

Глубинное обучение

Лекция 7

Архитектуры свёрточных нейронных сетей

Михаил Гущин

mhushchyn@hse.ru

НИУ ВШЭ, 2024



НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

В предыдущей серии

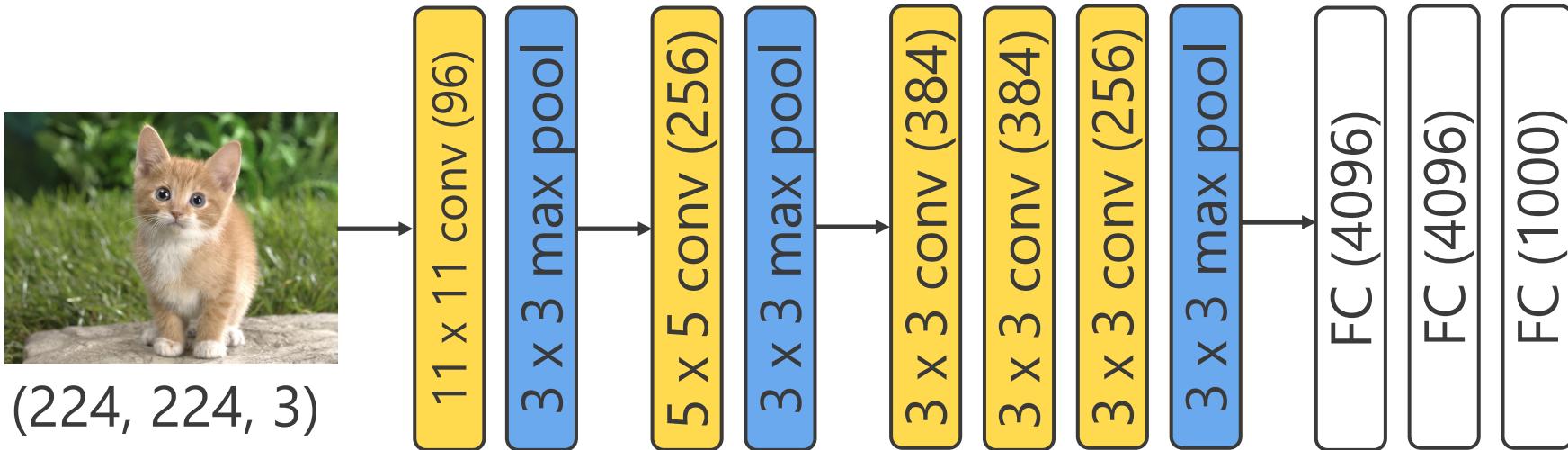
ImageNet



Подробнее: <https://image-net.org/challenges/LSVRC>

- ▶ ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- ▶ 1000 классов изображений
- ▶ Более 1 000 000 изображений

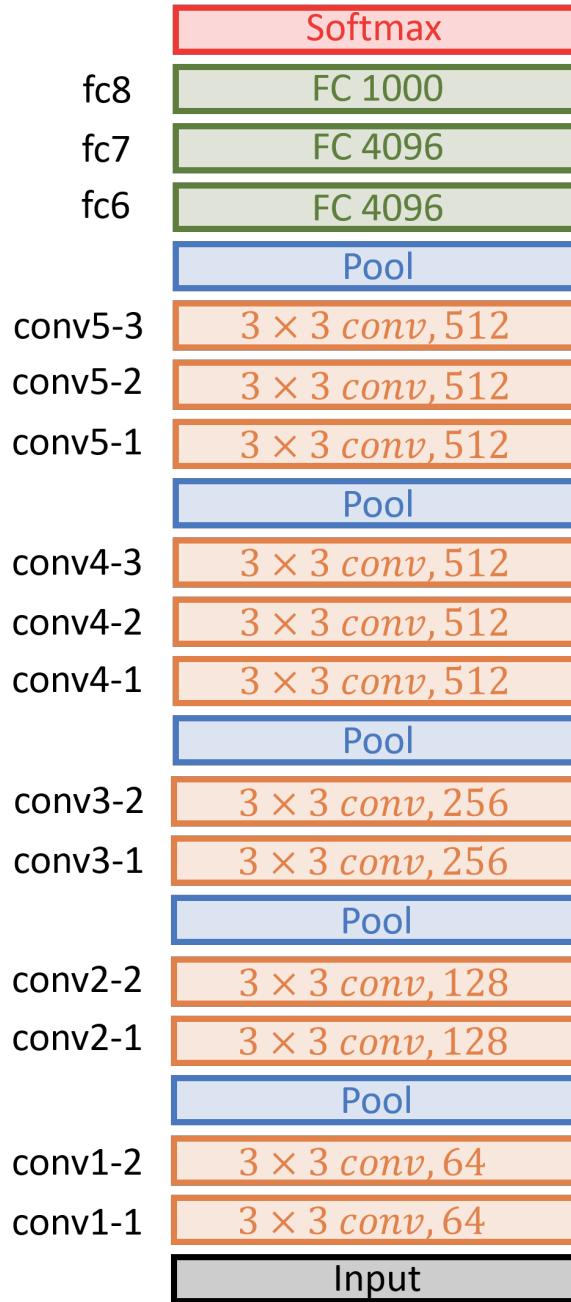
AlexNet



- ▶ Больше слоев и нейронов
- ▶ ReLU функция активации во всех слоях
- ▶ Softmax функция активации выходного слоя

Подробнее: A. Krizhevsky et al. ImageNet classification with deep convolutional neural networks,
NIPS 2012 (URL: <https://doi.org/10.1145/3065386>)

VGG-16



Рекомендации по архитектуре

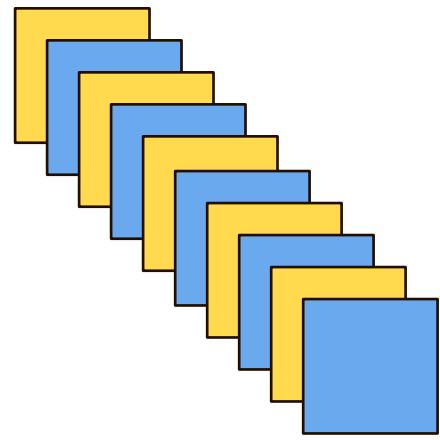
- ▶ Берем больше свёрточных слоев
- ▶ Лучше использовать маленькие свертки
 - $1 \times 1, 3 \times 3, 5 \times 5$
- ▶ Увеличиваем число каналов с каждым свёрточным слоем
- ▶ Используем ReLU функцию активации после каждой свертки (но до pooling)

GoogLeNet

GoogLeNet

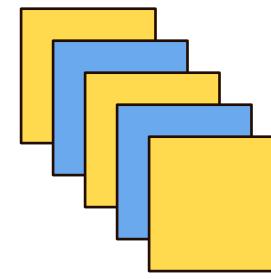
- ▶ Победитель ImageNet 2014 года
- ▶ Использует 1×1 свертки
- ▶ Состоит из **inception** модулей
- ▶ Вычислительно эффективна

