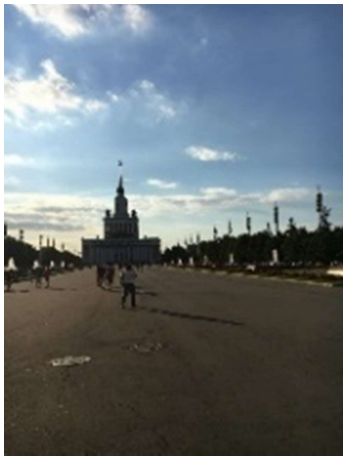
Feed-forward stylization



feedforward

optimization

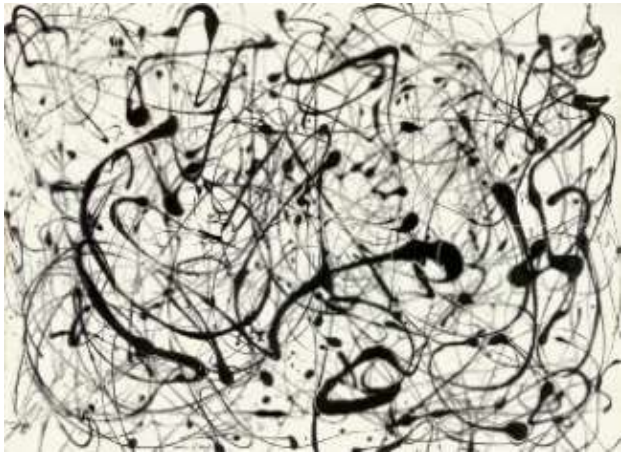
Feed-forward stylization



feedforward

optimization

Feed-forward stylization



feedforward

optimization

План

1. «Стандартные» свёрточные сети
2. «Практическая» оптимизация
3. Синтезирующие сети с простыми функциями потери
4. Синтез через повторение статистик
5. **Играющие сети (adversarial networks)**

Let us look at reconstruction deficiencies

- How can we tell between real and synthetic?
- Can a **computer** tell between them?

content

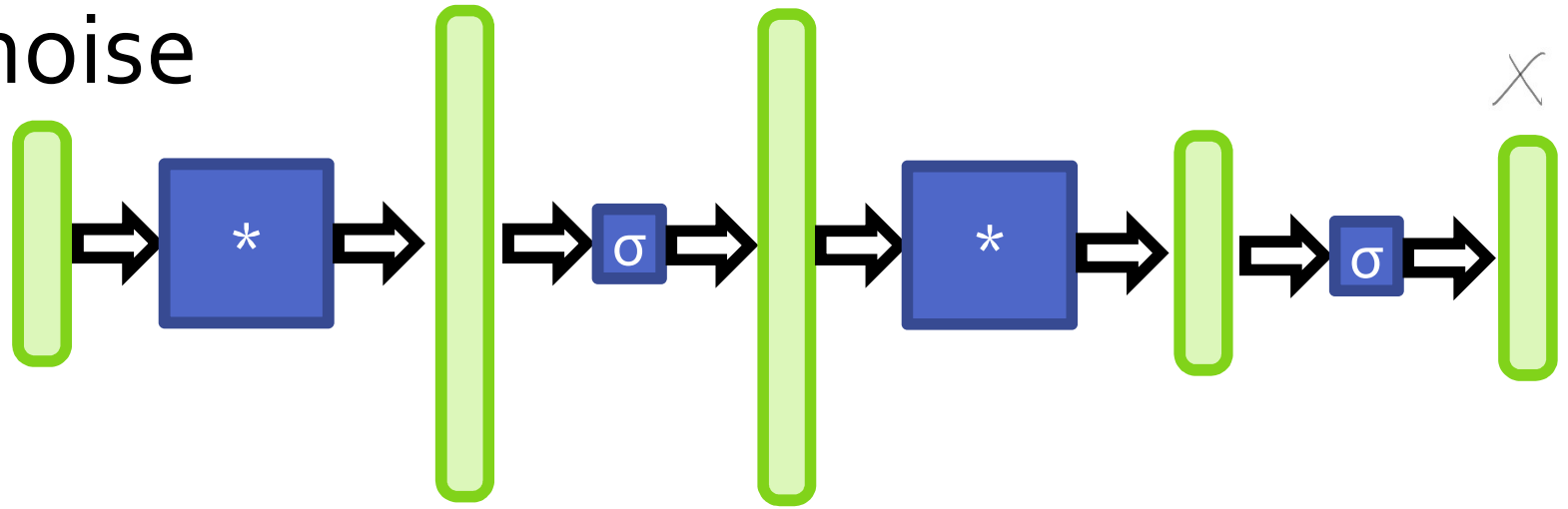
feedback

Generator Network: generates content (e.g. images)

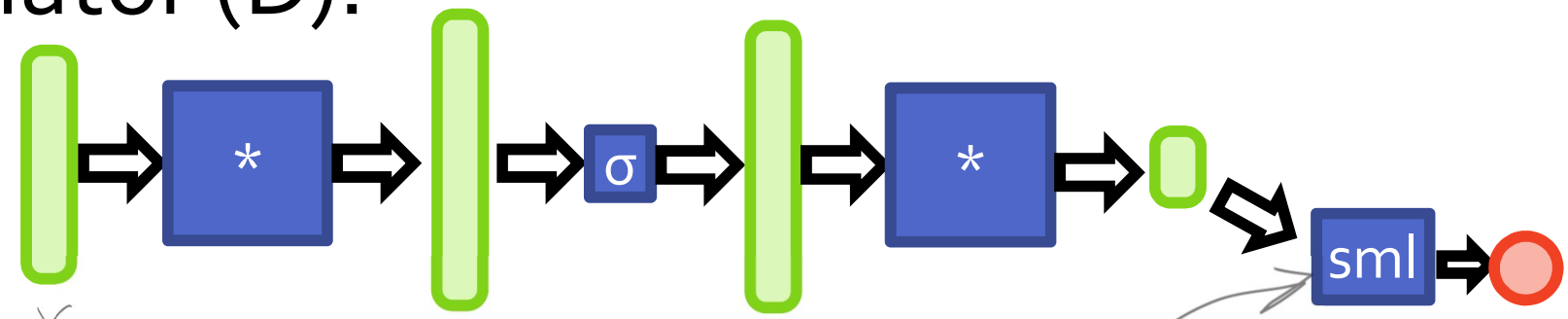
Discriminator Network: assesses the result (e.g. images)

Adversarial generative networks

Generator (G):



Discriminator (D):



[Goodfellow et al. NIPS 14]

Adversarial learning

for number of training iterations **do**

for k steps **do**

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

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

end for

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

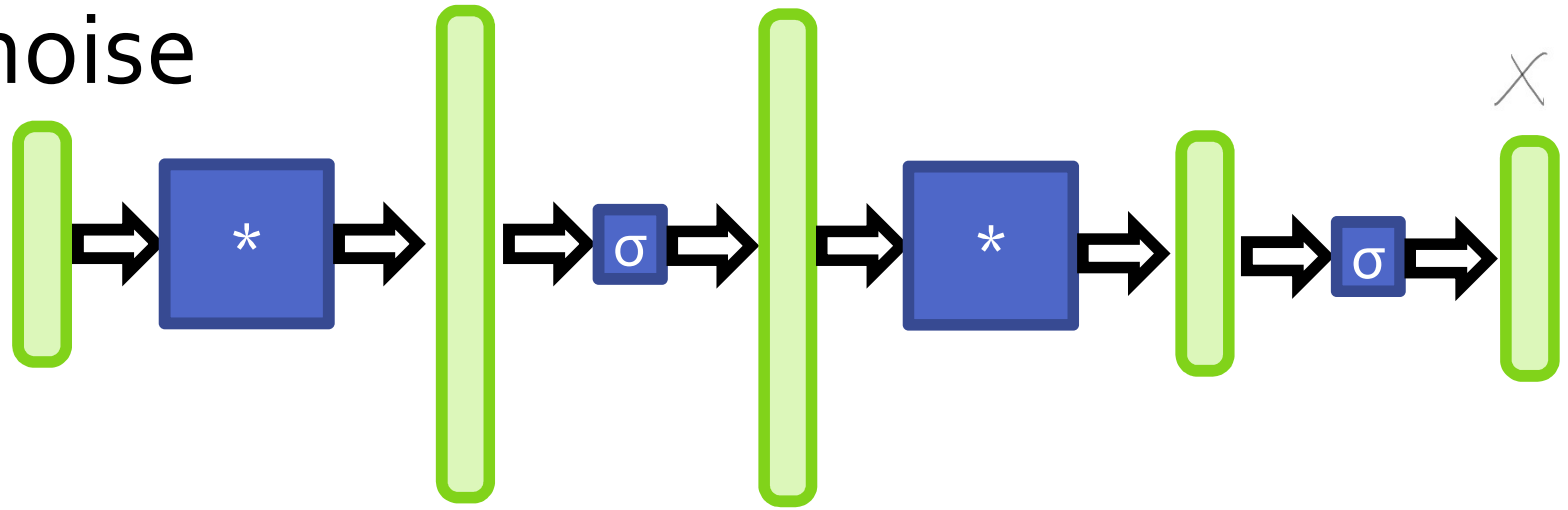
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

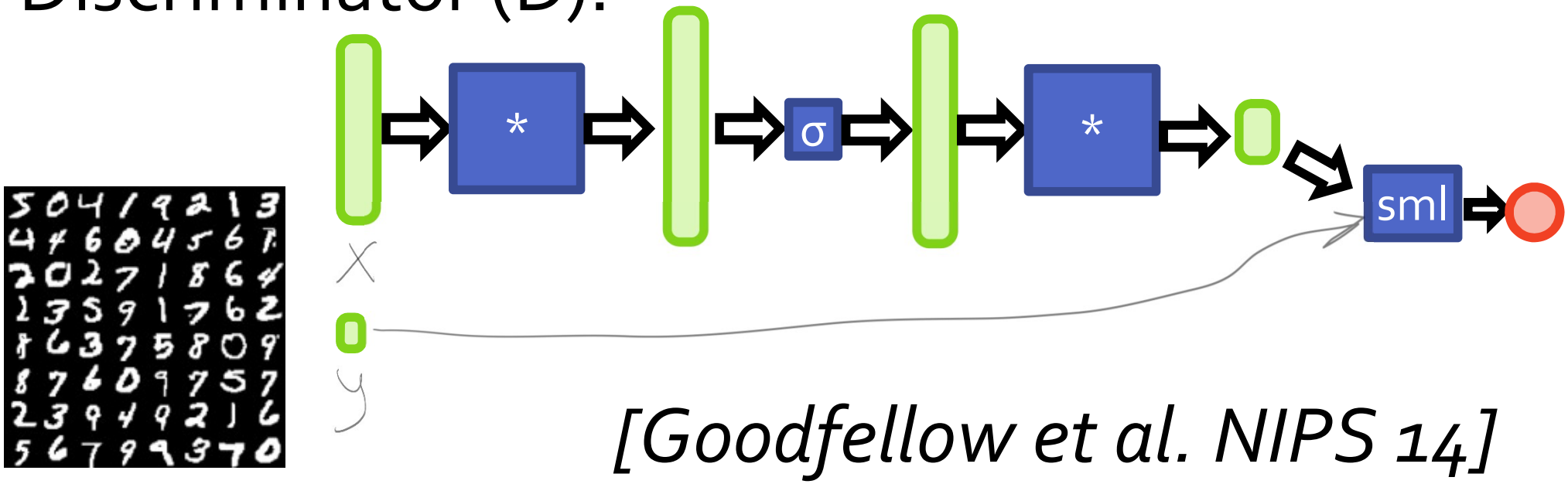
[Goodfellow et al. NIPS 14]

Adversarial generative networks

Generator (G):



Discriminator (D):

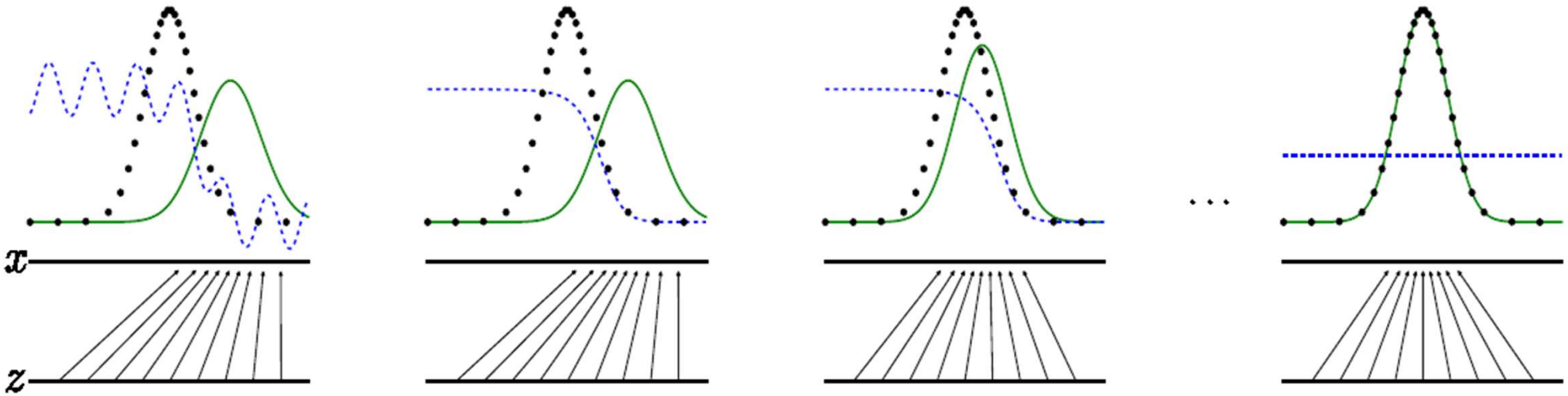


[Goodfellow et al. NIPS 14]

Adversarial learning

Поиск игрового равновесия:

$$\min_G \max_D V(D,G) = E_{x \sim p_D(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$



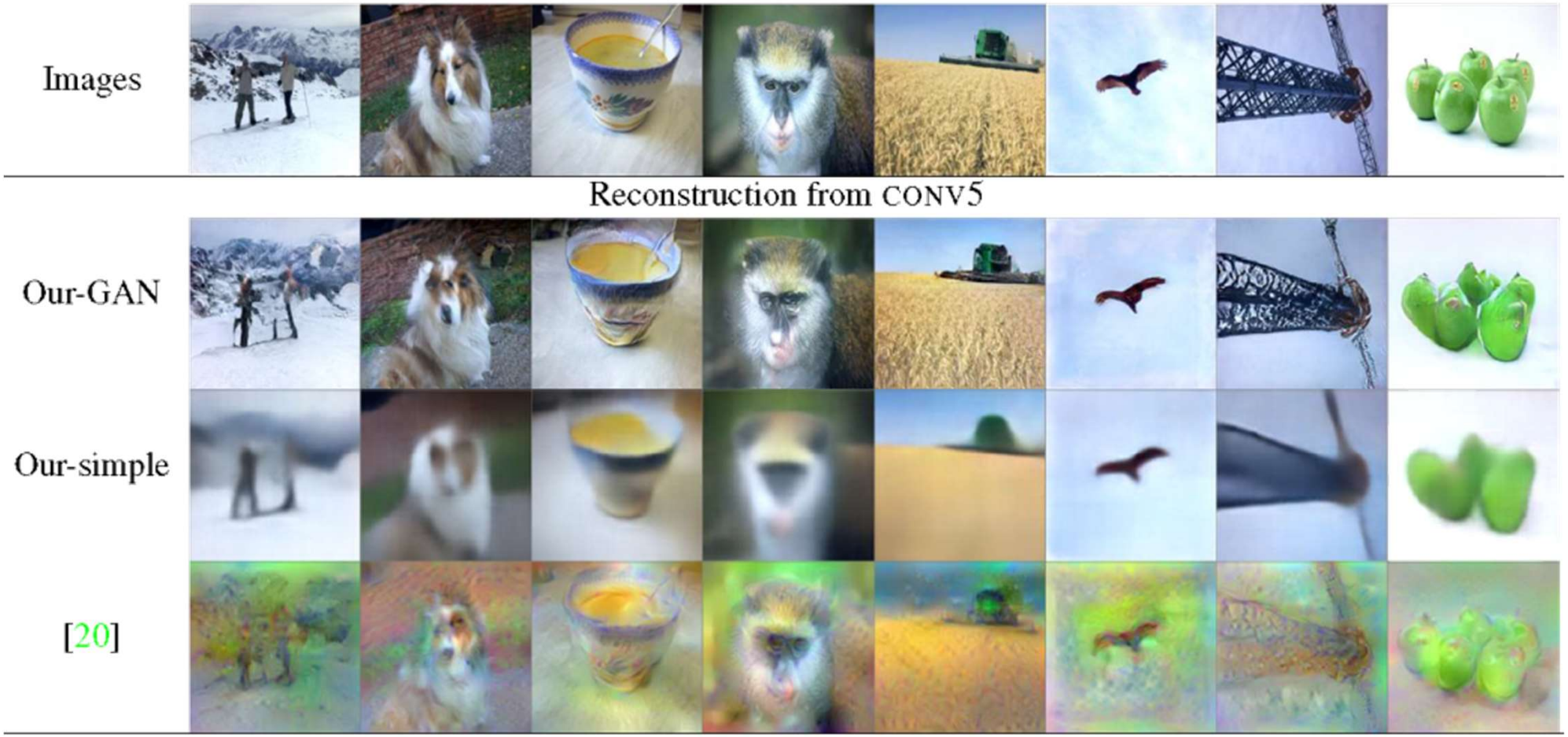
[Goodfellow et al. NIPS 14]