Свертка 1×1



Входное
изображение
 $(28 \times 28 \times 10)$

1×1 свертка
с **5** фильтрами

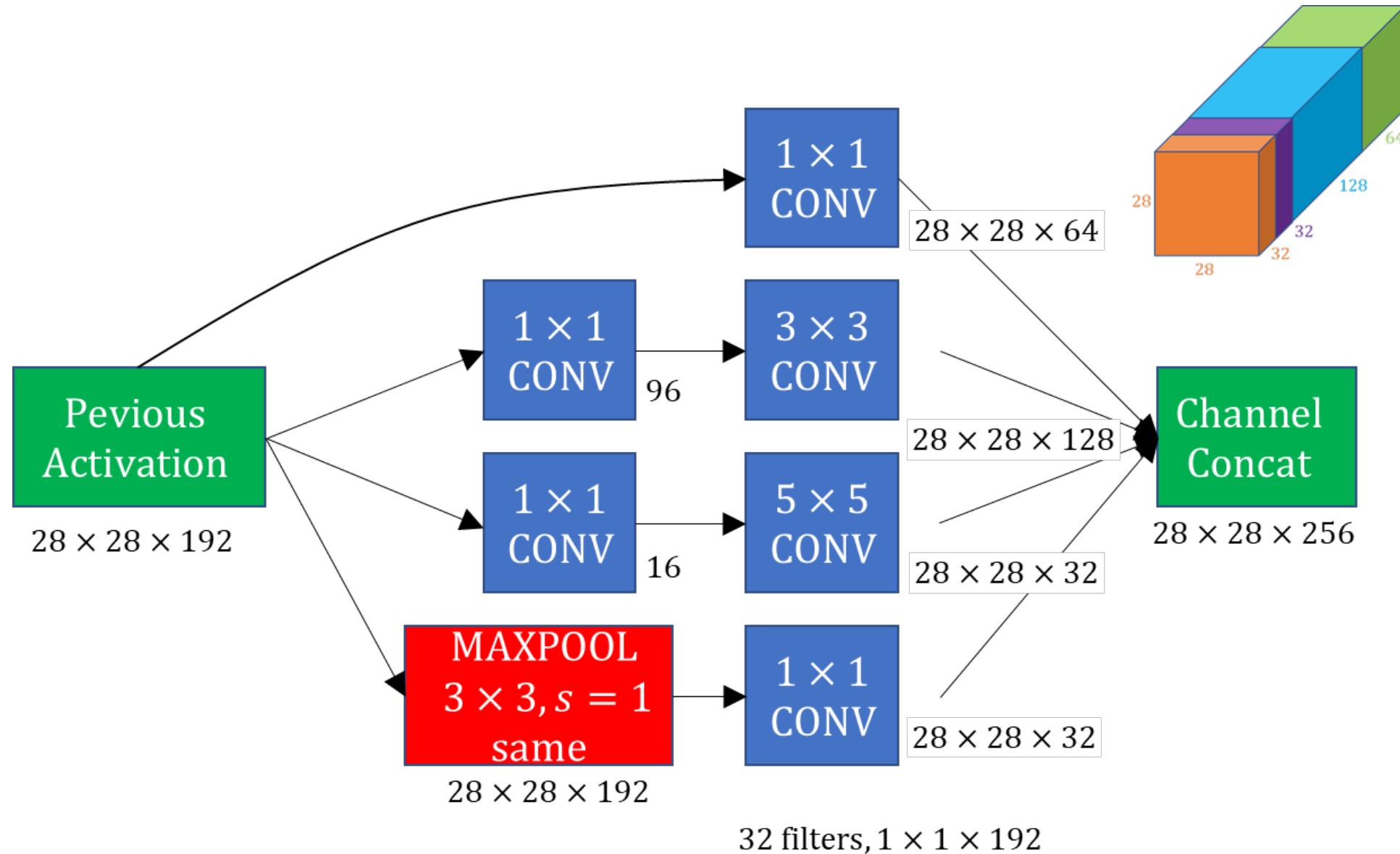


Выходное
изображение
 $(28 \times 28 \times 5)$

Свойства 1×1 свертки

- ▶ Свертка только по каналам изображения
- ▶ Уменьшает число каналов изображения
- ▶ Сохраняет пространственные размерности изображения

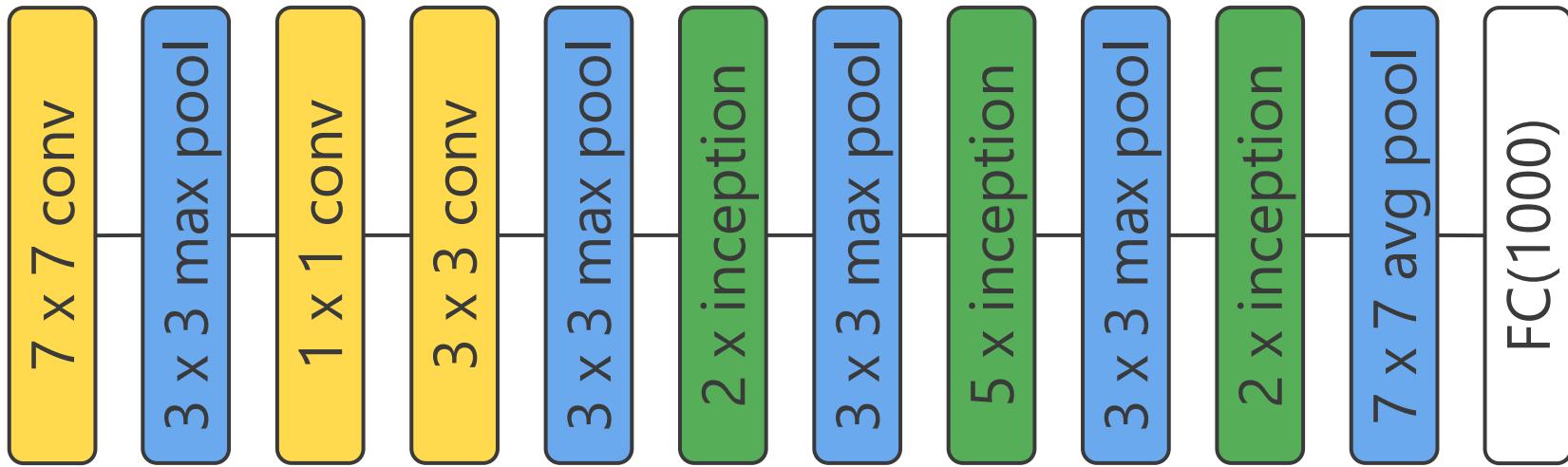
Inception модуль



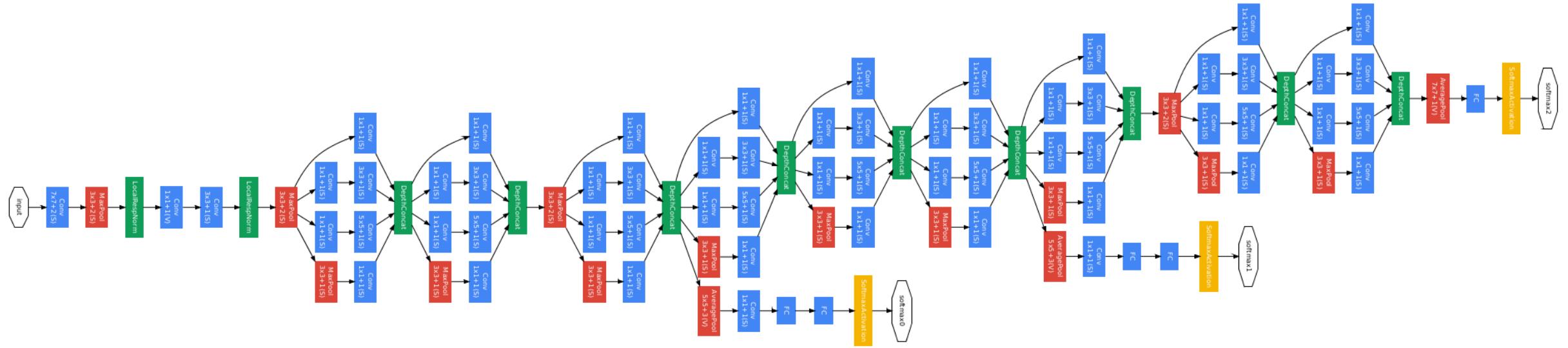
Свойства inception модуля

- ▶ Используются разные свертки параллельно
- ▶ Свертки сохраняют размер изображения:
 - 1×1 , padding 0
 - 3×3 , padding 1
 - 5×5 , padding 2
- ▶ 1×1 свертки уменьшают количество каналов
- ▶ ReLU функция активации на всех слоях

GoogLeNet



GoogLeNet

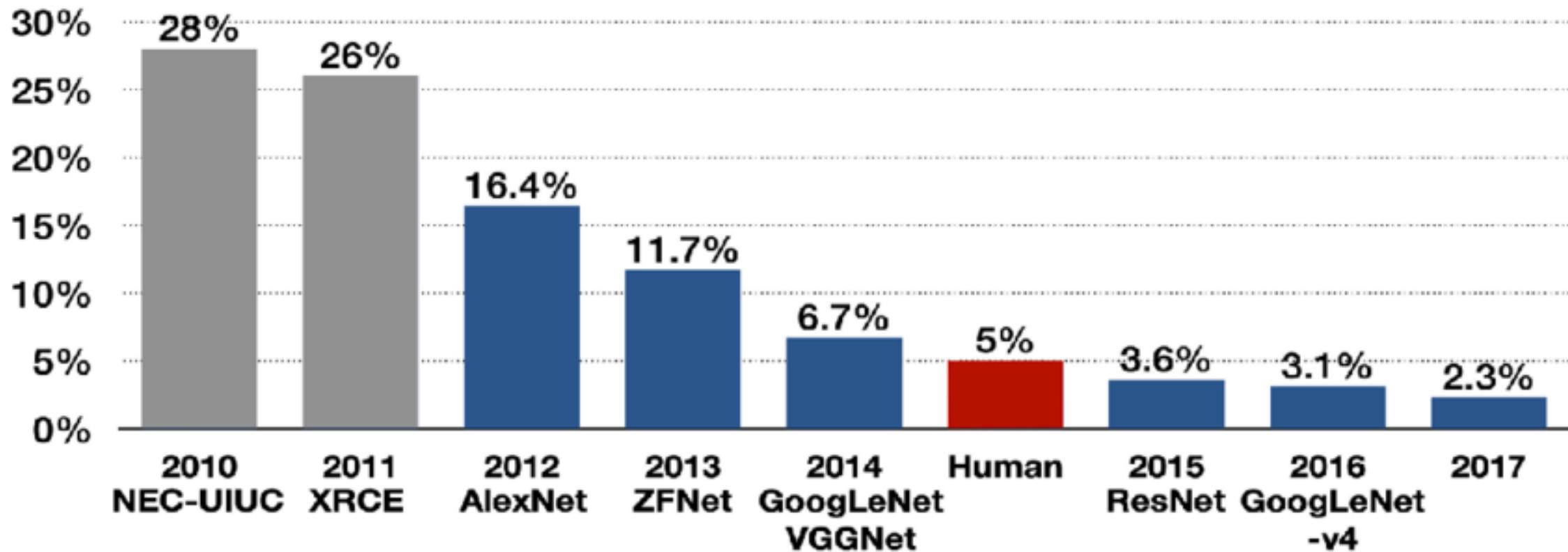


GoogLeNet

- ▶ 22 слоя
- ▶ Нет полносвязных слоев, кроме выходного
- ▶ 1×1 свертки уменьшают число параметров обучения
- ▶ В 12 раз меньше параметров, чем в AlexNet
- ▶ Качество лучше, чем у AlexNet

Результаты ImageNet

Top-5 error



Выводы

- ▶ Развитие технологий позволило обучать глубокие сети
- ▶ Больше слоев — лучше
- ▶ Блоки сверточных слоев
- ▶ Лучше использовать маленькие свертки
- ▶ 1×1 свертка уменьшает размерность