Adversarial networks

[Goodfellow et al. NIPS 14]

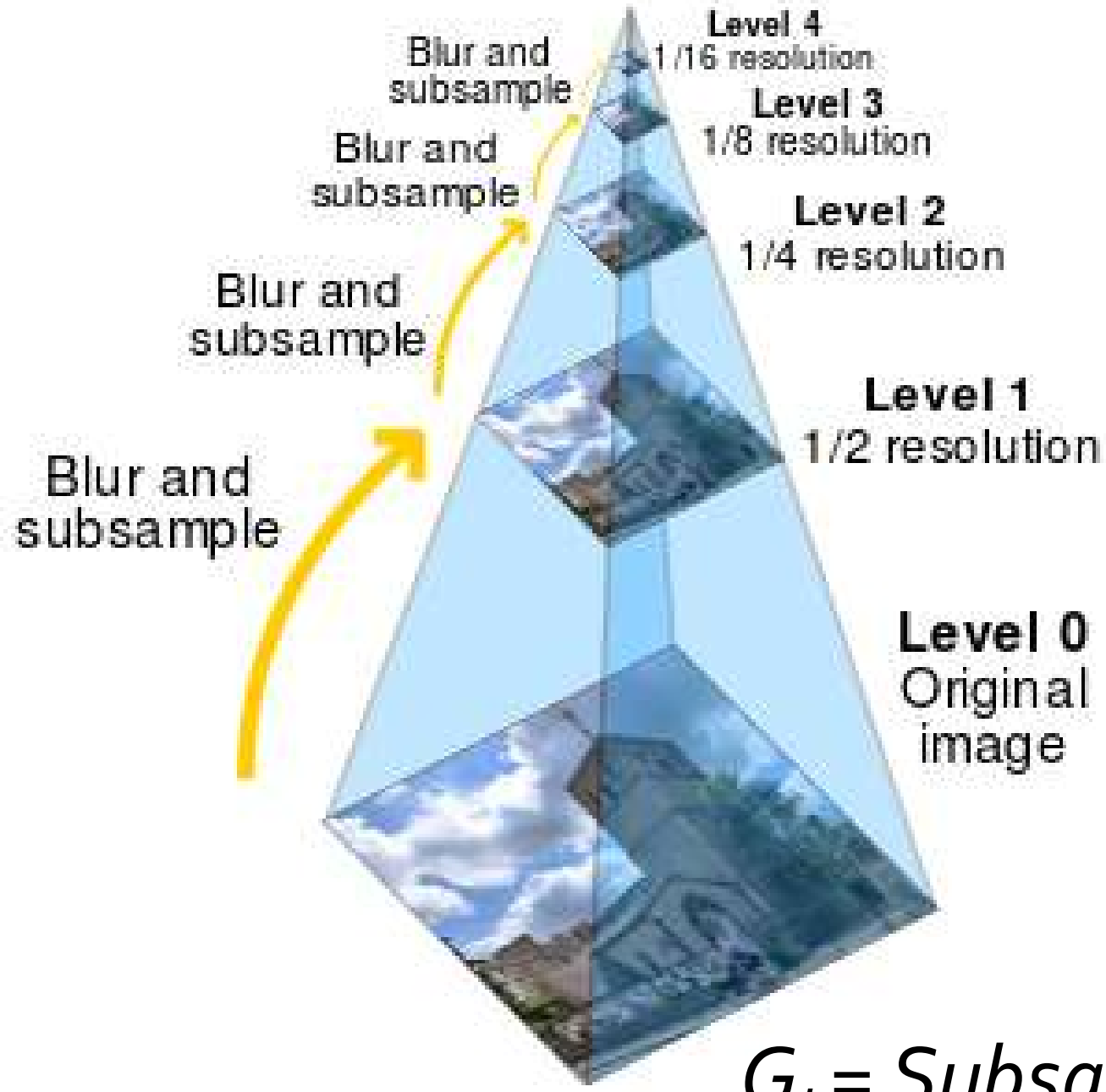
Adversarial loss

Adding adversarial loss greatly improves the sharpness:



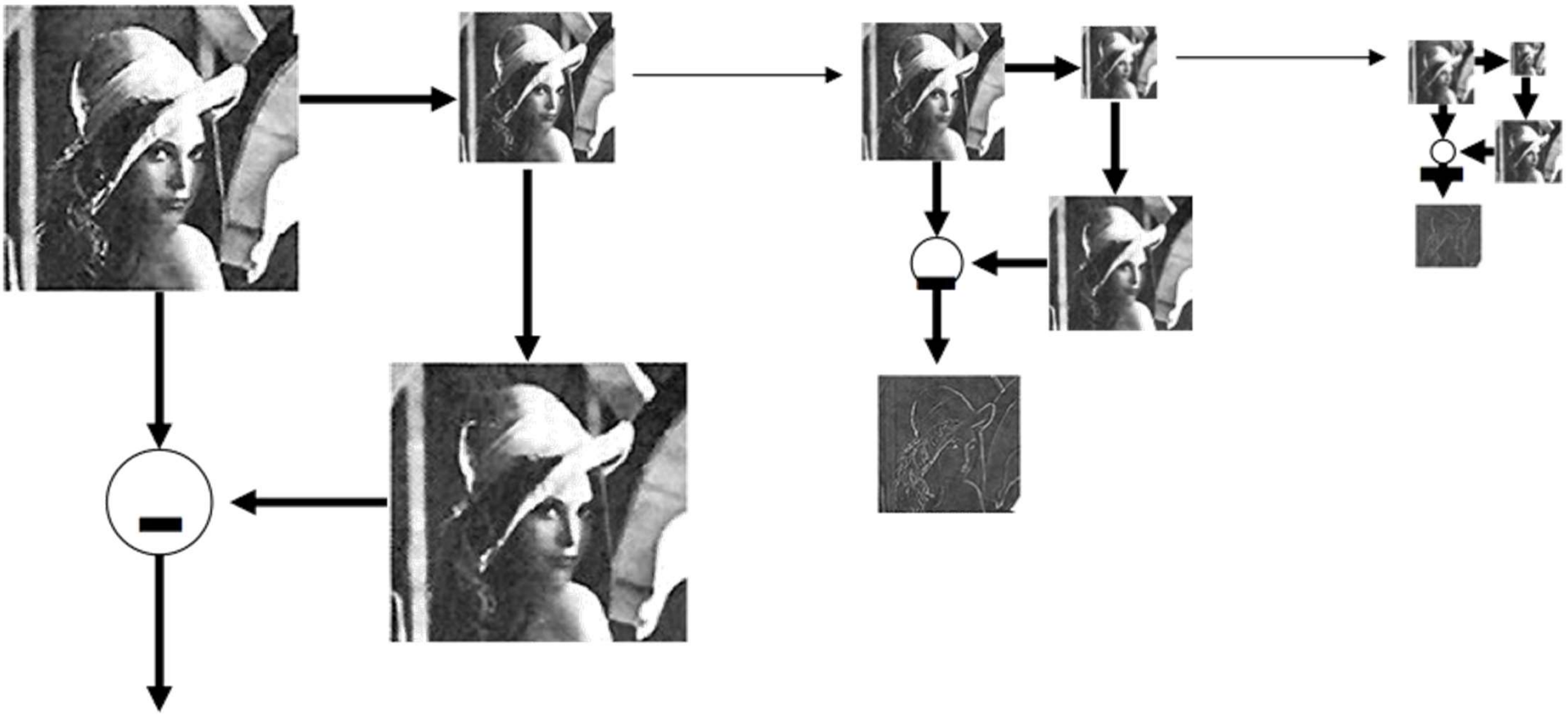
[Dosovitskiy and Brox, Arxiv15]

Gaussian pyramid



$$G_i = Subsample(Blur(G_i\text{-}1))$$

Laplacian pyramid

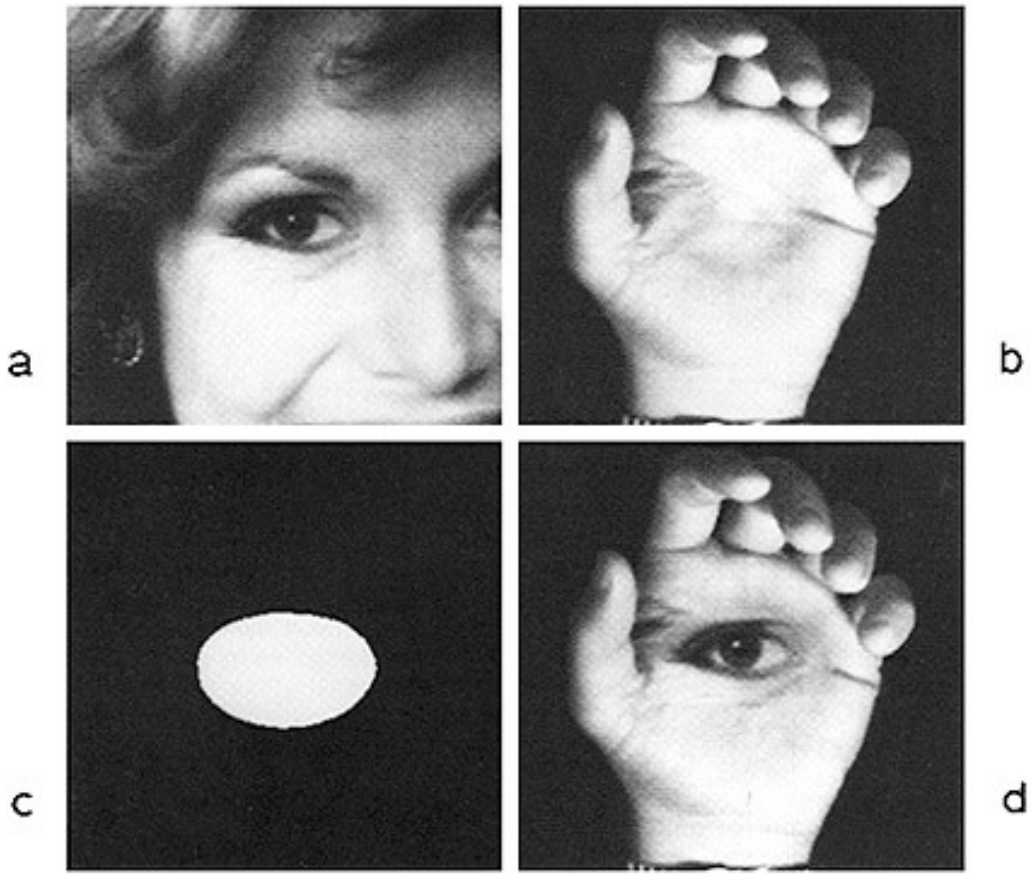


$$L_i = G_i - Upsample(G_{i+1})$$

- Classical multi-scale image model

[Burt Adelson IEEE TC83]

Multiresolutional splining

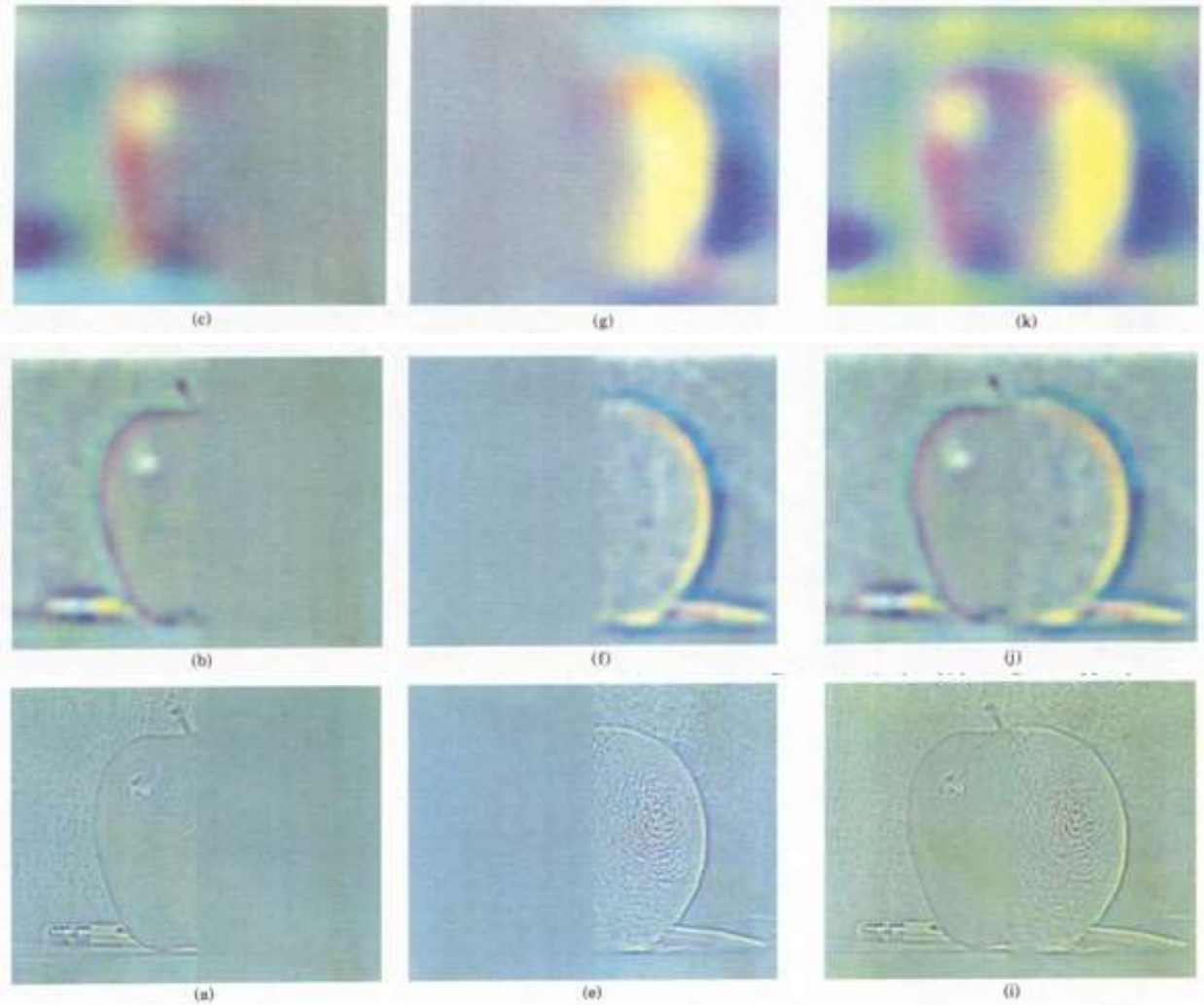


$$L_i = L^1_i G^M_i + L^2_i (1-G^M_i)$$

[Burt Adelson TOG83]

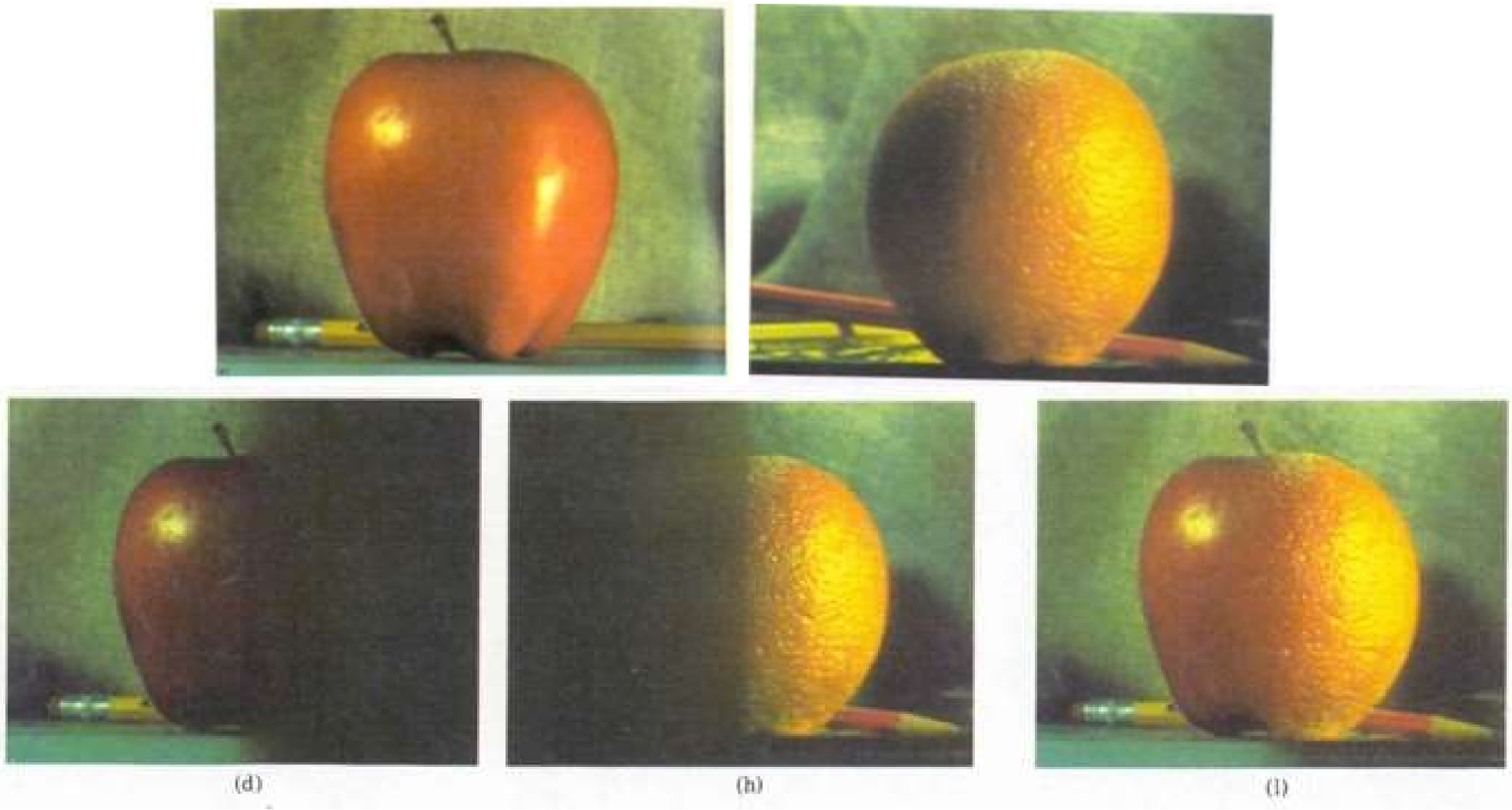
Multiresolutional splining

$$L_i = L^1_i G^M_i + L^2_i (1 - G^m_i)$$



[Burt Adelson TOG83]

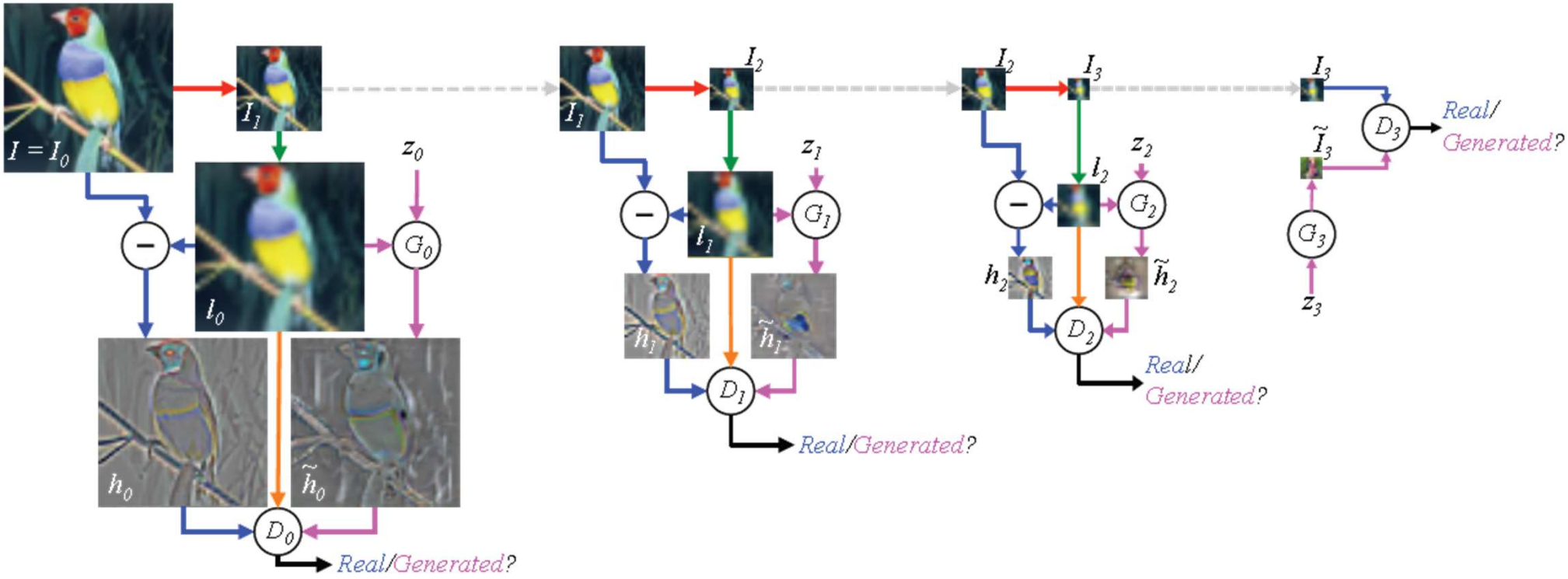
Multiresolutional splining



$$L_i = L^1{}_i G^M{}_i + L^2{}_i (1\text{-}G^m{}_i)$$

[Burt Adelson TOG83]

Applying GAN in Laplacian fashion



- GAN are applied at each level to generate L_i
- Generator accepts noise and Upsample(G_{i+1})
- At testtime, L_i is generated and G_i is recovered

$$G_i = L_i + Upsample(G_{i+1})$$

[Denton et al. ICLR15]

Applying GAN in Laplacian fashion

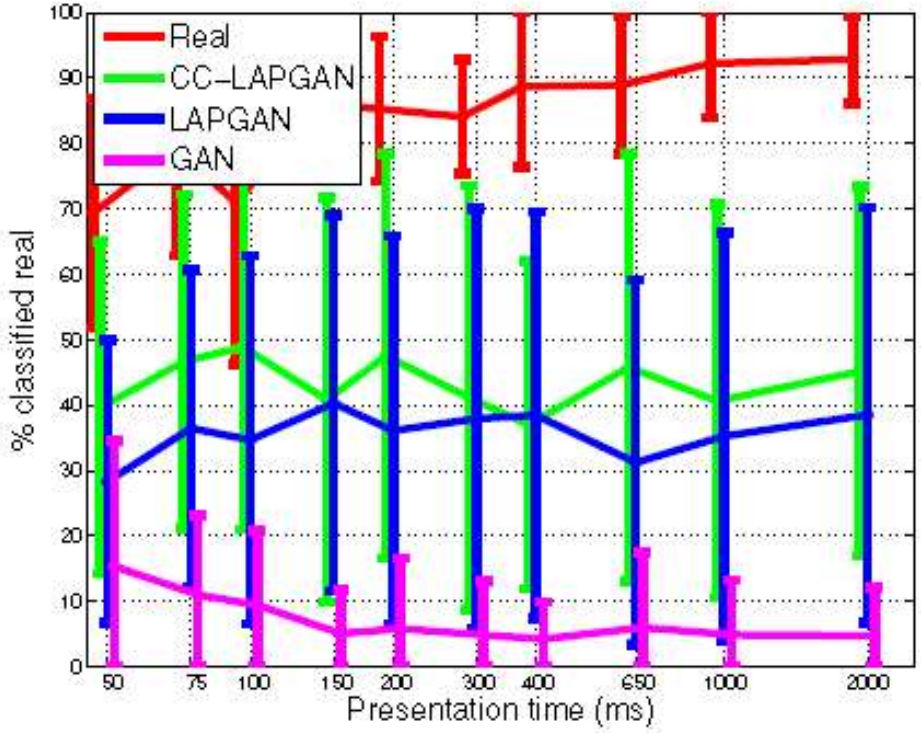
[Denton et al. ICLR15]

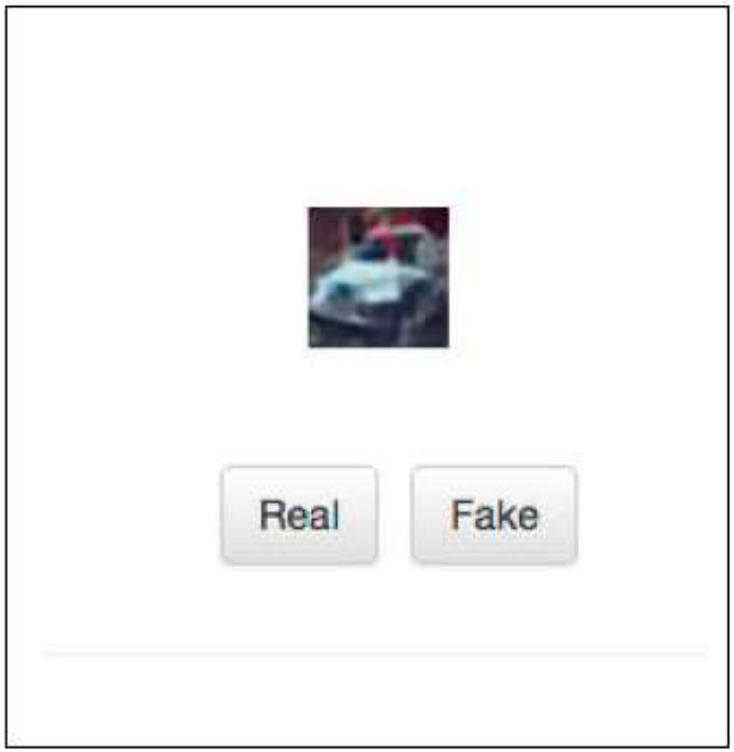
Applying GAN in Laplacian fashion

[Denton et al. ICLR15]

Assessing realism

Ultimate measure: user study

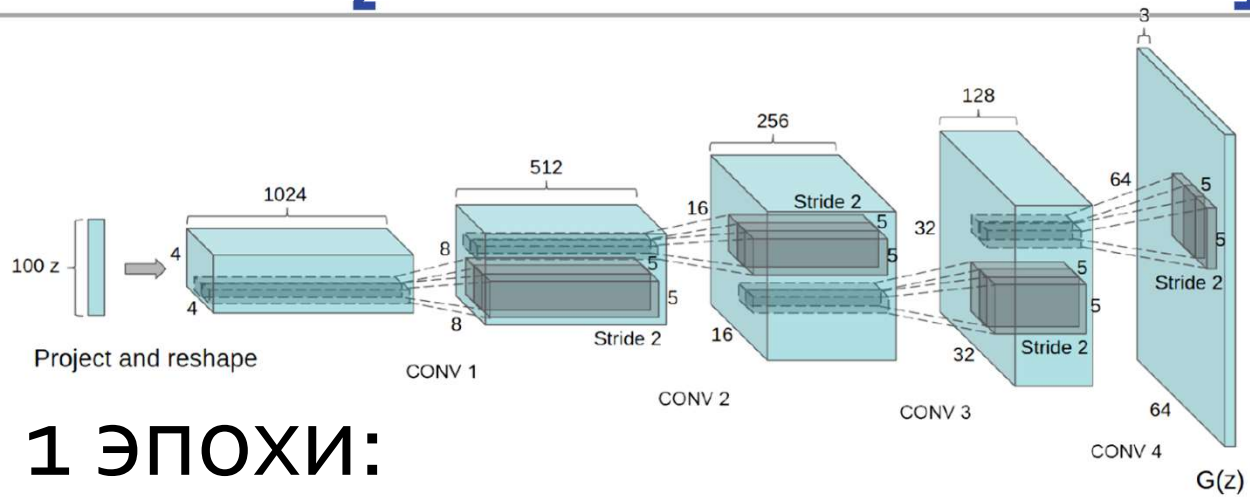




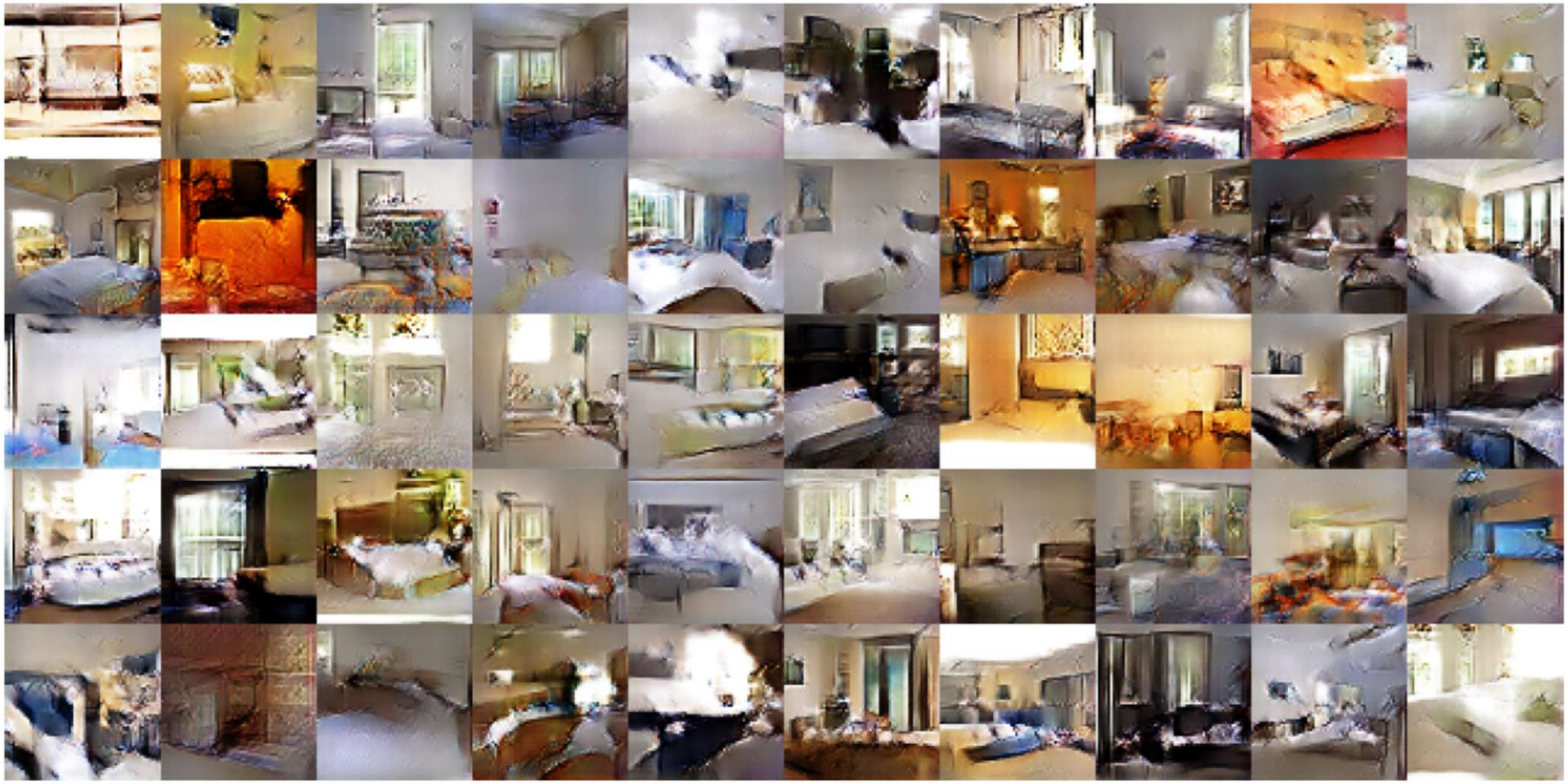
Открытая проблема: каким образом можно автоматизировать измерение качества?