ResNet

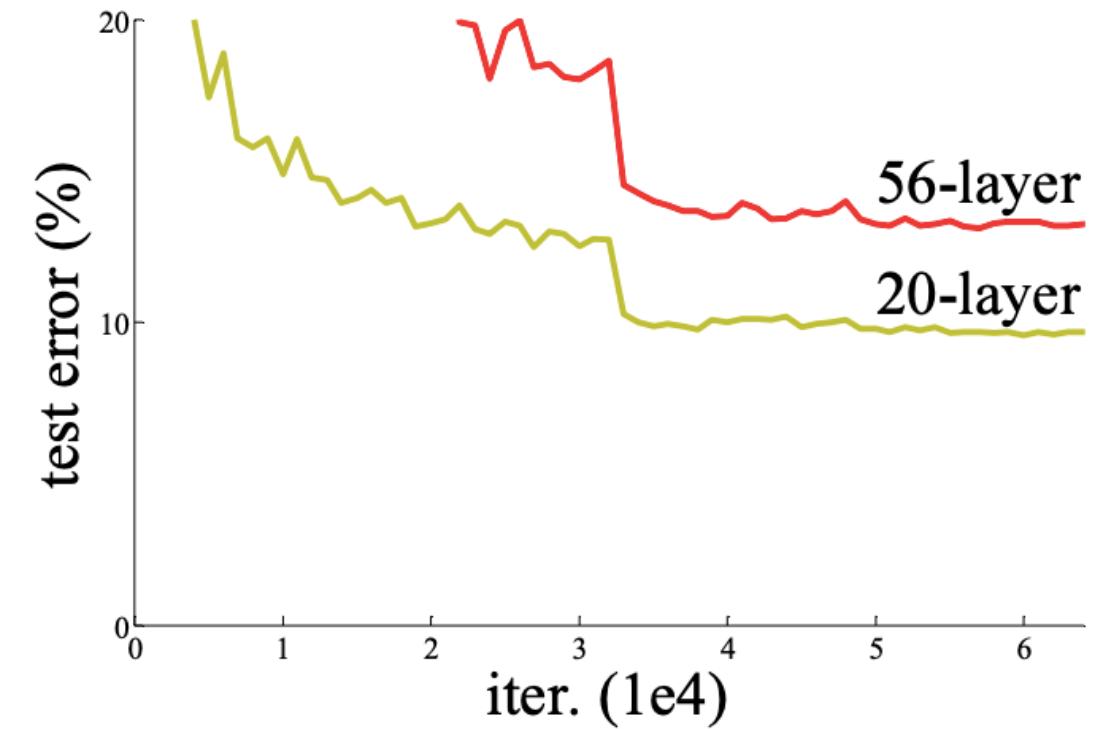
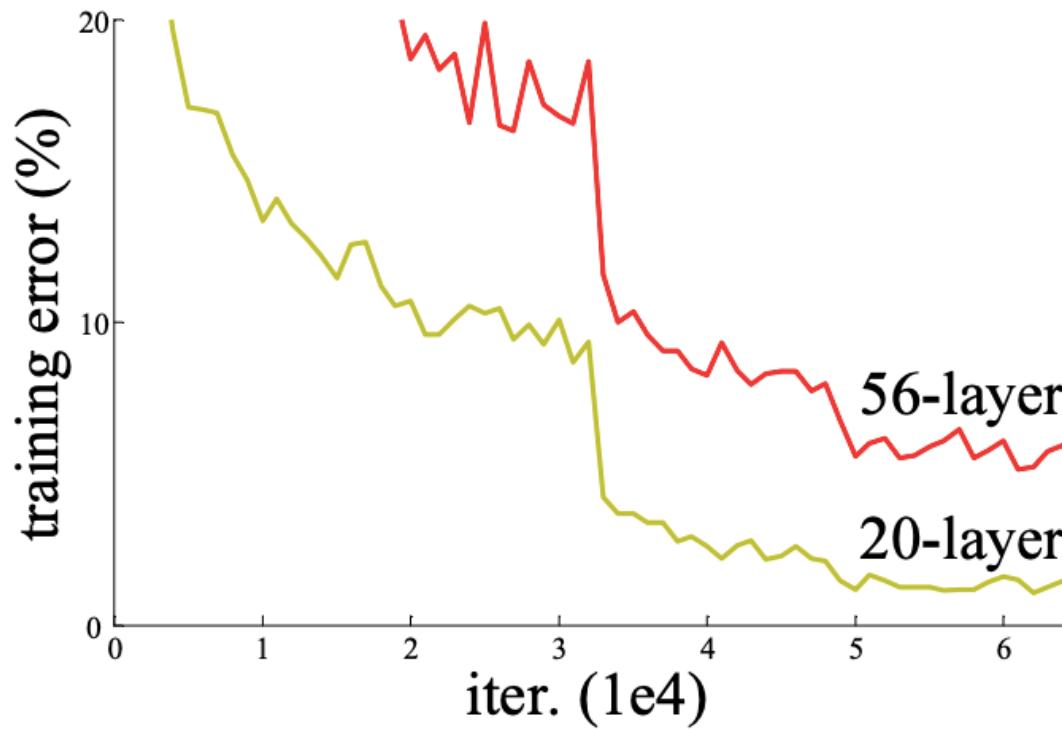


Глубокие сети

- ▶ Как улучшить качество сети?
- ▶ ImageNet показал, что модели с **большим** числом слоев показывали **лучшее качество**
- ▶ Что произойдет если взять больше слоев?

Глубокие сети

Ошибки глубоких нейронных сетей с 20 и 56 слоями на данных CIFAR-10



Подробнее: Kaiming He et al. Deep Residual Learning for Image Recognition, CVPR 2016 (URL: <https://arxiv.org/abs/1512.03385>)

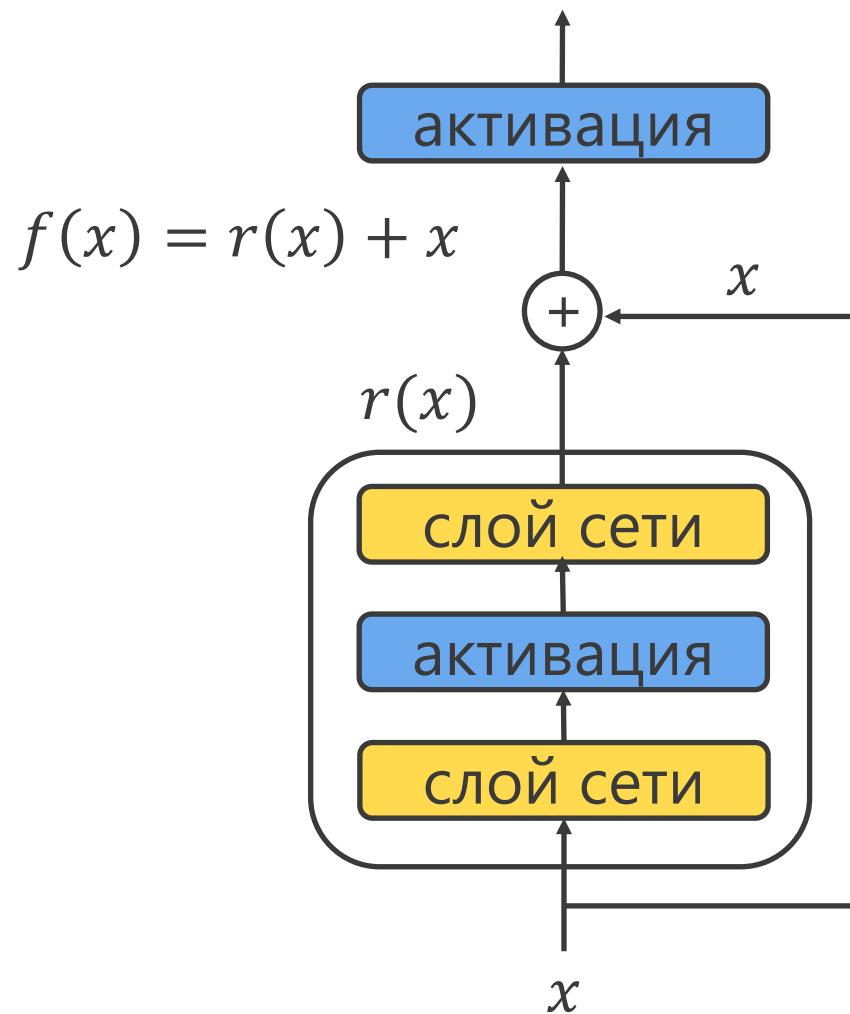
Глубокие сети

- ▶ Очень глубокие сети показывают хуже результат
- ▶ Это не связано с переобучением
- ▶ Возможно, проблема в процессе оптимизации сети
- ▶ Более глубокие сети труднее обучать

Глубокие сети

- ▶ Более глубокая сеть должна показывать качество не хуже, чем менее глубокая сеть
- ▶ Как можно этого достичь?