[Denton et al. ICLR15]

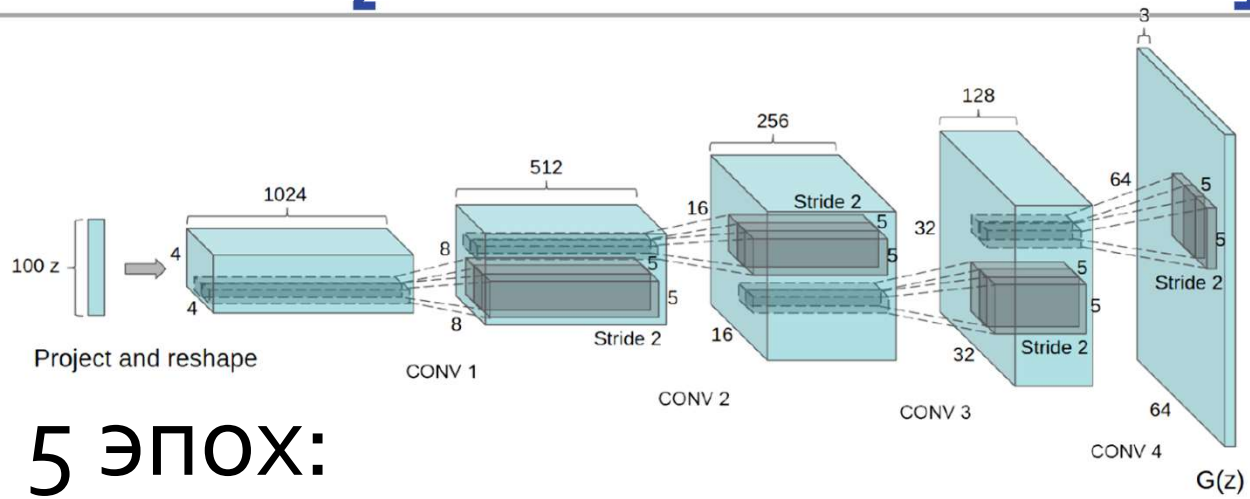
DCGAN [Radford et al. 2016]



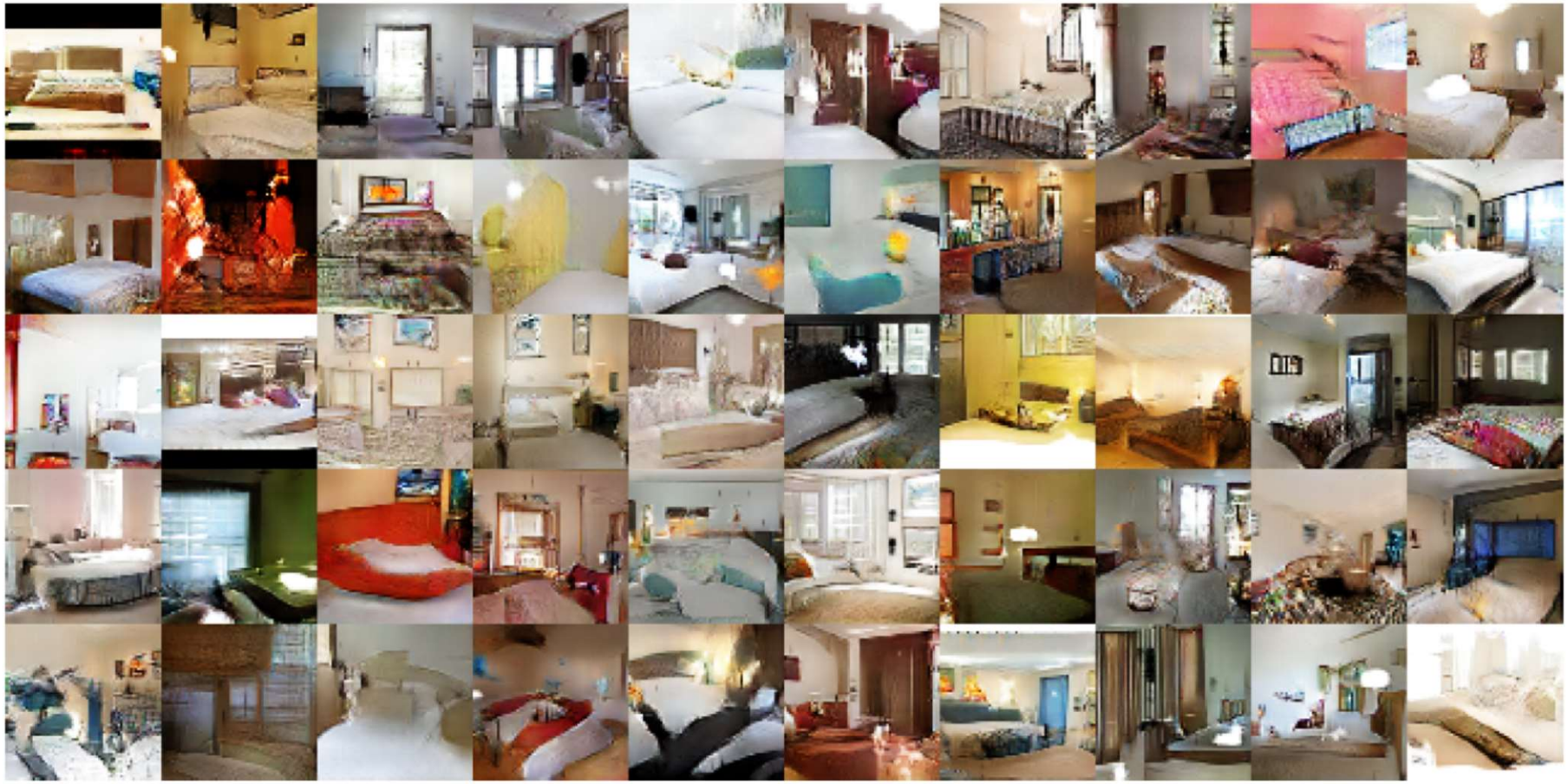
После 1 эпохи:



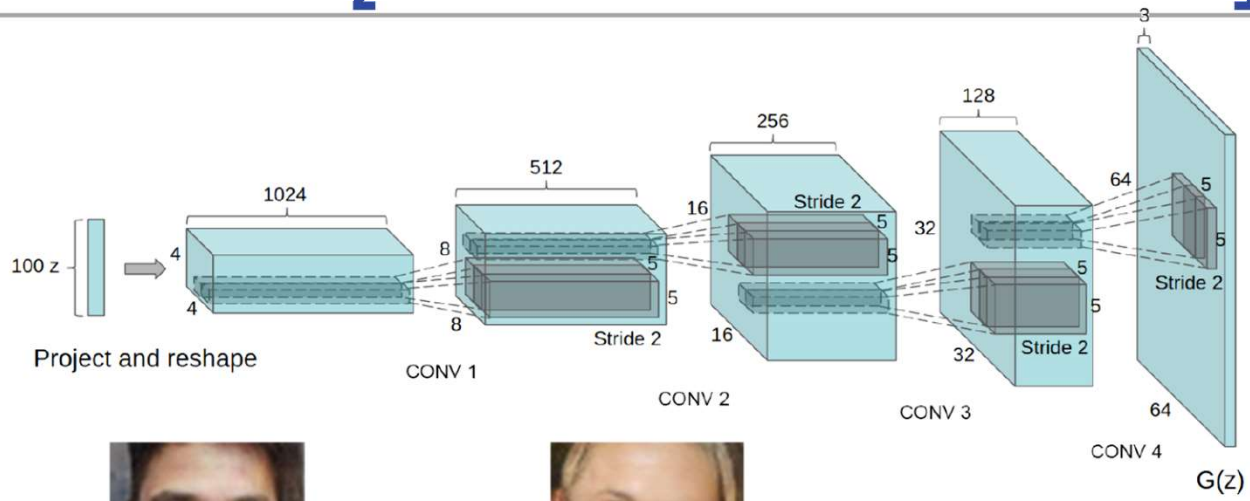
DCGAN [Radford et al. 2016]

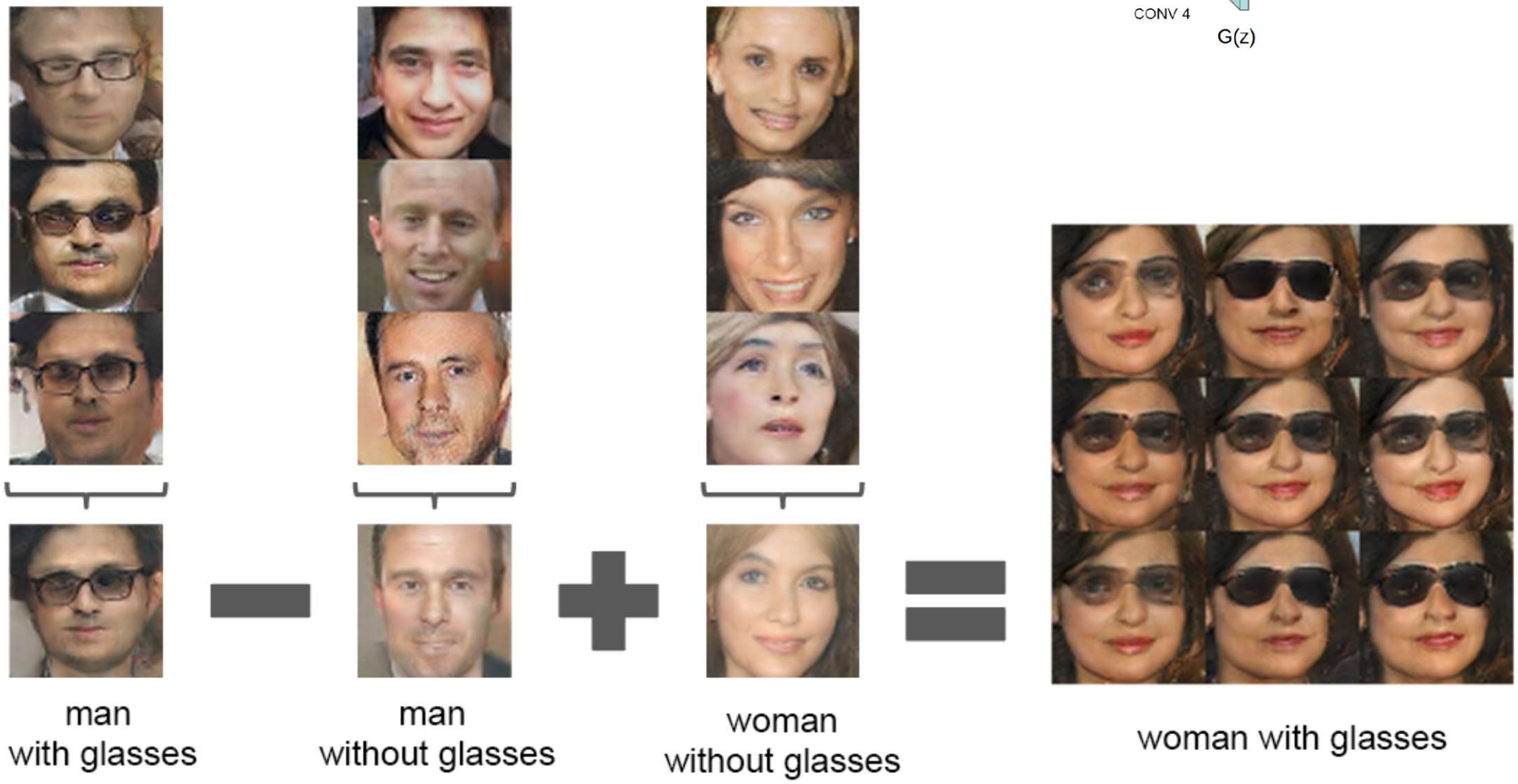


После 5 эпох:

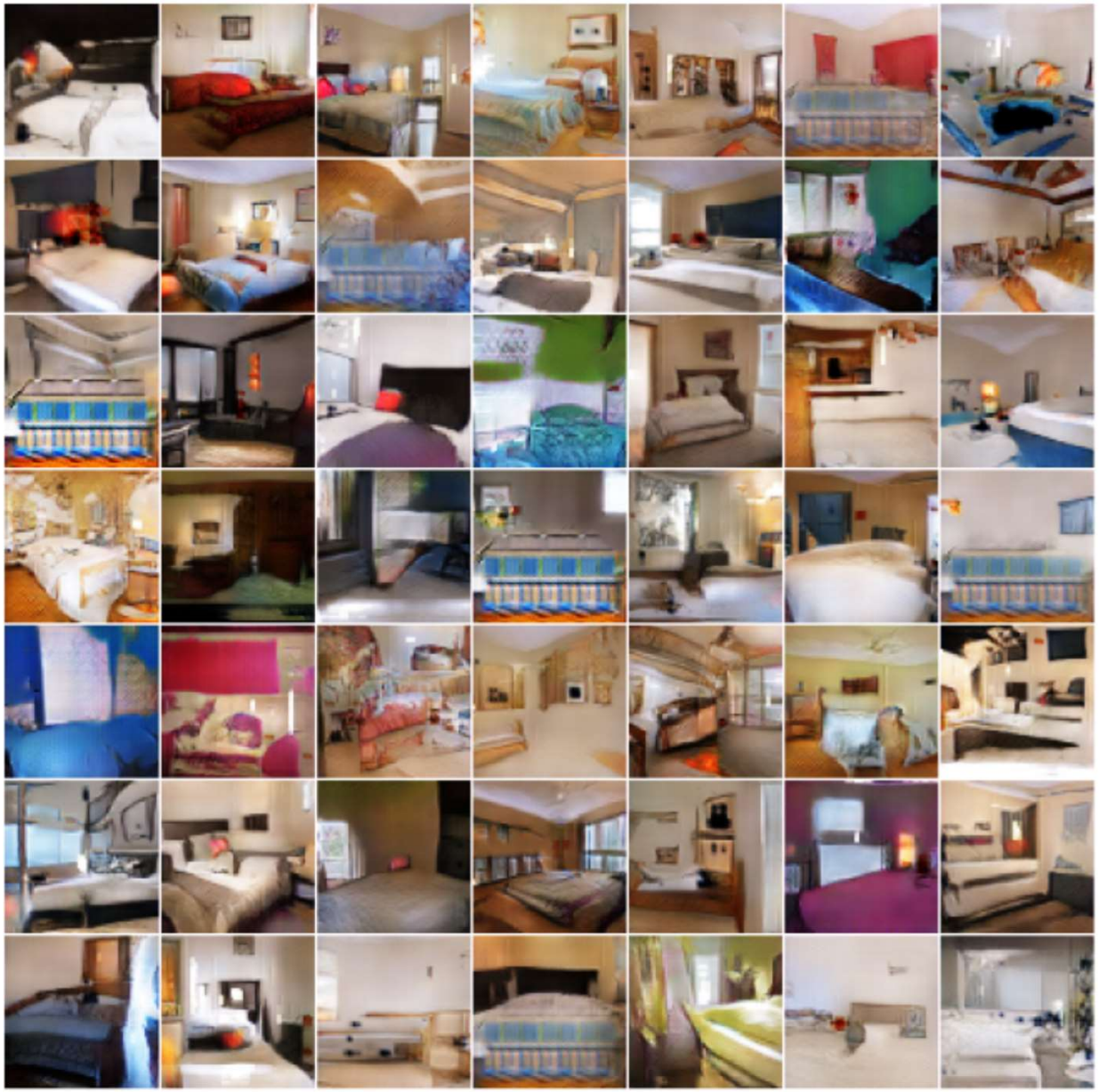


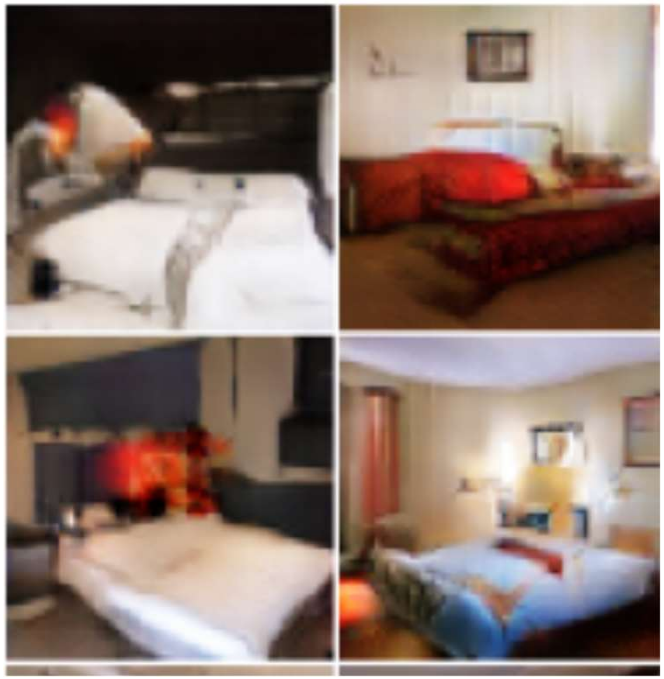
DCGAN [Radford et al. 2016]





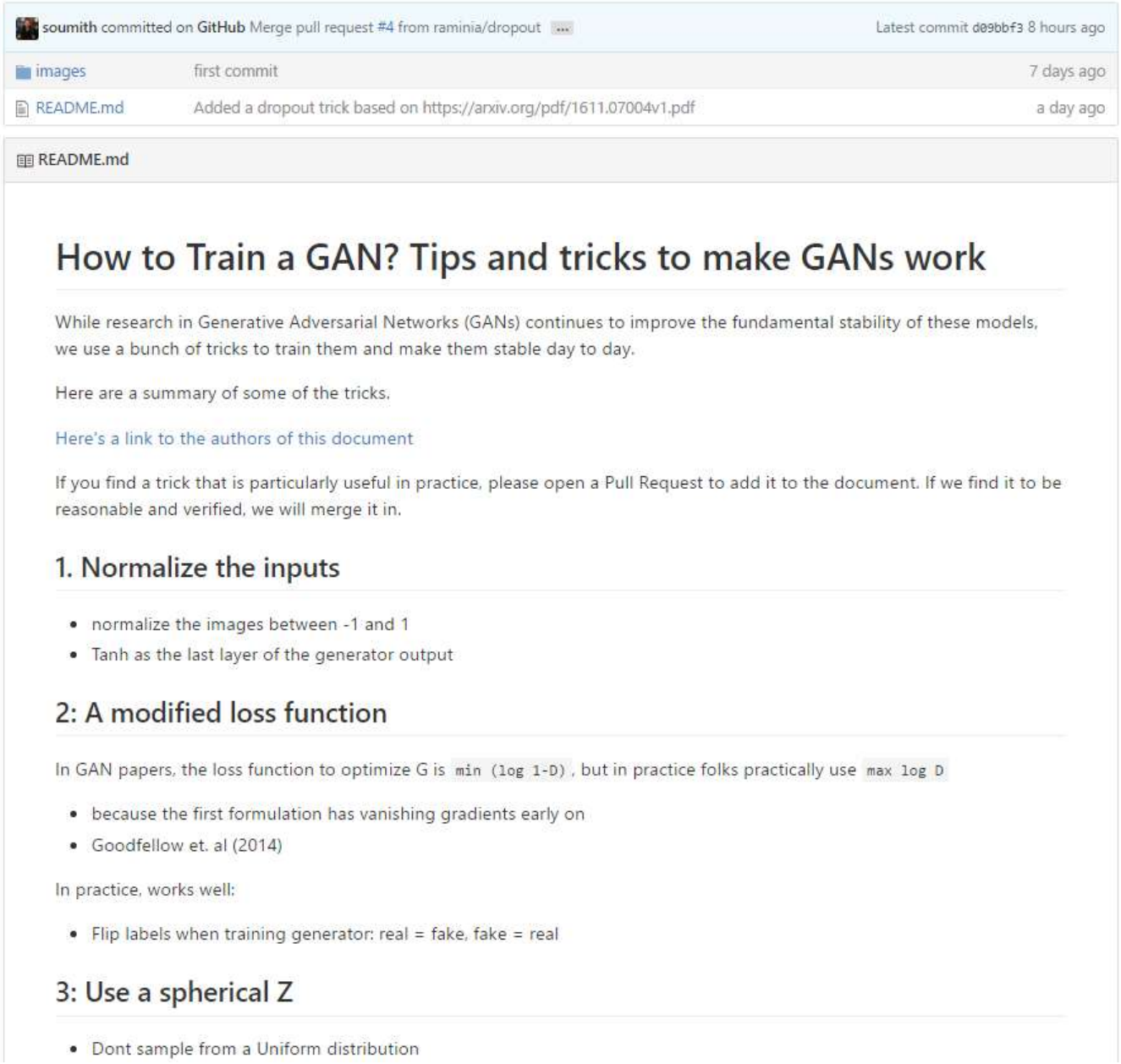
DCGAN [Radford et al. 2016]





DCGAN reproduced by [Zhao et al.2016]

Оптимизационные эвристики



soumith committed on GitHub Merge pull request #4 from raminia/dropout ... Latest commit d09bbf3 8 hours ago

images	first commit	7 days ago
README.md	Added a dropout trick based on https://arxiv.org/pdf/1611.07004v1.pdf	a day ago

README.md

How to Train a GAN? Tips and tricks to make GANs work

While research in Generative Adversarial Networks (GANs) continues to improve the fundamental stability of these models, we use a bunch of tricks to train them and make them stable day to day.

Here are a summary of some of the tricks.

Here's a link to the authors of this document

If you find a trick that is particularly useful in practice, please open a Pull Request to add it to the document. If we find it to be reasonable and verified, we will merge it in.

1. Normalize the inputs

- normalize the images between -1 and 1
- Tanh as the last layer of the generator output

2: A modified loss function

In GAN papers, the loss function to optimize G is `min (log 1-D)`, but in practice folks practically use `max log D`

- because the first formulation has vanishing gradients early on
- Goodfellow et. al (2014)

In practice, works well:

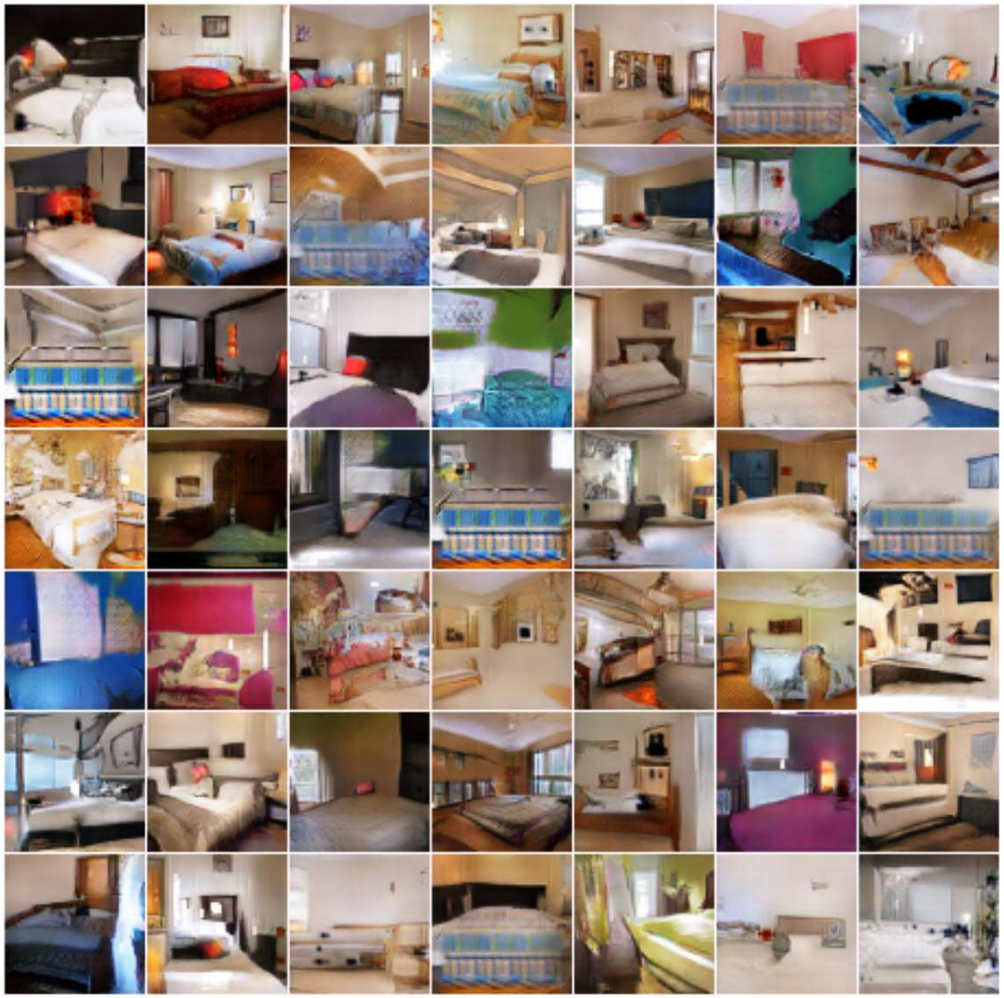
- Flip labels when training generator: real = fake, fake = real

3: Use a spherical Z

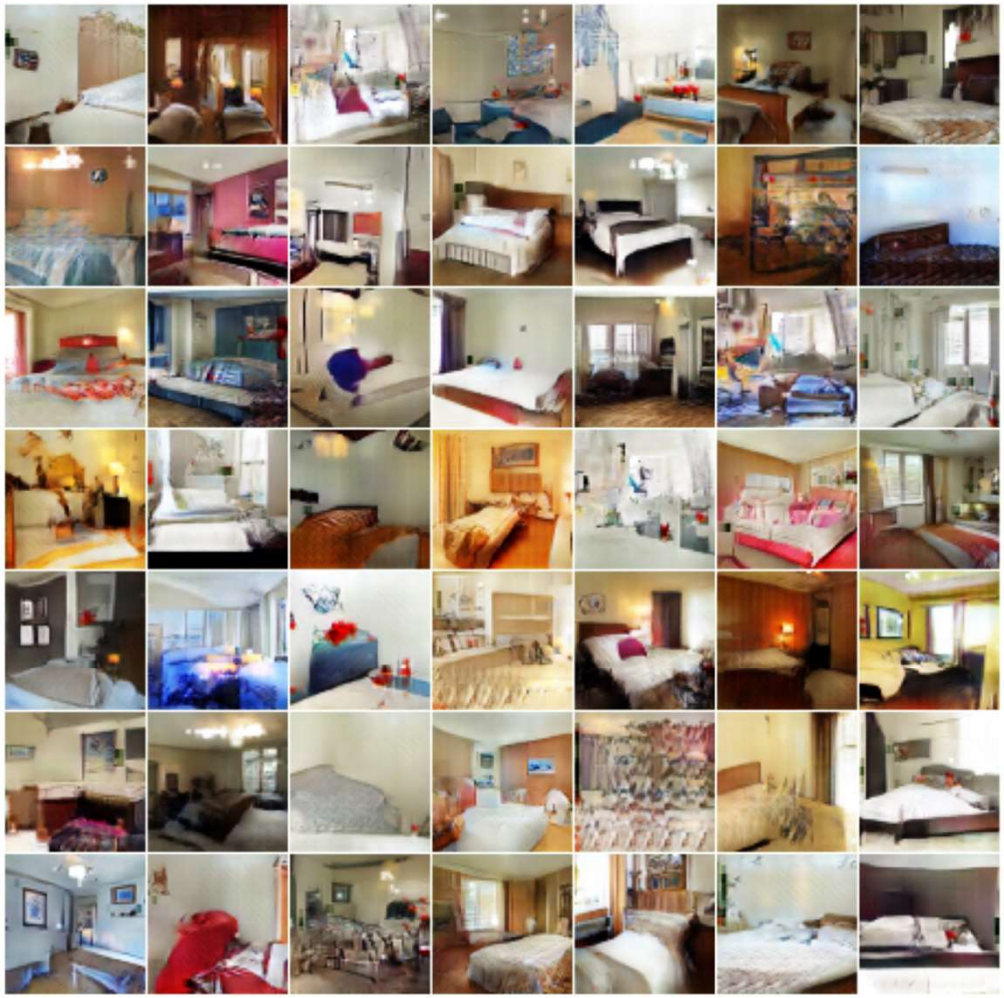
- Dont sample from a Uniform distribution

https://github.com/soumith/ganhacks

State-of-the-art: EBGAN [Zhao et al. 2016]



DCGAN

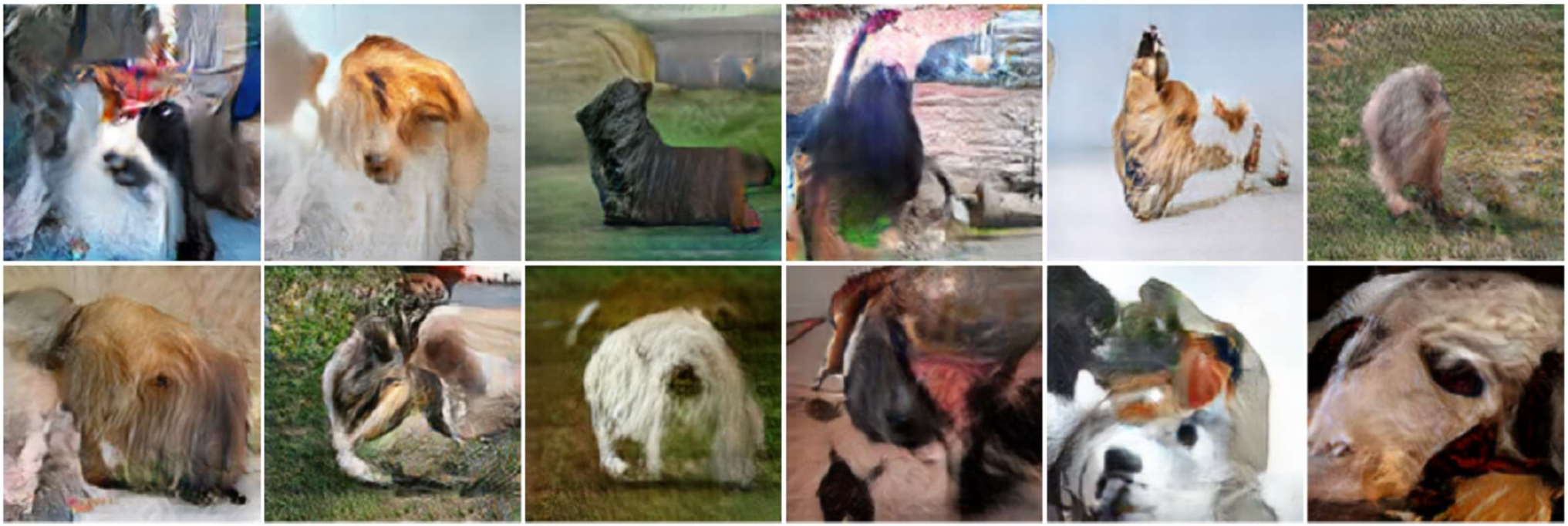


EBGAN

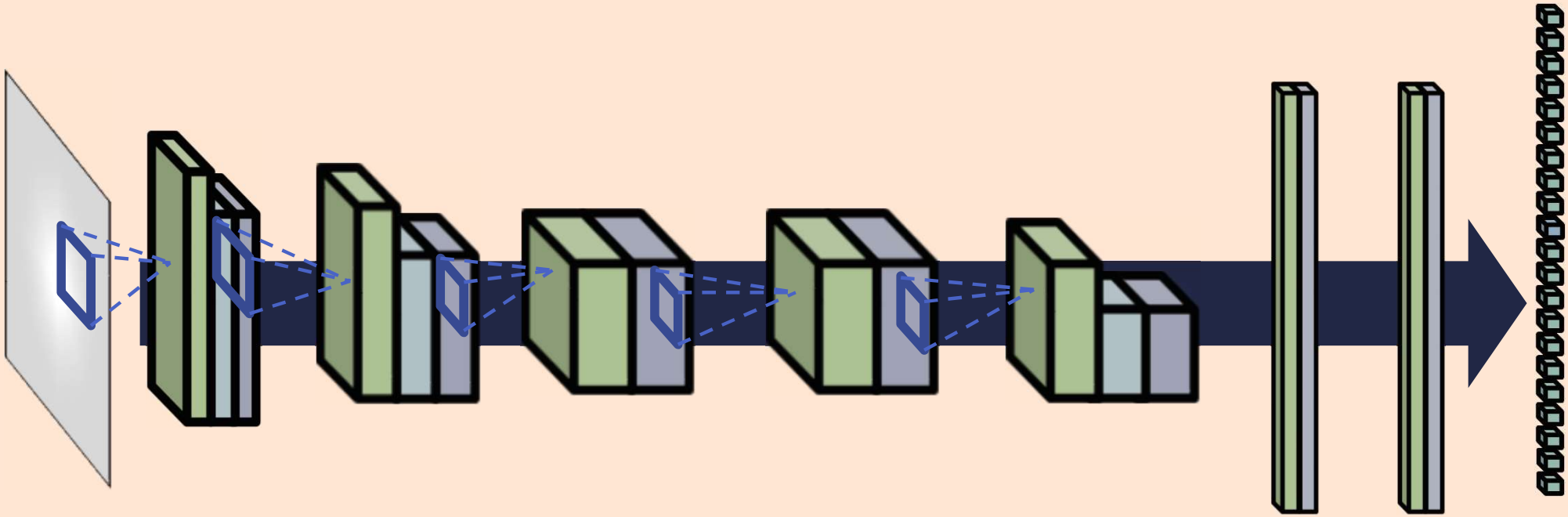
$$f_{PT}(S) = \frac{1}{N(N-1)} \sum_i \sum_{j \neq i} \left(\frac{S_i^\top S_j}{\|S_i\| \|S_j\|} \right)^2$$

State-of-the-art: EBGAN [Zhao et al. 2016]

1000-классовая генерация 256x256:



Deep supervised neural networks



- are a “big thing” in computer vision and beyond
- **are hungry for labeled data**

Where to get the data?

Lots of modalities do not have large labeled data sets:

- Biomedical
- Unusual cameras / image types
- Videos
- Data with expert-level annotation (not mTurkable)
- ….

Surrogate training data often available:

- Borrow from adjacent modality
- Generate synthetic imagery (computer graphics)
- Use data augmentation to amplify data (*image-based rendering, morphing, re-synthesis,….*)

Resulting training data are shifted. *Domain adaptation* needed.