Остаточный (residual) блок



Остаточный (residual) блок

Посчитаем градиент для $f(x)$ по параметрам сети w до блока:

$$\frac{\partial f(x)}{\partial w} = \frac{\partial r(x)}{\partial x} \frac{\partial x}{\partial w} + \frac{\partial x}{\partial w}$$

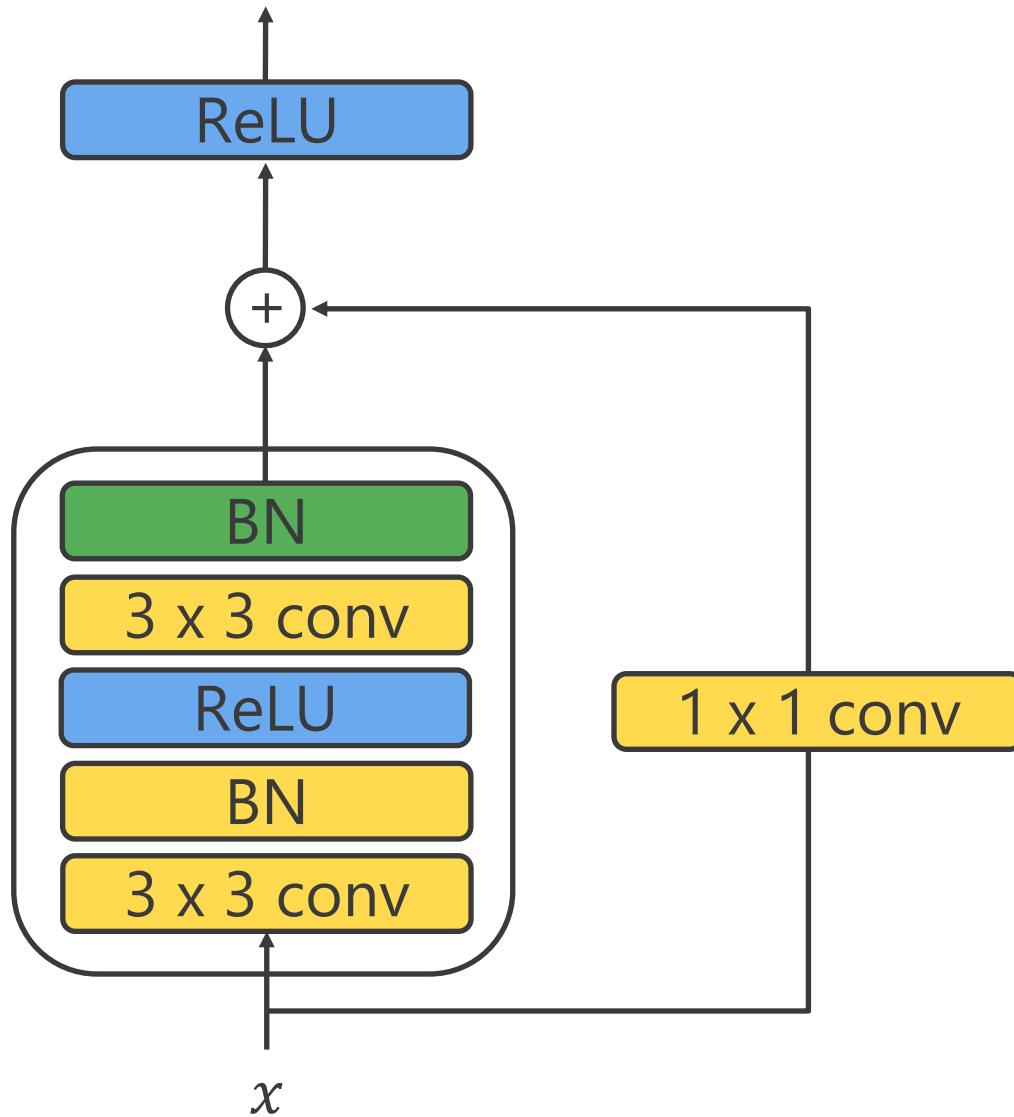
$$\frac{\partial f(x)}{\partial w} = \left(\frac{\partial r(x)}{\partial x} + 1 \right) \frac{\partial x}{\partial w}$$

Более эффективное обратное распространение ошибки в случае малых значений $\frac{\partial r(x)}{\partial x}$

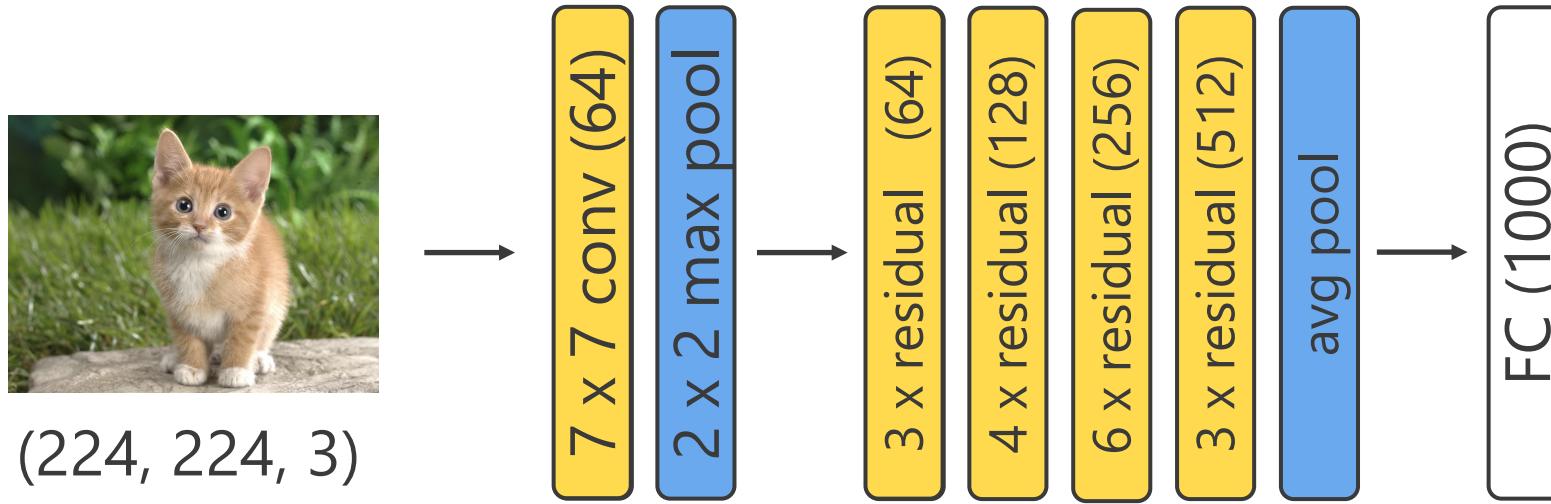
Остаточный (residual) блок

- ▶ Каждый блок учит добавку $r(x)$ к предыдущему блоку
- ▶ Обратное распространение ошибки более эффективно
- ▶ Глубокая сеть обучается лучше

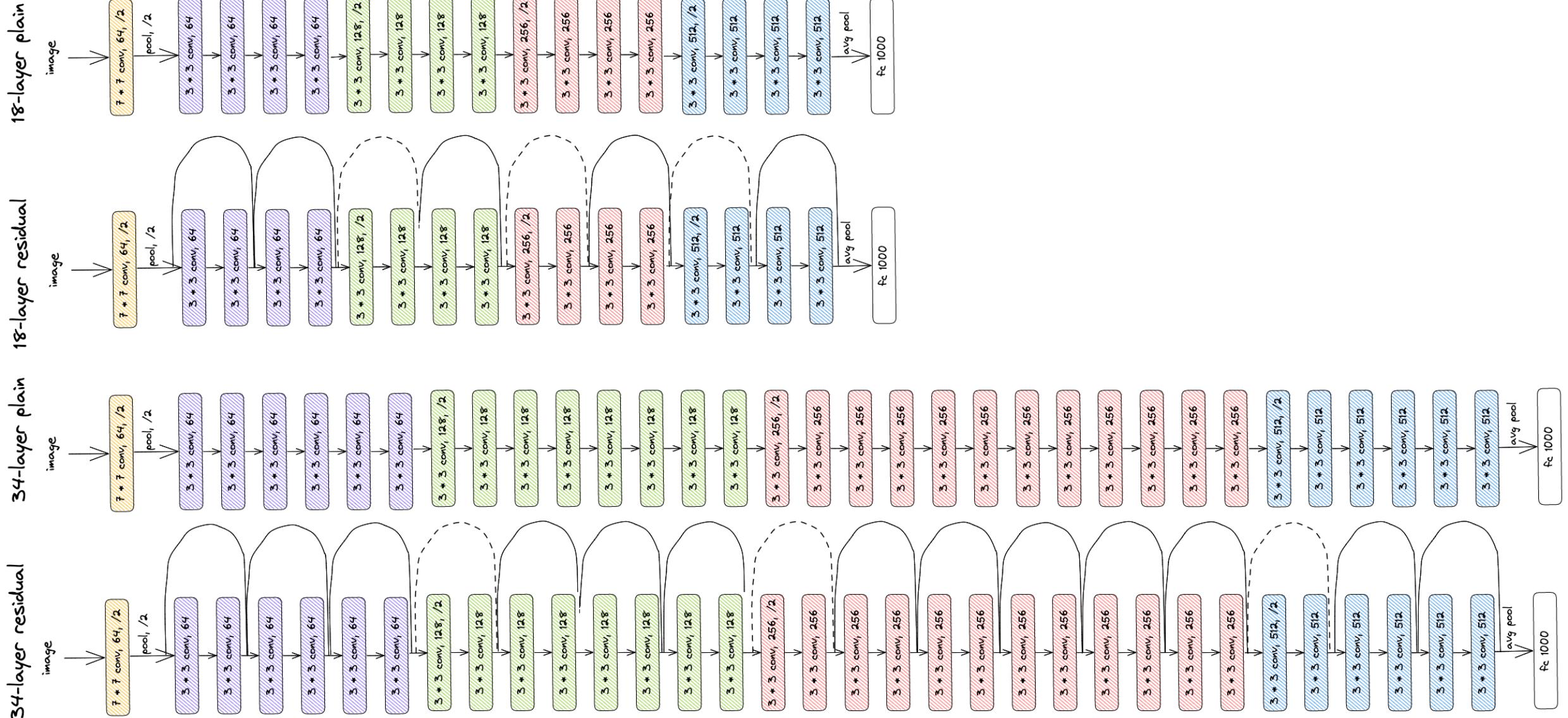
Остаточный (residual) блок



Residual Network (ResNet-34)