Example: Internet images -> Webcam sensor

[Saenko et al. ECCV2010]

Example: (semi-)synthetic to real

Assumptions and goals

- Lots of *labeled* data in the *source domain* (e.g. synthetic images)
- Lots of *unlabeled* data in the *target domain* (e.g. real images)
- **Goal**: train a **deep neural net** that does well on the **target domain**

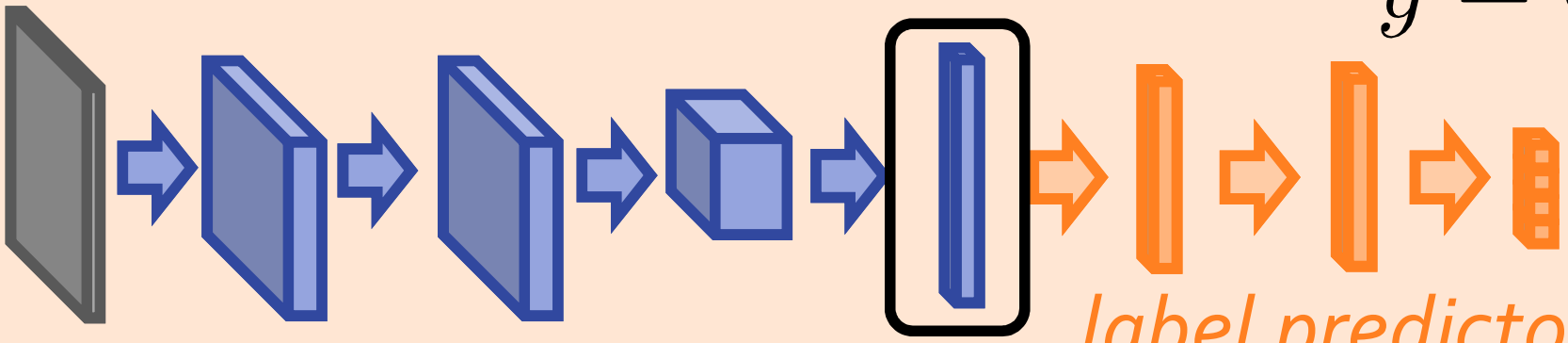
Large-scale deep unsupervised domain adaptation

Domain shift in a deep architecture

$$\mathbf{f} = G_f(\mathbf{x}; \theta_f)$$

$$y = G_y(\mathbf{f}; \theta_y)$$

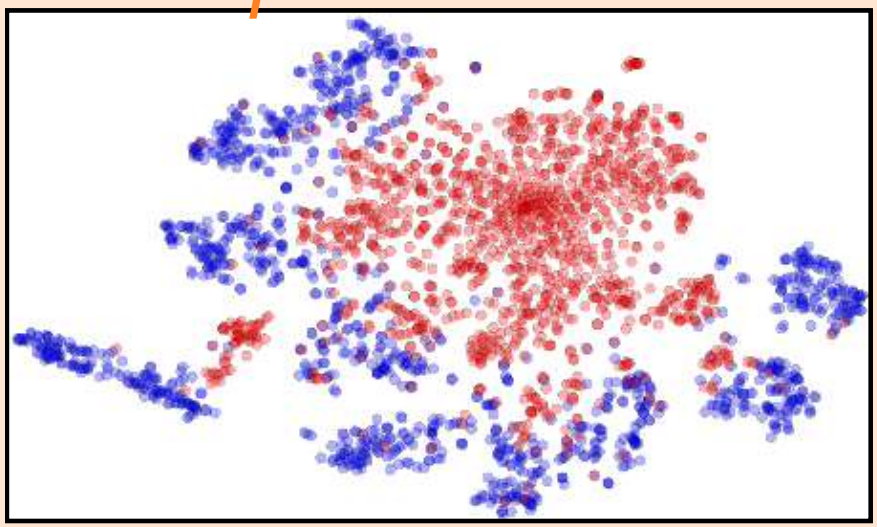
$\mathbf{x}$



feature extractor

label predictor

When trained on source only, feature distributions do not match:

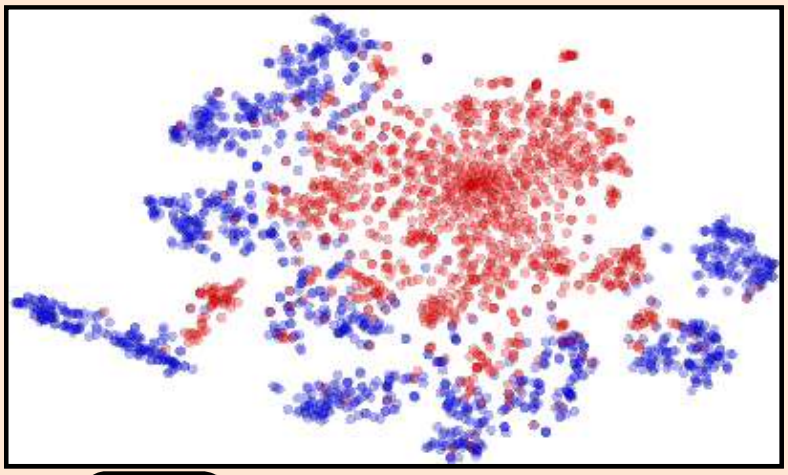


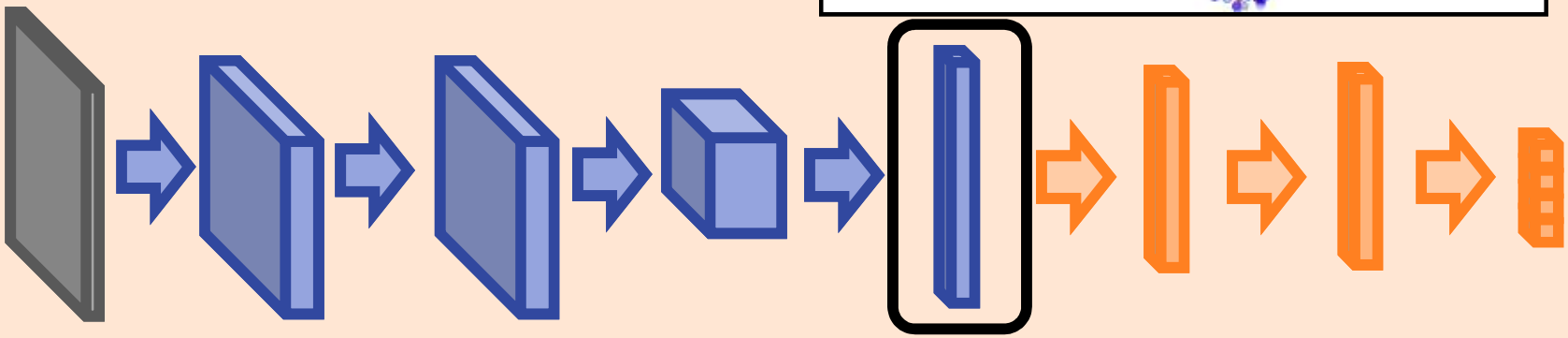
$$S(\mathbf{f}) = \{G_f(\mathbf{x}; \theta_f) \,|\, \mathbf{x} \sim S(\mathbf{x})\}$$

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

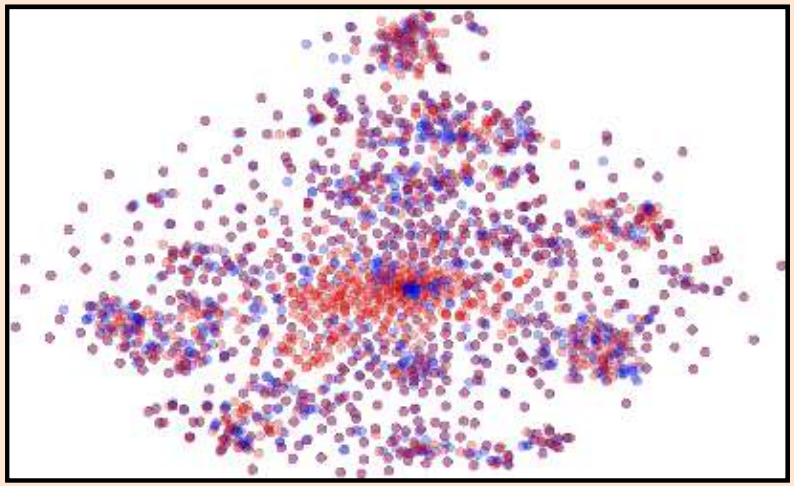
Idea 1: domain-invariant features wanted

Feature distribution without adaptation:

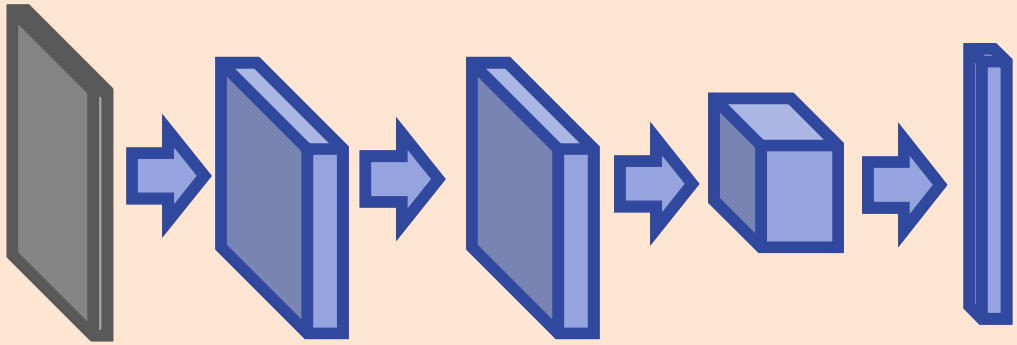


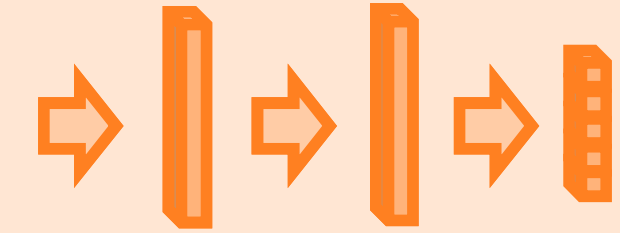


Our goal (after adaptation):



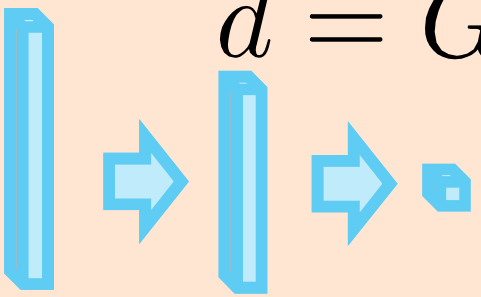
Idea 2: measuring domain shift

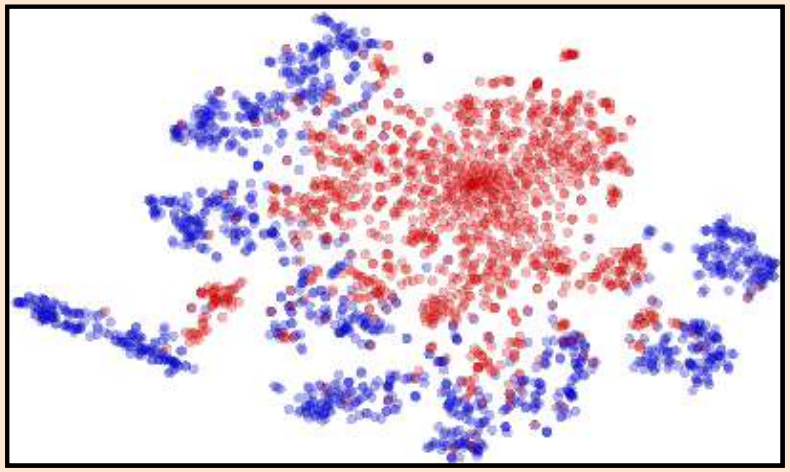




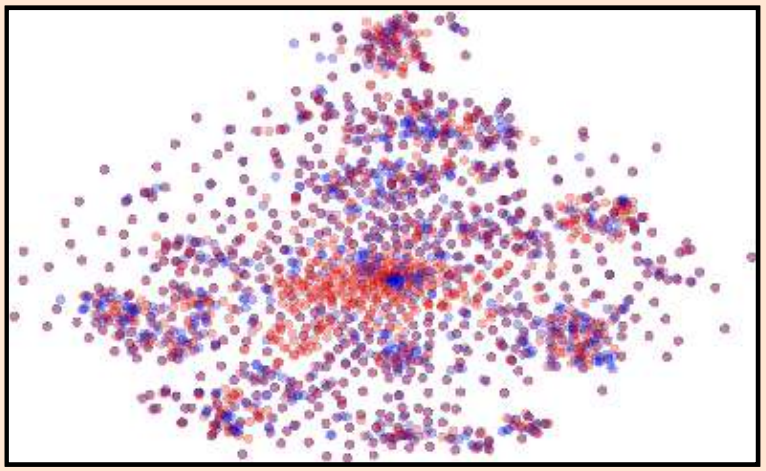
$$d = G_d(\mathbf{f}; \theta_d)$$

Domain classifier:



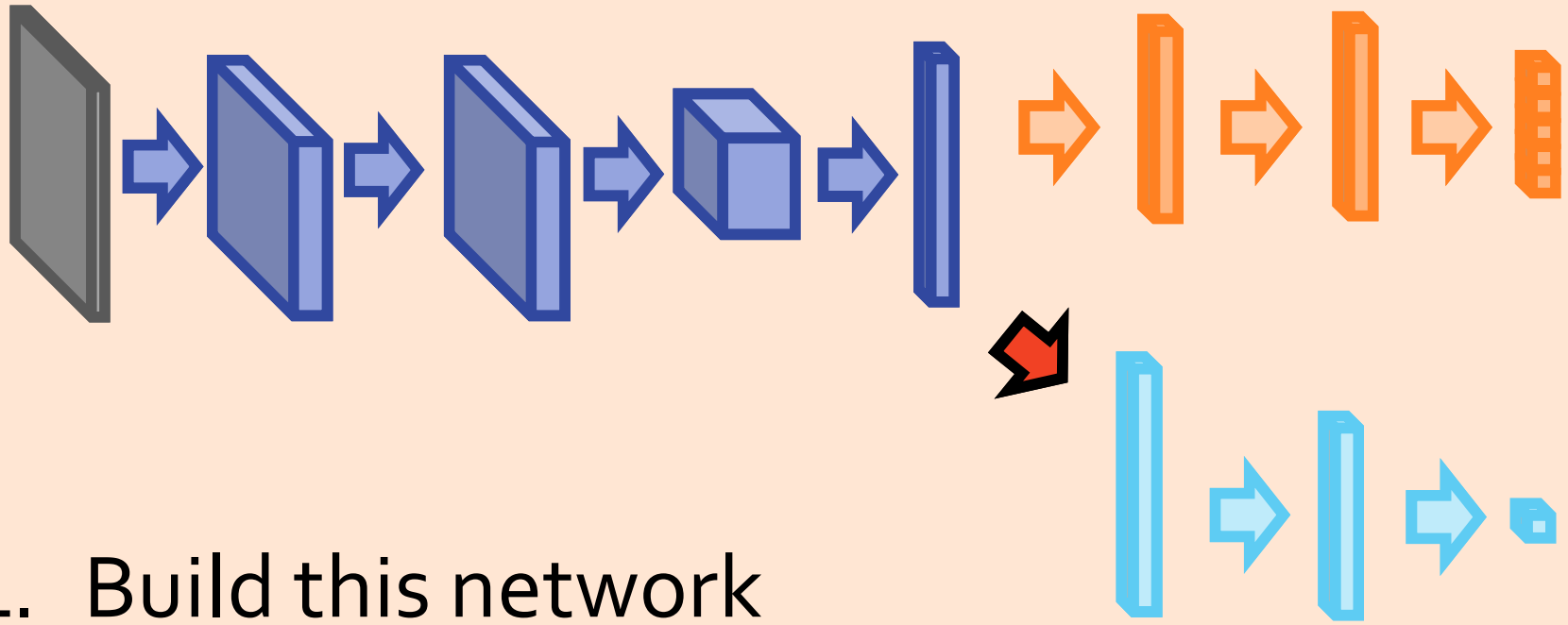


Domain loss low



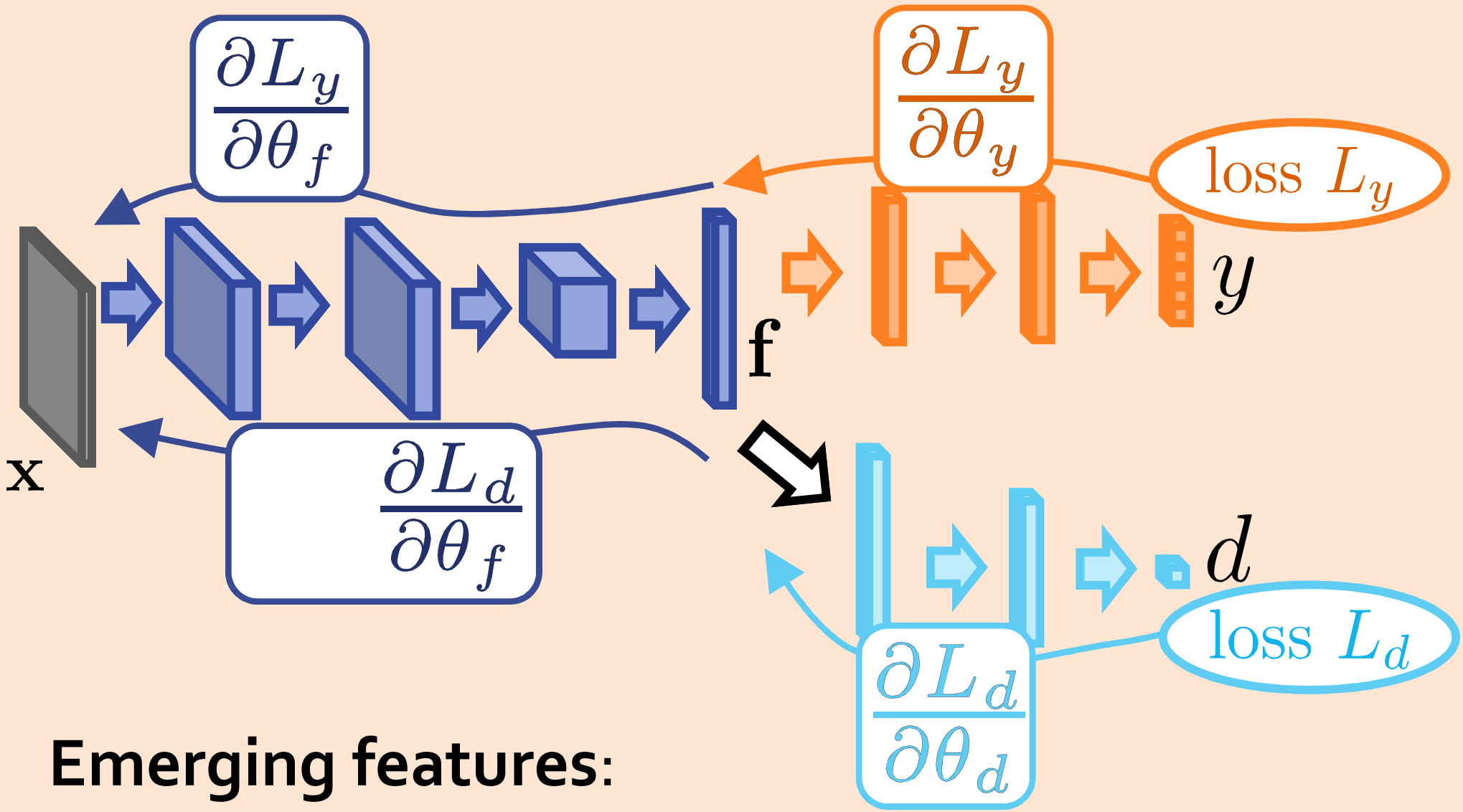
Domain loss high

Learning with adaptation



1. Build this network
2. Train feature extractor + class predictor on source data
3. Train feature extractor + domain classifier on source+target data
4. Use feature extractor + class predictor at test time