ResNet 18 и 34

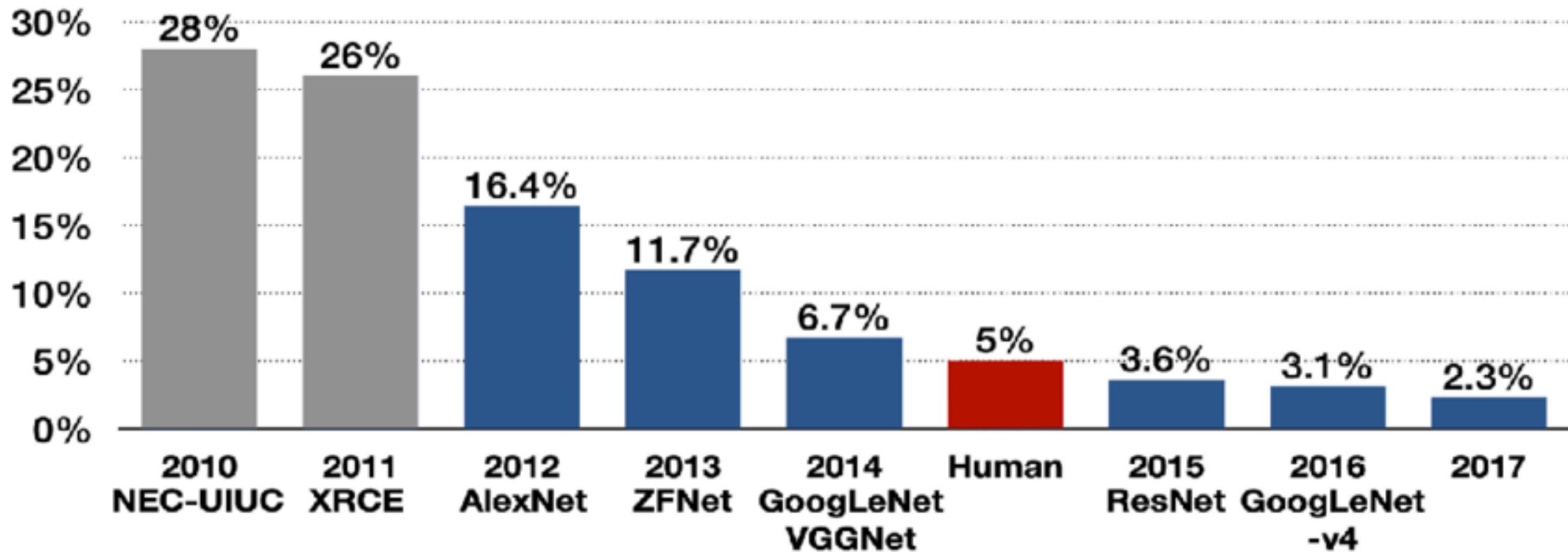


ResNet 18 и 34

- ▶ Позволяет обучать очень глубокие сети
- ▶ Каждый блок учит поправку в предыдущему
- ▶ Обратное распространение ошибки более эффективно

Результаты ImageNet

Top-5 error



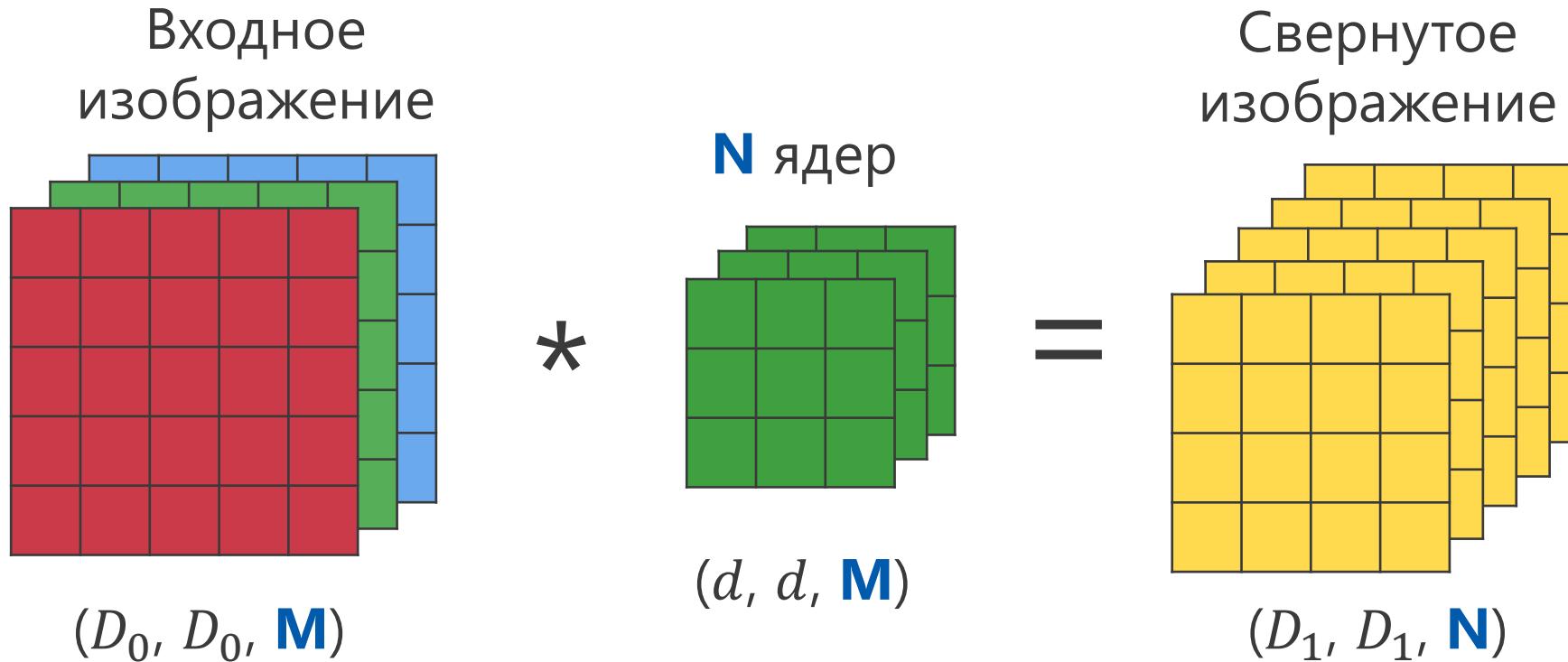
MobileNet



MobileNet

- ▶ ResNet позволил обучать очень глубокие сети.
- ▶ Но такие сети содержат много параметров. Их долго обучать, они медленные в применении.
- ▶ Как ускорить сверточные сети?

Свертка изображения



Число операций для **обычной** свертки:

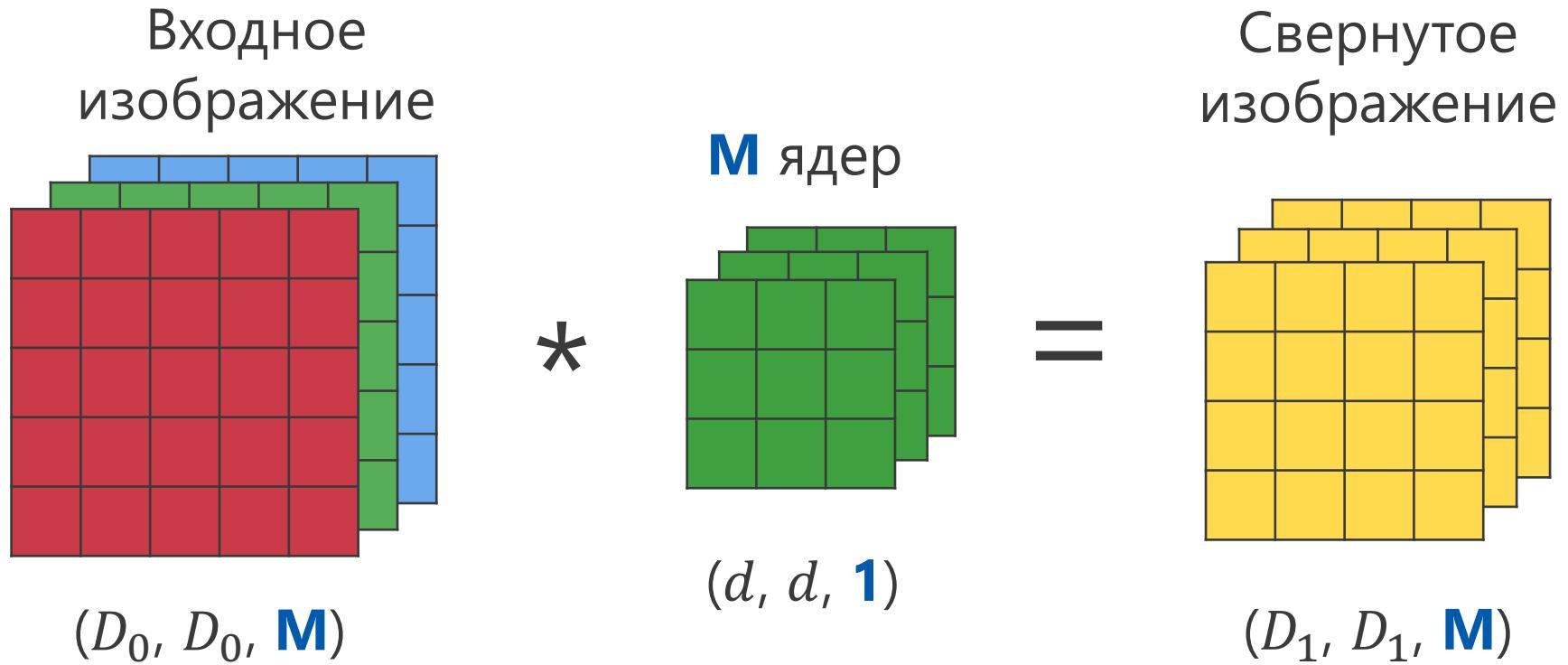
$$ND_1^2(d^2M)$$

Разложение операции свертки

Разложим обычную свертку на

- ▶ Свертку в глубину (depthwise)
- ▶ Точечную свертку (pointwise)