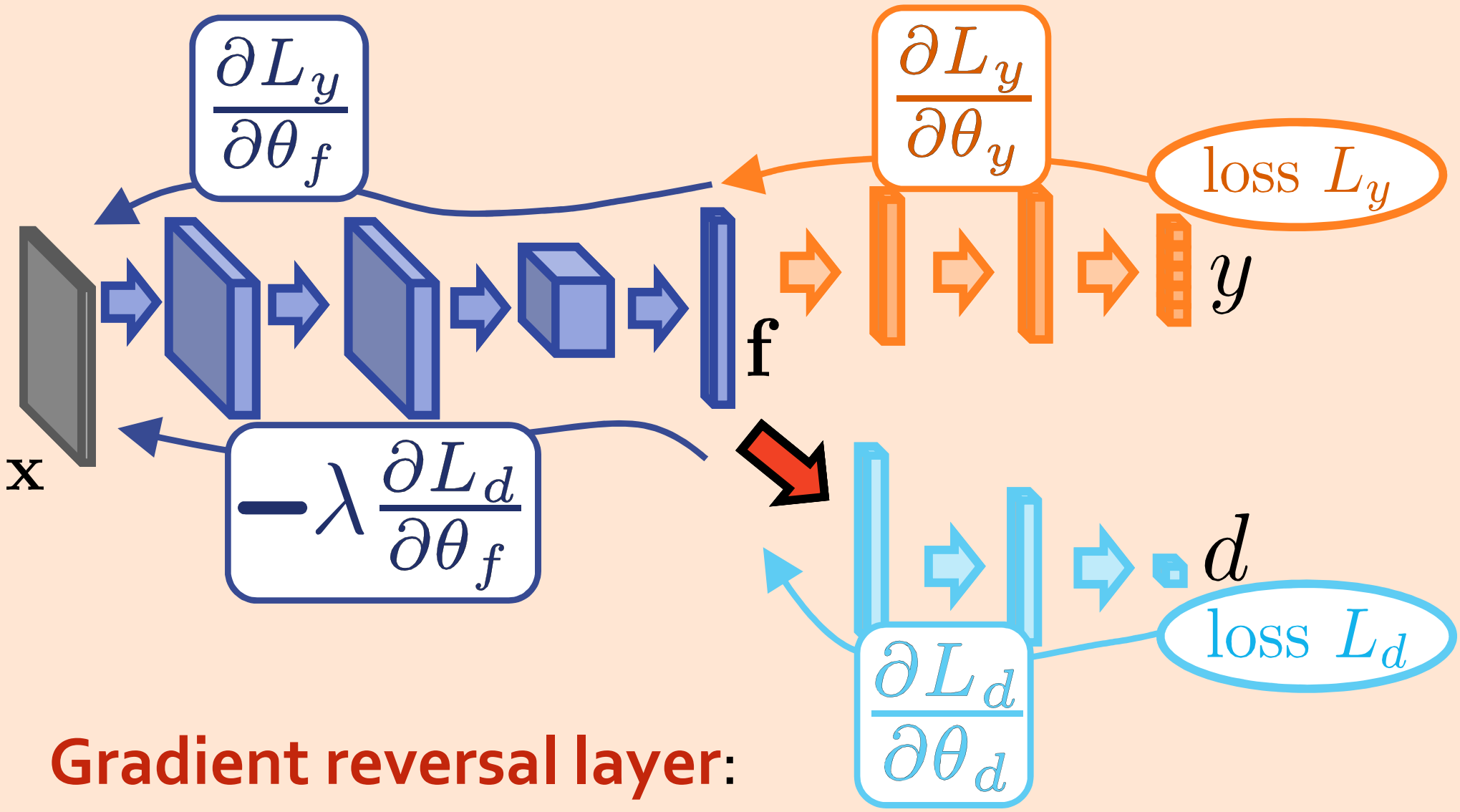
Idea 3: minimizing domain shift



Emerging features:

- Discriminative (good for predicting *y*)
- Domain-discriminative (good for predicting *d*)

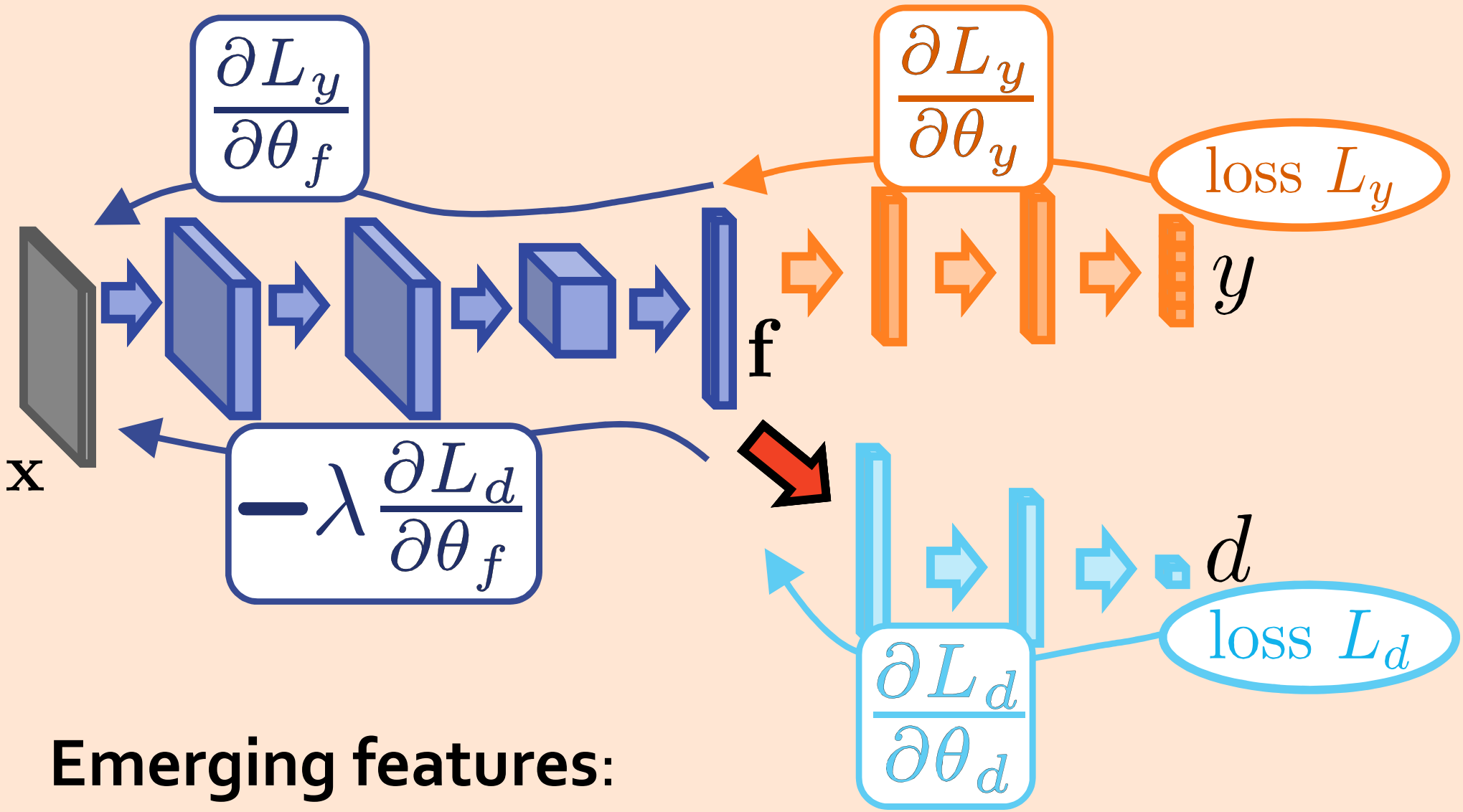
Idea 3: minimizing domain shift



Gradient reversal layer:

- Copies data without change at forwardprop
- Multiplies the gradient by -λ at backprop

Idea 3: minimizing domain shift



Emerging features:

- Discriminative (good for predicting *y*)
- Domain-invariant (not good for predicting *d*)

Gradient reversal layer

```
class GradReversalLayer : Layer {

        float lambda;

        blob forward (blob x)       {
                return x
        }

        blob backward(blob dzdy) {
                return multiply(dzdy, -lambda)
        }
}
```

Saddle point interpretation

Our objective (small label prediction loss + large domain classification loss wanted)

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

The backprop converges to a saddle point:

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

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

Similar idea for generative networks:

[Goodfellow et al. Generative adversarial nets. In NIPS, 2014]

Backprop updates

Domain classification loss for the *i*th example

Label prediction loss for the *i*th example

$$\theta_f \longleftarrow \theta_f - \mu\left(\frac{\partial L_y^i}{\partial \theta_f} - \lambda\frac{\partial L_d^i}{\partial \theta_f}\right)$$

$$\theta_y \longleftarrow \theta_y - \mu\frac{\partial L_y^i}{\partial \theta_y}$$

$$\theta_d \longleftarrow \theta_d - \mu\frac{\partial L_d^i}{\partial \theta_d}$$

Office dataset

[Saenko et al. ECCV2010]

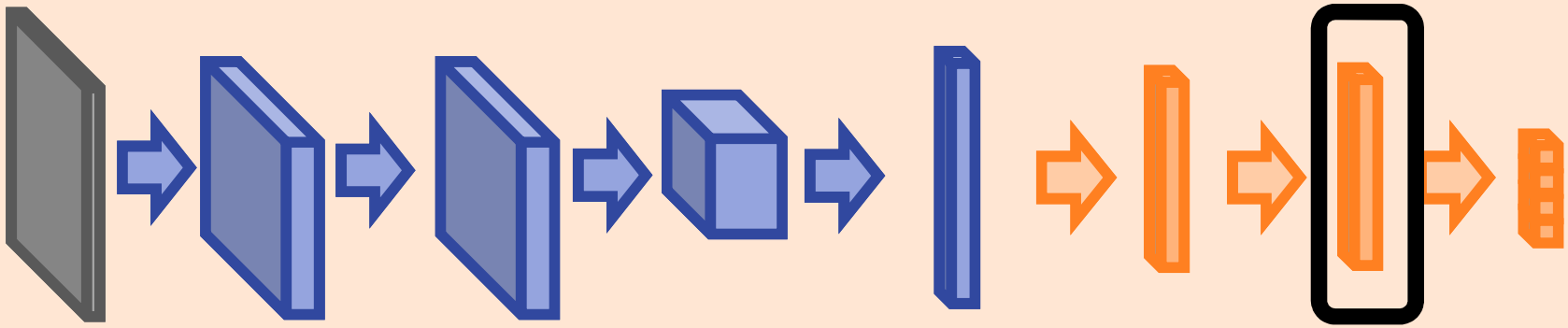
Results on Office dataset

Method / Source / Target	Amazon / Webcam	DSLR / Webcam	Webcam / DSLR
GFK(PLS, PCA) (Gong et al., 2012)	.464 ± .005	.613 ± .004	.663 ± .004
SA (Fernando et al., 2013)	.450	.648	.699
DA-NBNN (Tommasi & Caputo, 2013)	.528 ± .037	.766 ± .017	.762 ± .025
DLID (S. Chopra & Gopalan, 2013)	.519	.782	.899
DeCAF$_6$ Source Only (Donahue et al., 2014)	.522 ± .017	.915 ± .015	–
DaNN (Ghifary et al., 2014)	.536 ± .002	.712 ± .000	.835 ± .000
DDC (Tzeng et al., 2014)	.594 ± .008	.925 ± .003	.917 ± .008
Proposed Approach	**.673 ± .017**	**.940 ± .008**	**.937 ± .010**

Most similar approach (matches means of distributions):
[Tzeng et al. Deep domain confusion: Maximizing for domain invariance. CoRR, abs/1412.3474, 2014]

- In the comparison we use more target examples than [Tseng et al.] (and same as DLID)
- Both our results and Tseng et al. fine-tune *AlexNet*

Further experiments: baselines

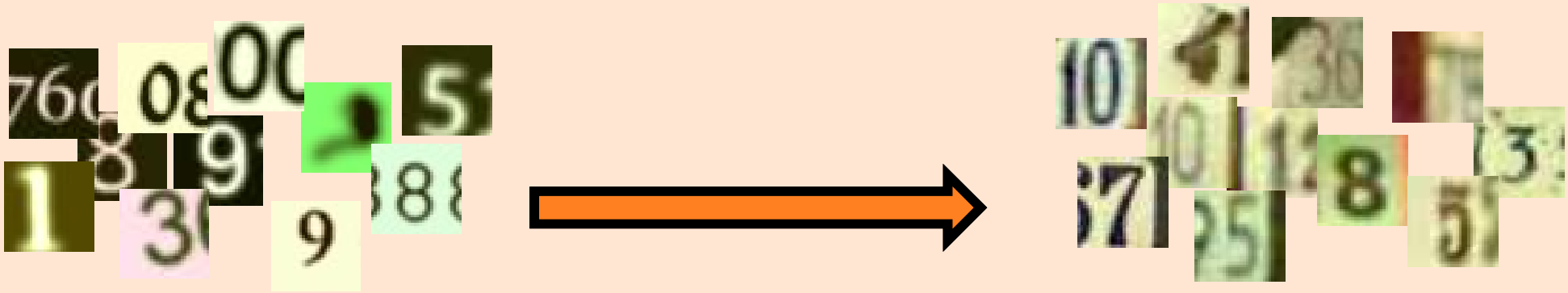


Upper bound: training on target domain **with labels**

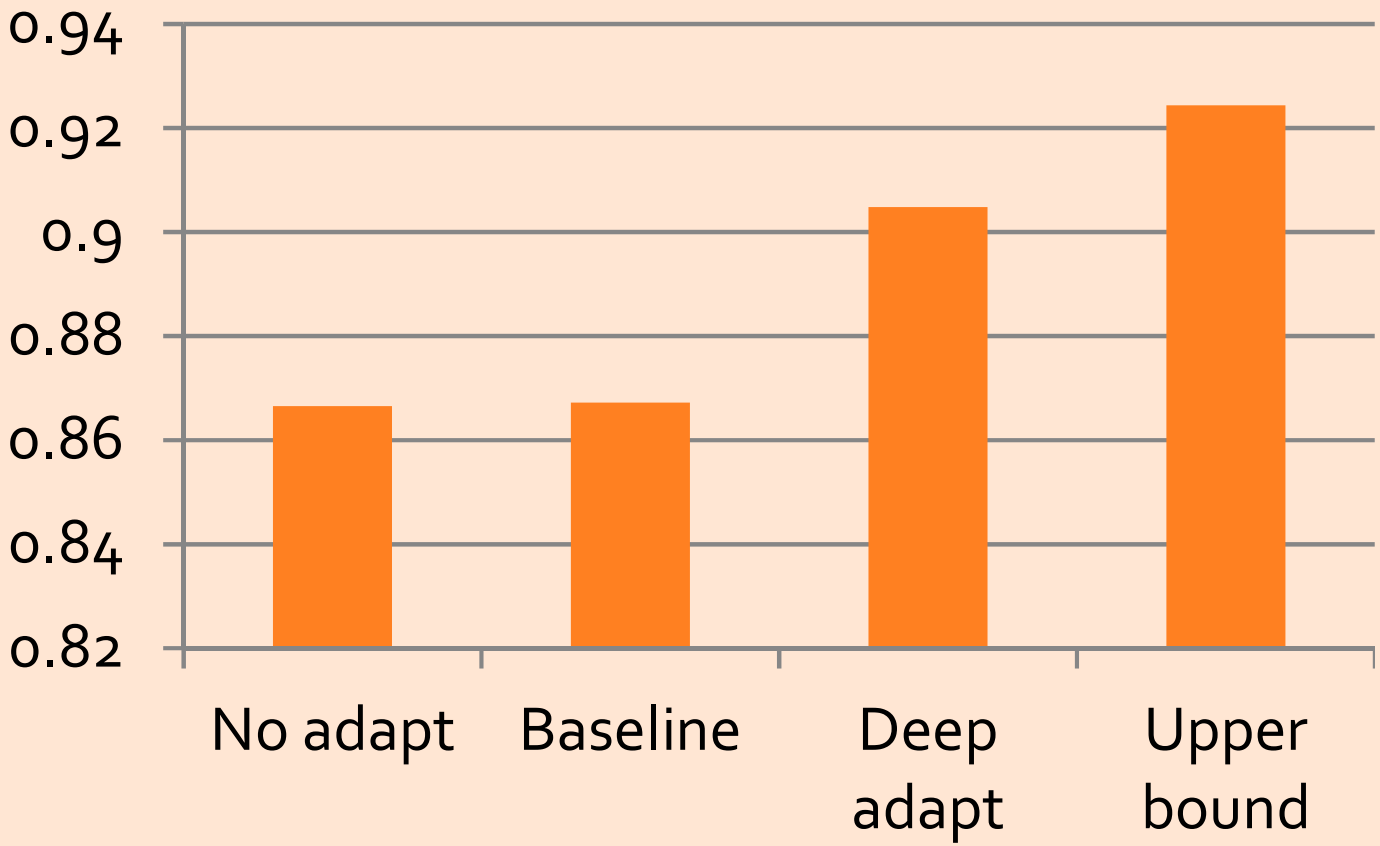
Shallow adaptation baseline: *[Fernando et al., Unsupervised visual domain adaptation using subspace alignment. ICCV, 2013]* applied to the last-but-one layer