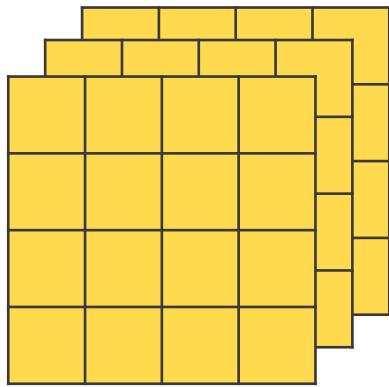
Свертка в глубину (depthwise)



- Отдельное ядро на каждый канал входного изображения
- Число каналов после свертки сохраняется
- Число операций $MD_1^2(d^2)$

Точечная свертка (pointwise)

Входное
изображение



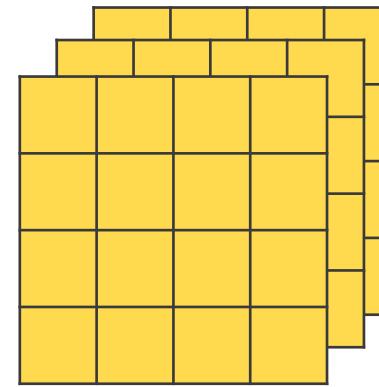
N ядер

*

(1, 1, **M**)

=

Свернутое
изображение



(D_1, D_1, \mathbf{M})

(D_1, D_1, \mathbf{N})

- 1 x 1 свертка с M каналами
- Размерность изображения сохраняется
- Число операций $ND_1^2(M)$

Разложение операции свертки

Отношение числа операций:

$$\frac{N_{\text{в глубину}} + N_{\text{точечная}}}{N_{\text{свертка}}} = \frac{MD_1^2 d^2 + ND_1^2 M}{ND_1^2 d^2 M}$$

$$= \frac{MD_1^2(d^2 + N)}{MD_1^2(d^2 N)} =$$

$$= \frac{1}{N} + \frac{1}{d^2}$$

Разложение операции свертки

Для $N = 100$ выходных каналов и размера ядра свертки $d = 5$ получим:

$$\frac{N_{\text{в глубину}} + N_{\text{точечная}}}{N_{\text{свертка}}} = \frac{1}{N} + \frac{1}{d^2} = 0.05$$

В 20 раз меньше вычислительных операций!

MobileNet

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

MobileNet

Модель	ImageNet, точность, %	Операции, млн.	Параметры, млн.
MobileNet-224	70.6	569	4.2
GoogleNet	69.8	1550	6.8
VGG 16	71.5	15300	138

- ▶ Меньше количество вычислительных операций
- ▶ Меньше параметров сети
- ▶ Качество лучше, либо с небольшой потерей

EfficientNet

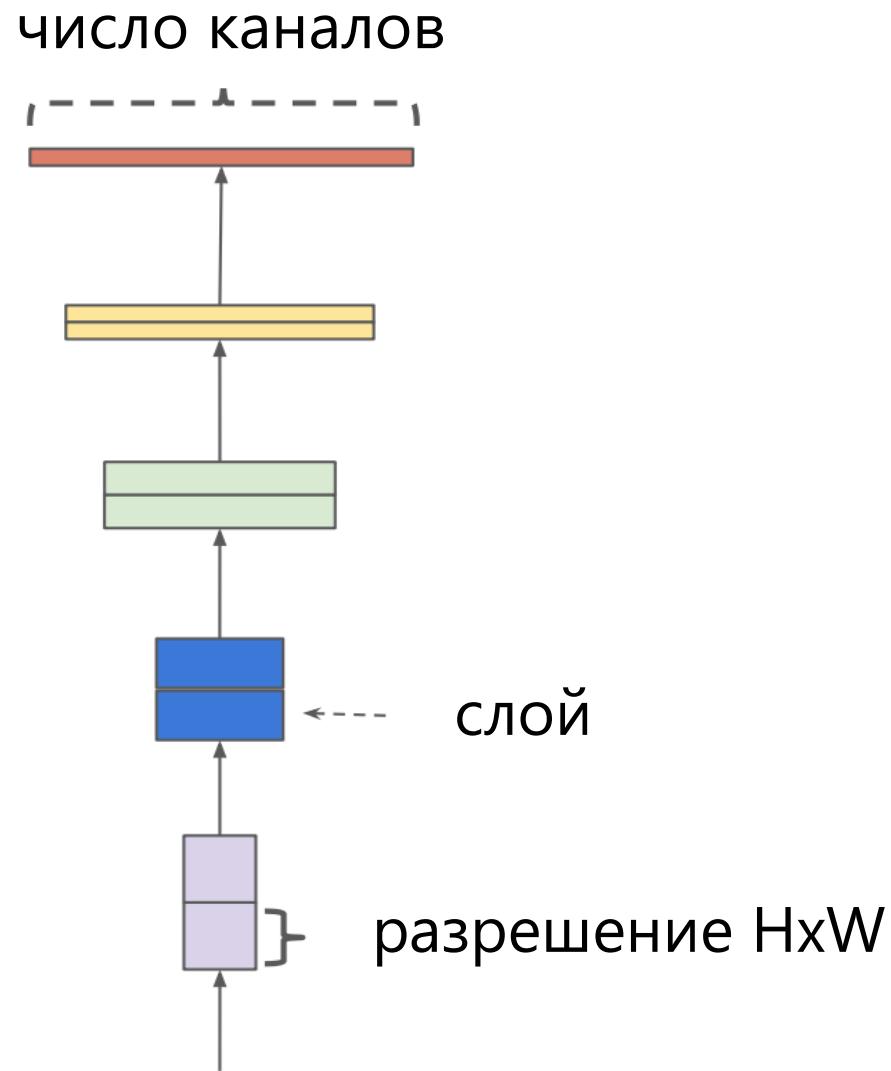
EfficientNet

- ▶ Как подобрать число слоев сети?
- ▶ Как выбрать количество каналов в каждом слое?
- ▶ Изображения с каким разрешением более оптимально?