Lower bound: training on source domain only

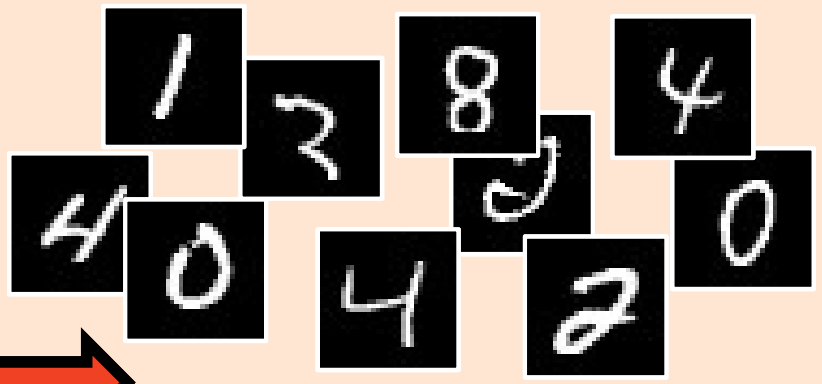
Example: from synthetic to real



"Windows digits" "House numbers"

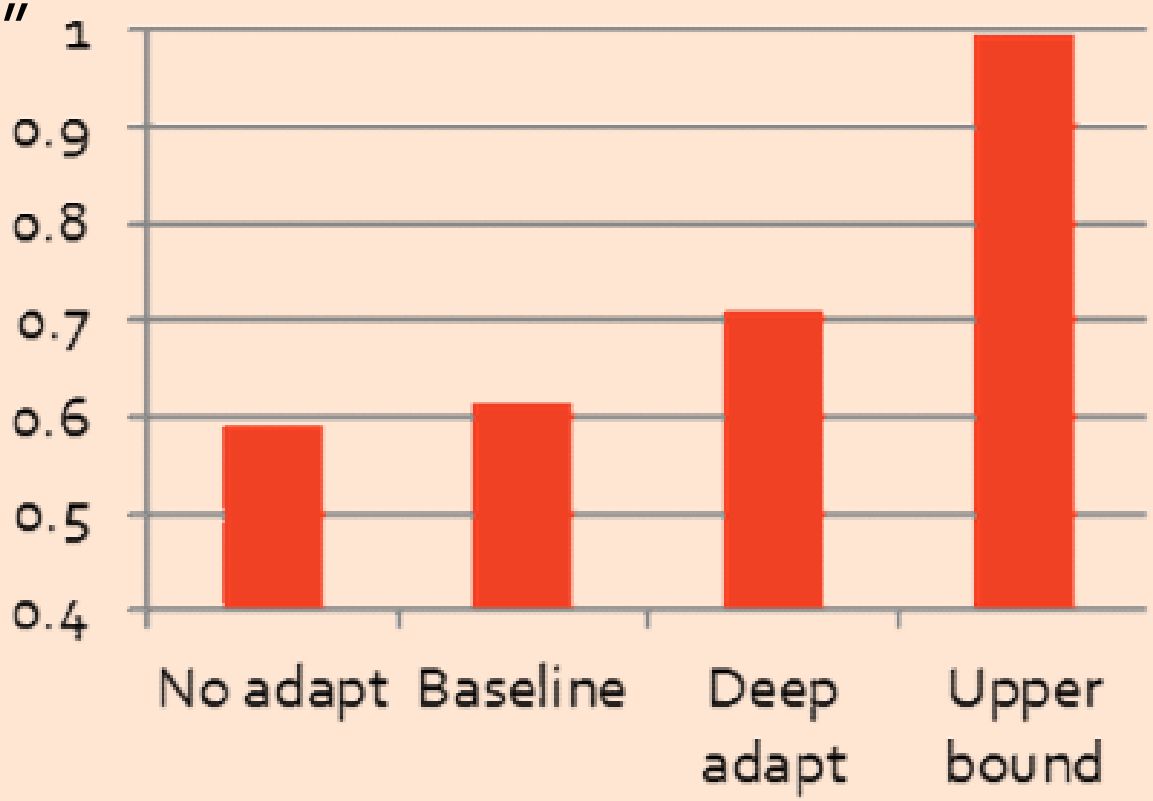


Example: large gap



MNIST

"House numbers"



Reverse direction does not work ☹

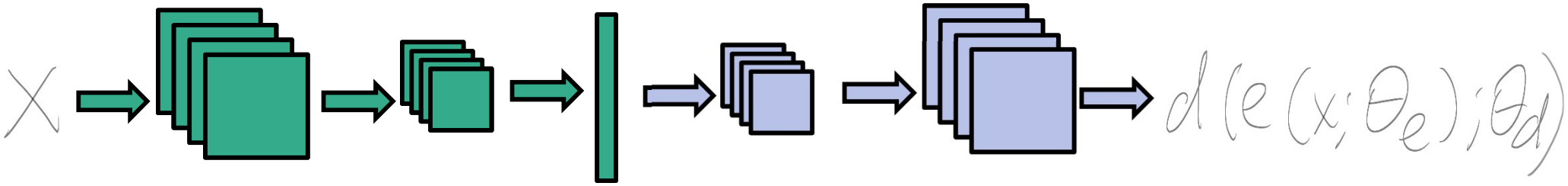
TODO: what I have not covered

1. **Автоэнкодеры, вариационные автоэнкодеры**

2. Последовательная генерация (Pixel CNNs/Pixel RNNs/DRAW,….)

3. Обращаемые отображения

Автоэнкодер



$e(x;\theta_e)$

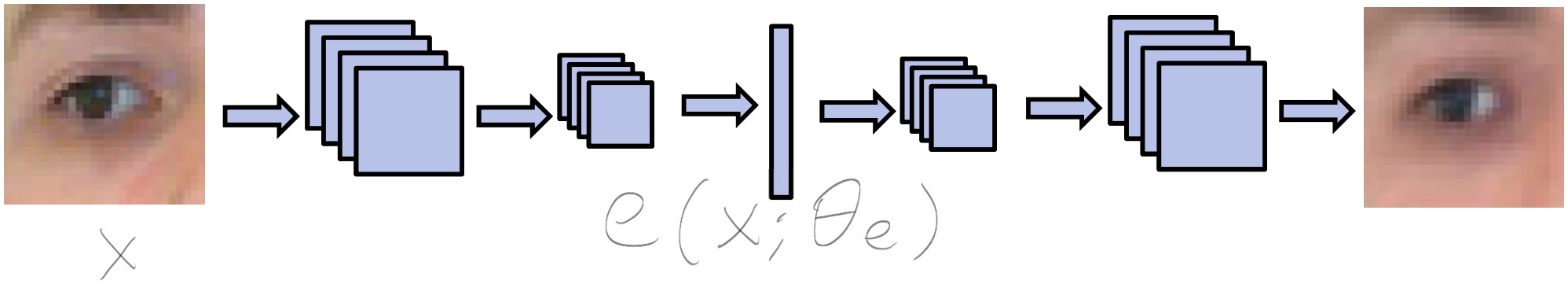
низкоразмерное представление

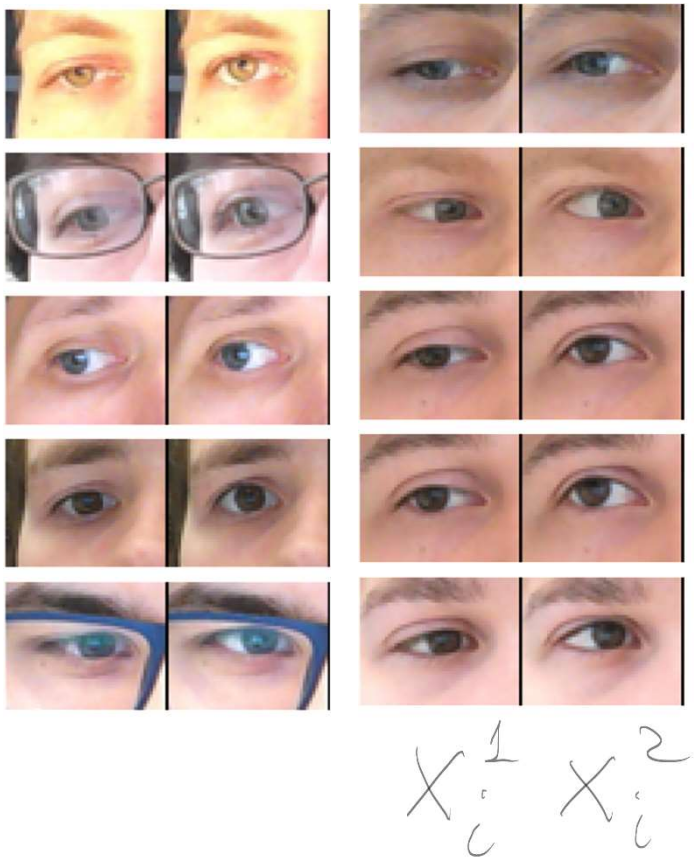
$$L(\theta_e,\theta_d) = \| d(e(x;\theta_e);\theta_d) - x \|^2$$

$$\theta_{e,d}^{t+1} - \theta_{e,d}^{t} - \lambda_t \frac{dL}{d\theta_{e,d}}(x_t)$$

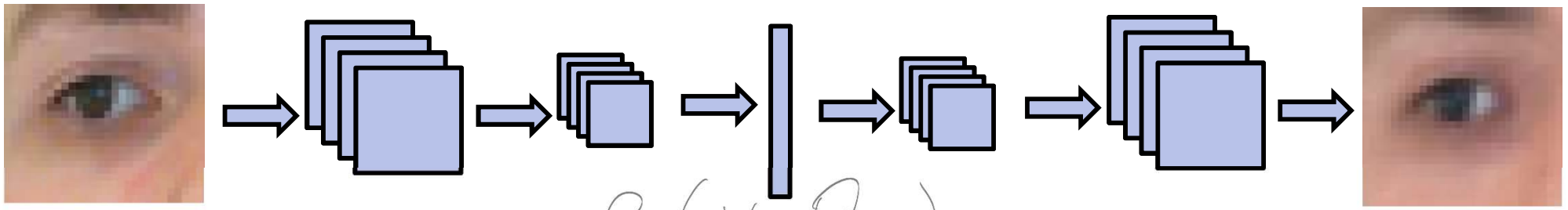
- Автоэнкодер обеспечивает сокращение размерности $x \to e(x;\theta_e)$
- Обобщает линейные методы

Редактирование с помощью автоэнкодера



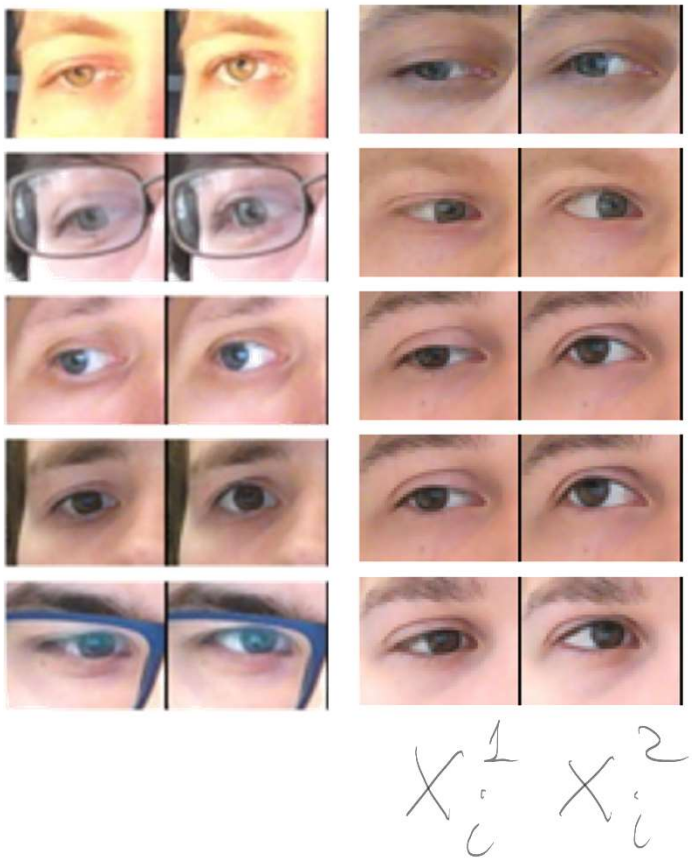


Редактирование с помощью автоэнкодера



x

$e(x;\theta_e)$



$x_i^1 \quad x_i^2$

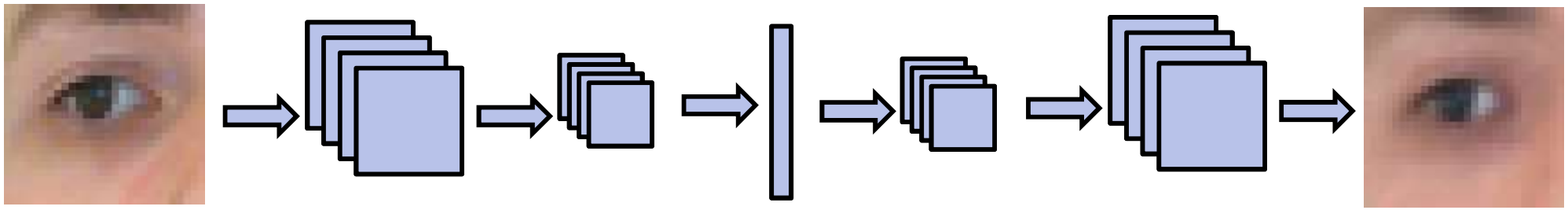
$$v_i = e(x_i^2;\theta_e) - e(x_i^1,\theta_e)$$

$$v = \frac{1}{N}\sum_{i=1}^{N} v_i$$

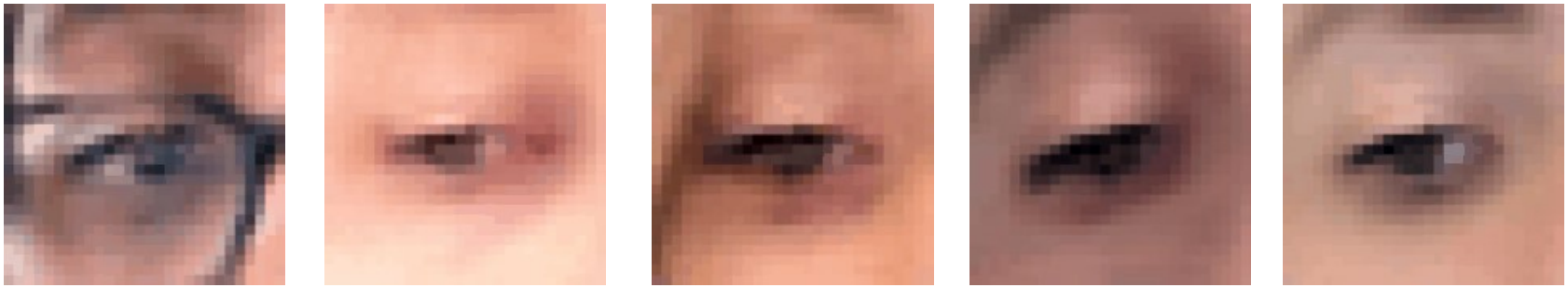
$$x^1 \rightarrow e(x^1;\theta_e) \pm \bar{v}$$

$$e(x^1) + \bar{v} \rightarrow d(e(x^1) \pm \bar{v}) \approx x^2$$

Редактирование с помощью автоэнкодера



$$d(e(x^t;\theta_e) + \alpha v, \theta_d)$$



Thanks to L.Yekimov
(+F.Chervinsky,D.Kononenko,D.Sungatullina,Y.Ganin)

План

1. «Стандартные» свёрточные сети
2. «Практическая» оптимизация
3. Синтезирующие сети с простыми функциями потери
4. Синтез через повторение статистик
5. **Играющие сети (adversarial networks)**