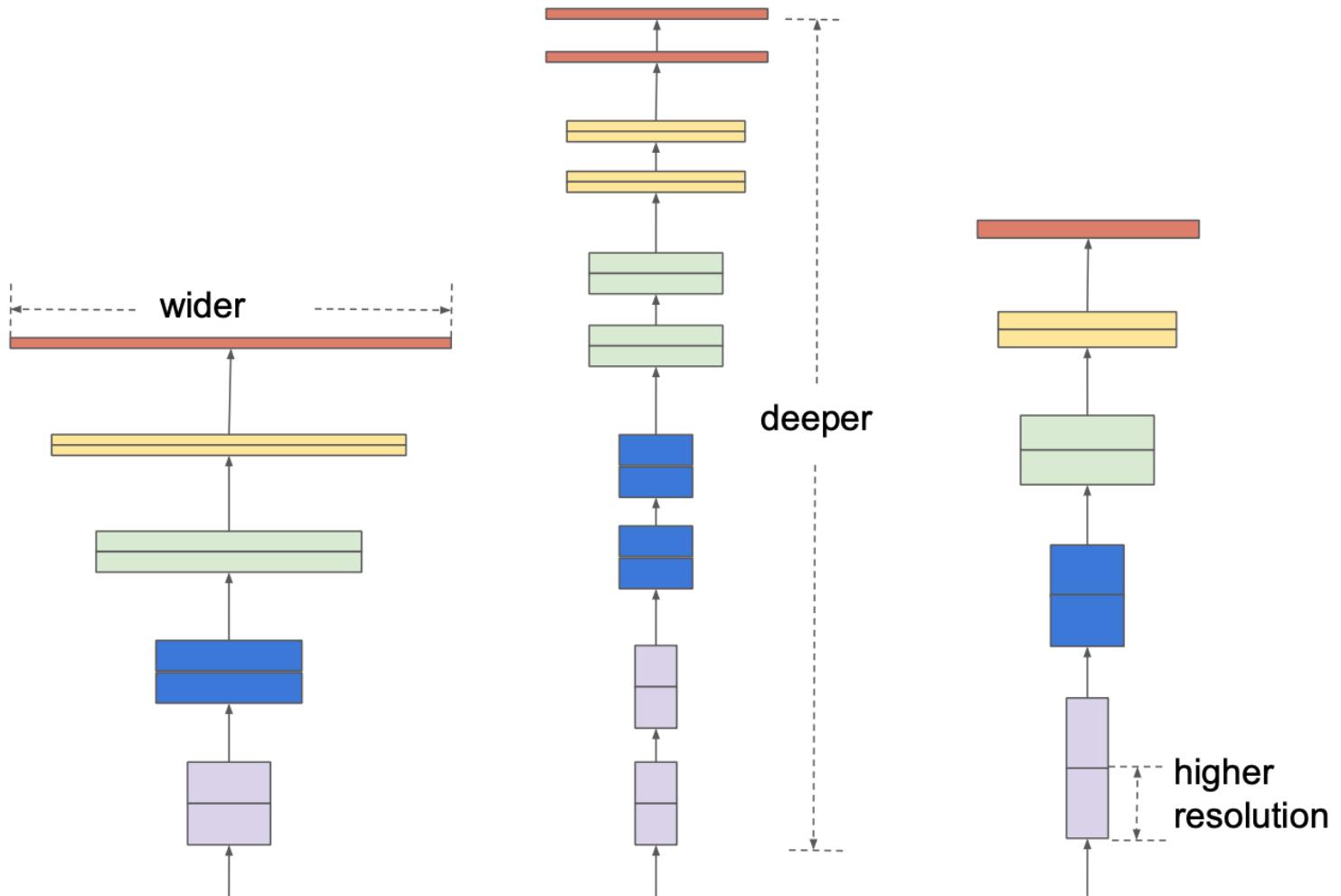
EfficientNet

- ▶ Оптимизирует архитектуру сверточной сети
- ▶ Максимизирует качество классификации
- ▶ Учитывает ограничения на объем вычислений

Базовая архитектура сети



Оптимизация архитектуры



EfficientNet

- ▶ Берем базовую сверточную сеть
- ▶ Масштабируем число каналов / глубину сети / размер изображения умножением на $\alpha^\phi, \beta^\phi, \gamma^\phi$
 - ϕ — настраиваемый параметр
 - α, β, γ — константы, которые нужно найти

EfficientNet

- ▶ α, β, γ находим поиском по сетке значений, оптимизируя целевую функцию
- ▶ Целевая функция:

$$Accuracy * \left(\frac{FLOPS}{T} \right)^{-0.07}$$

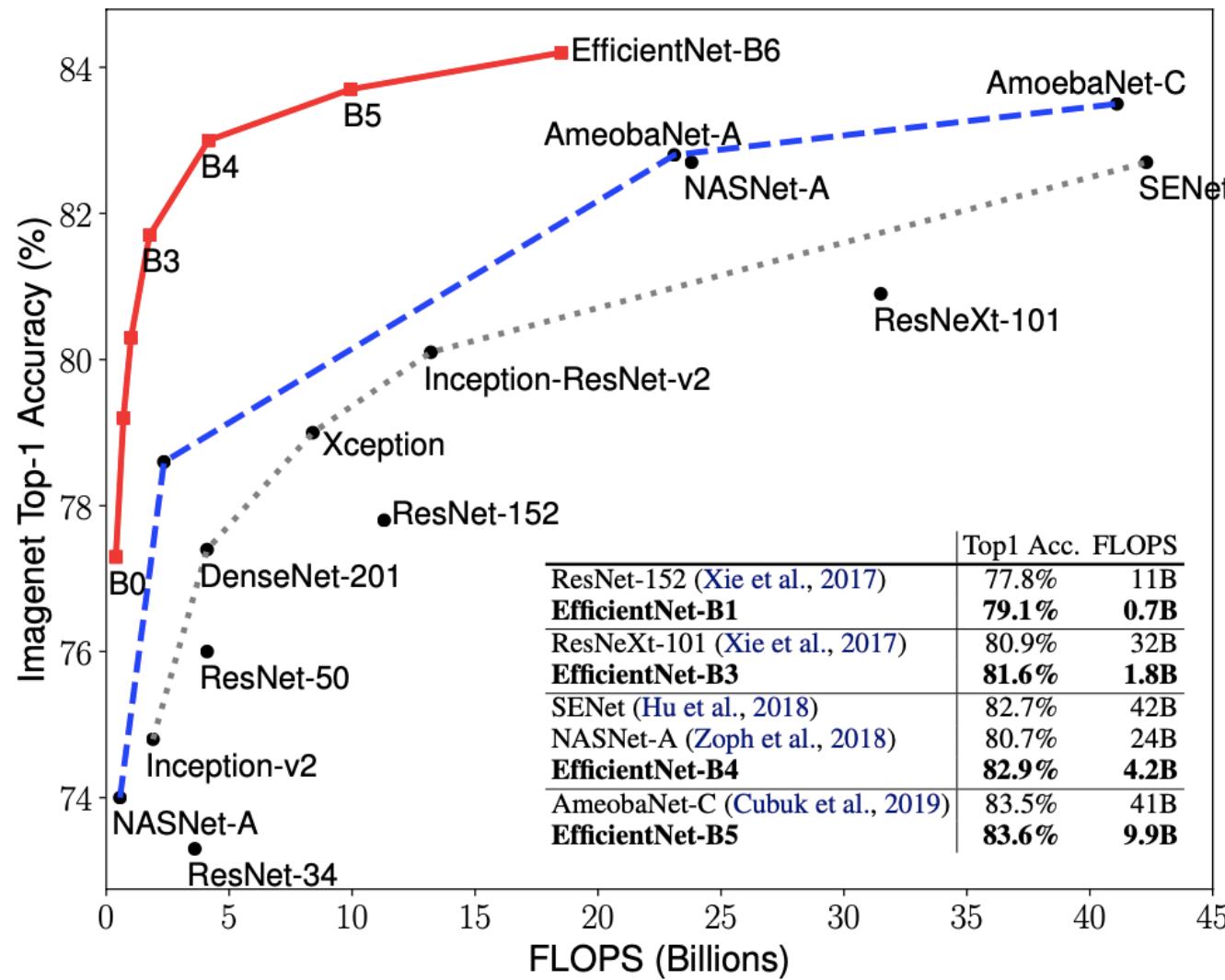
- ▶ T — наше целевое значение числа операций (FLOPS)
- ▶ Ограничение: $\alpha\beta^2\gamma^2 \approx 2$

EfficientNet

Алгоритм:

- ▶ Берем $\phi = 1$
- ▶ Находим α, β, γ поиском по сетке значений (B0)
- ▶ Для разных ϕ , масштабируем число каналов / глубину сети / размер изображения умножением на $\alpha^\phi, \beta^\phi, \gamma^\phi$ (B1-6)

EfficientNet



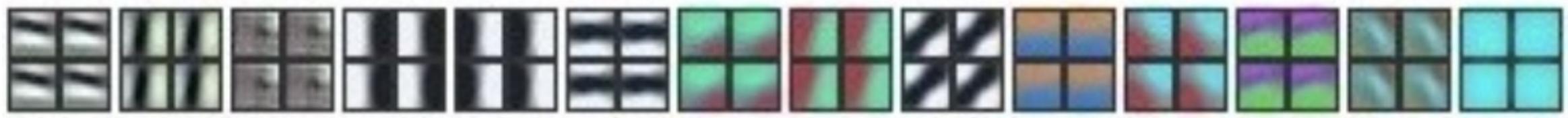
Перенос стиля

Интерпретация моделей

- ▶ Каждое ядро свертки находит определенный шаблон
- ▶ На первых слоях находятся простые шаблоны
- ▶ На глубоких слоях — более сложные
- ▶ Шаблон каждого нейрона можно найти максимизацией его активации по входной картинке методом градиентного спуска

Максимизация активации нейрона

1 слой

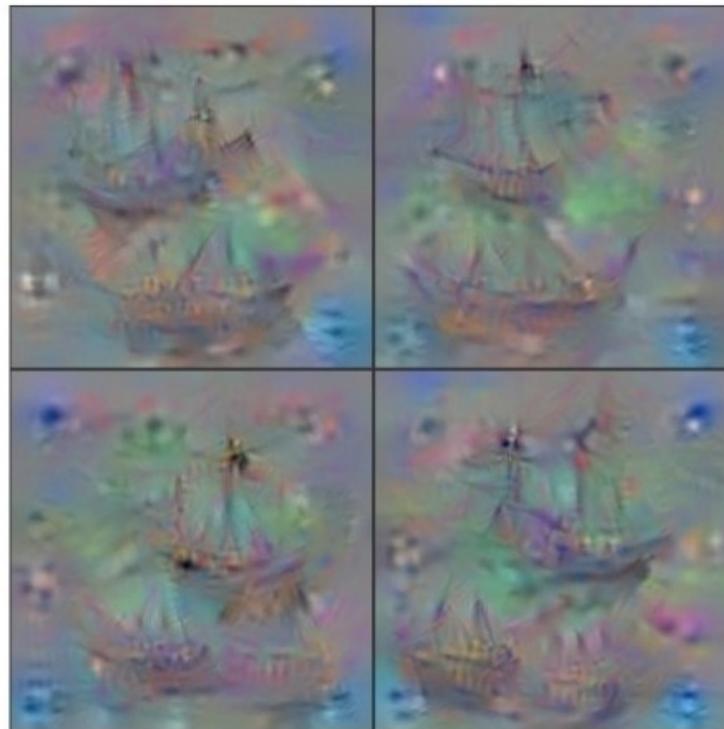


2 слой



Максимизация активации нейрона

8 слой



Перенос стиля изображения

Исходное
изображение



Стилевое
изображение



+

Итоговое
изображение



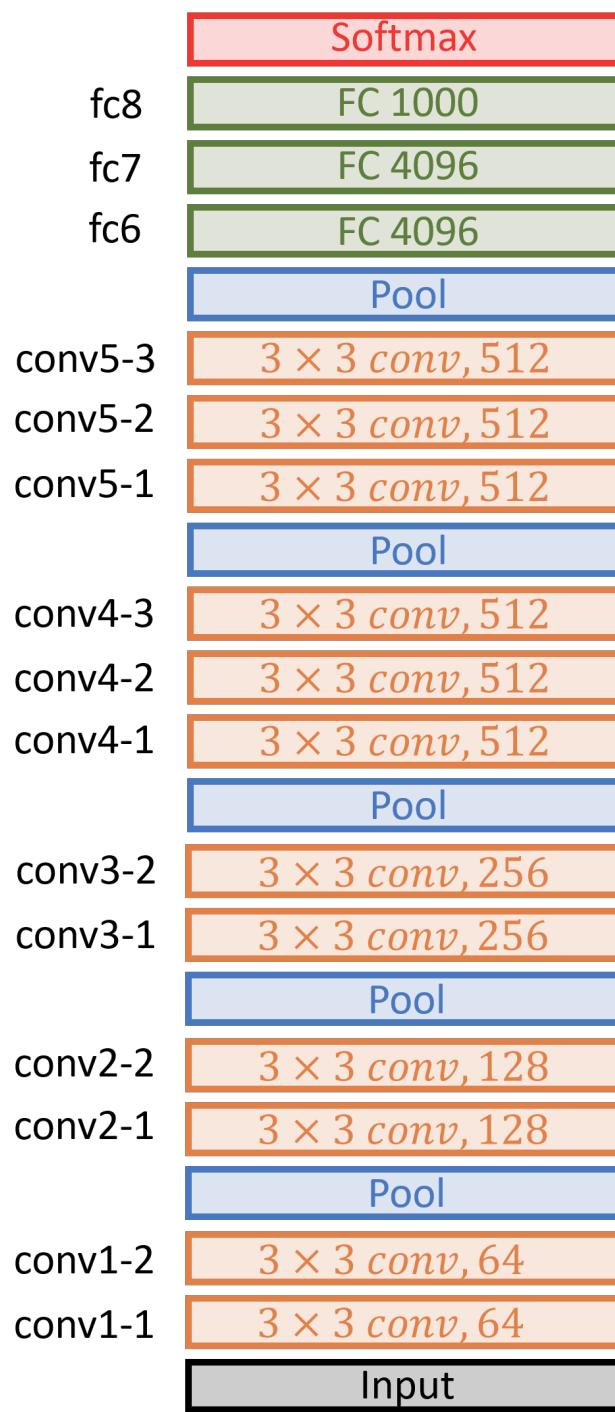
Перенос стиля изображения

Итоговое изображение:

- ▶ Похоже на исходное изображение по содержанию
- ▶ Похоже на стилевое изображение по стилю

Как измерить схожесть изображений?

VGG-16



VGG-16

- ▶ Возьмем обученную на ImageNet сеть VGG-16
- ▶ Первые слои сети детектируют простые шаблоны
- ▶ Глубокие слои находят более сложные шаблоны
- ▶ Можем сравнивать изображения по значениям разных фильтров в разных слоях сети

Сравнение изображений

Пусть даны два изображения: А и В

- ▶ Пропустим эти изображения через VGG-16 сеть
- ▶ Запомним значения всех нейронов сети
- ▶ Сравним эти значения для изображений А и В

Сравнение по содержанию

Разница в содержании изображений А и В:

$$L_{content}^l(A, B) = \sum_{i,j} (A_{ij}^l - B_{ij}^l)^2$$

A_{ij}^l — значение i -го фильтра на j -ой позиции l -го слоя сети

Сравнение по стилю

Разница стиля изображений А и В:

$$L_{style}^l(A, B) = \sum_{i,j} (G_{ij}^l(A) - G_{ij}^l(B))^2$$

$$G_{ij}^l(A) = \sum_k A_{ik}^l A_{jk}^l$$

A_{ik}^l — значение i -го фильтра на k -ой позиции l -го слоя сети

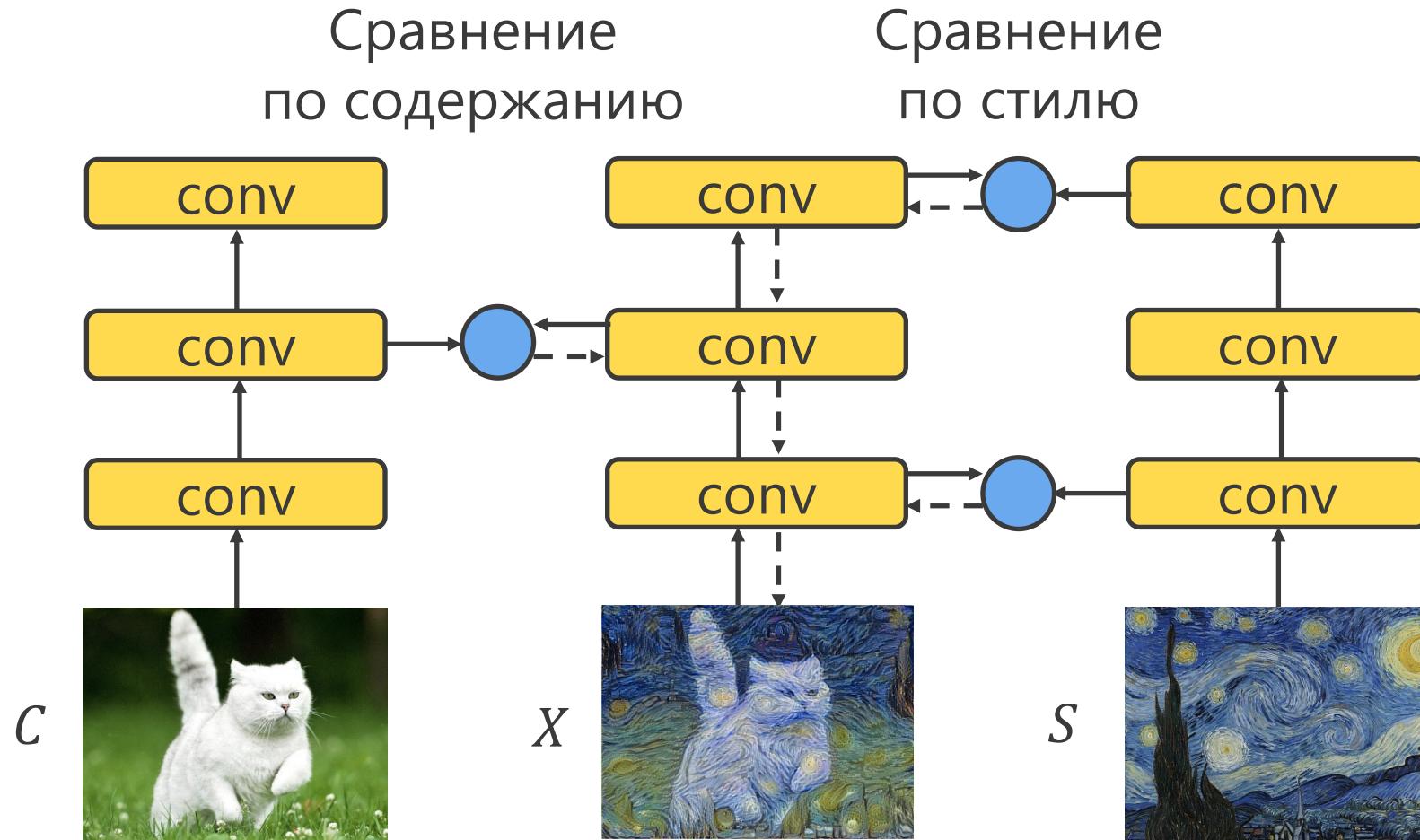
$G_{ij}^l(A)$ — скалярное произведение i -го и j -го фильтров

Сравнение по стилю

Свойства $L_{style}^l(A, B)$:

- ▶ Стиль описывается корреляцией между фильтрами
- ▶ Если эти корреляции для изображений схожи, то их стили тоже похожи
- ▶ При этом содержание изображений может быть разным

Perceptual loss



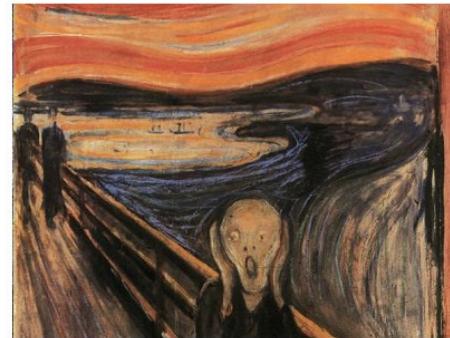
Perceptual loss

Исходное
изображение



+

Стилевое
изображение



Итоговое
изображение



C

S

X

Perceptual loss

Посчитаем сходство итогового изображения X с исходным C и стилем S изображениями:

$$L(C, S, X) = \alpha L_{content}(C, X) + \beta L_{style}(S, X)$$

α, β — настраиваемые параметры, как правило $\frac{\alpha}{\beta} \approx 10^{-3}$

С помощью градиентного спуска будем искать такое изображение X^* , что:

$$X^* = \arg \min_X L(C, S, X)$$

Пример



Пример

