



ОНЛАЙН ОБРАЗОВАНИЕ

Онлайн-образование

Меня хорошо видно && слышно?

Ставьте + , если все хорошо
Напишите в чат, если есть проблемы

Архитектуры свёрточных сетей



Ческидова Евгения

Deep Learning engineer
Wolf3d
t.me/fogside

Маршрут вебинара

От AlexNet до Inception-ResNet



Что модно сейчас



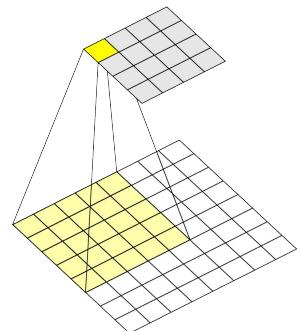
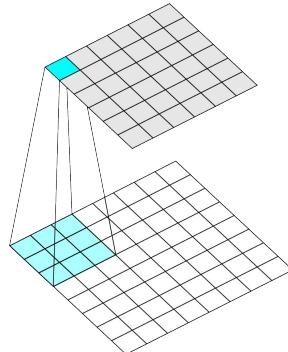
Итоги

Kernel size (filter size)

Типичные размеры свёрток

$1 \times 1 \leftrightarrow 11 \times 11$

1×1 - линейная комбинация каналов, не меняющая
пространственной информации
(да, тут тоже используется активация как и в других свёртках)





ОНЛАЙН ОБРАЗОВАНИЕ

От AlexNet до Inception-ResNet

MNIST: Handwriting recognition

50,000 картинок

28 x 28 x 1 (grayscale)

Цифры 0-9



LeNet: First Convnet for Images*

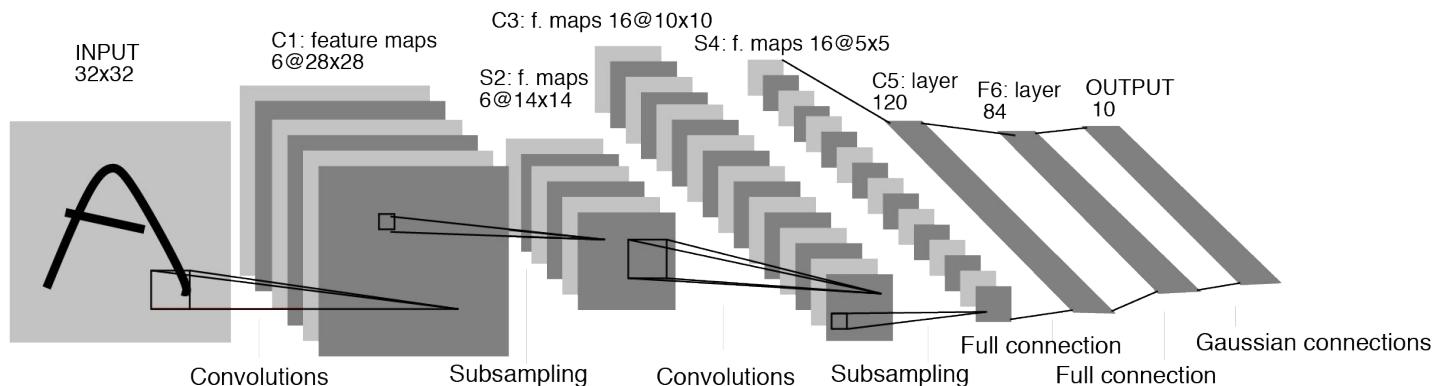
99% accuracy on MNIST (Yann LeCun 1998)

Использовались те же строительные блоки что сейчас

Convolutions, maxpooling, fully connected layers

Tanh activations after pooling layers (nowadays use RELU)

Weight updates through backpropagation

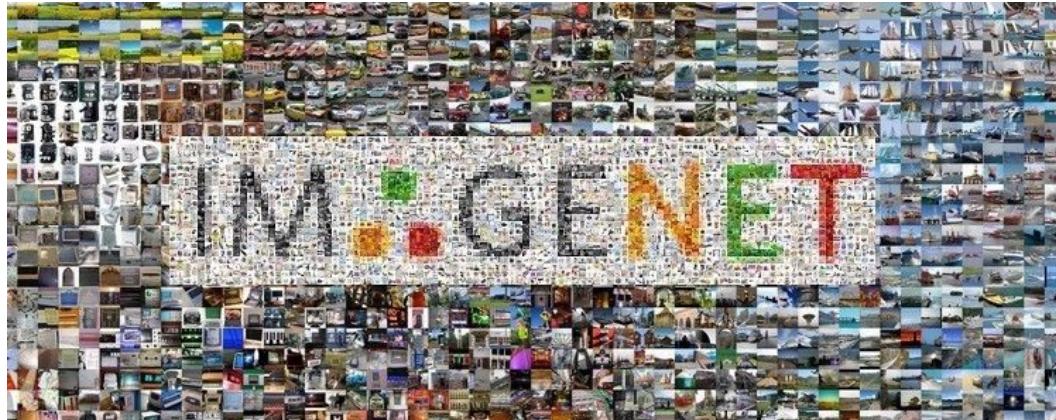


ImageNet

14 млн картинок, 22 тыс категорий

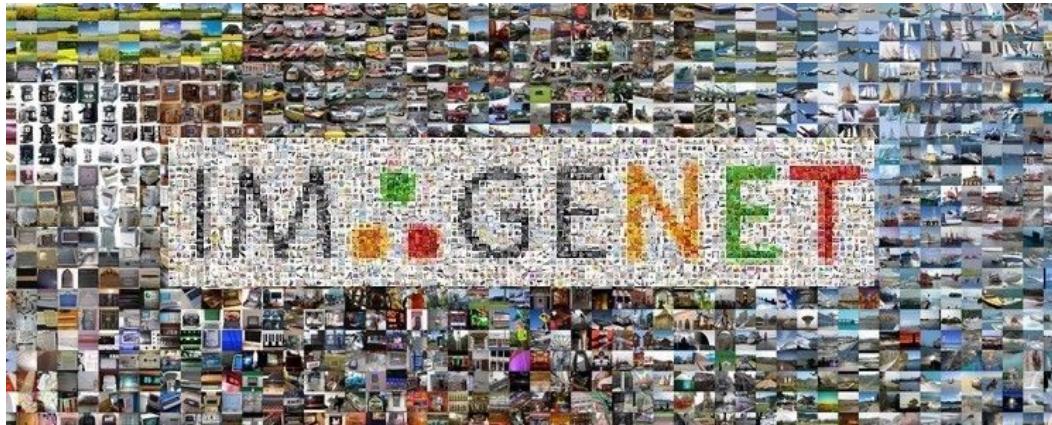
Challenge subset:

1.2 млн картинок, 1000 категорий

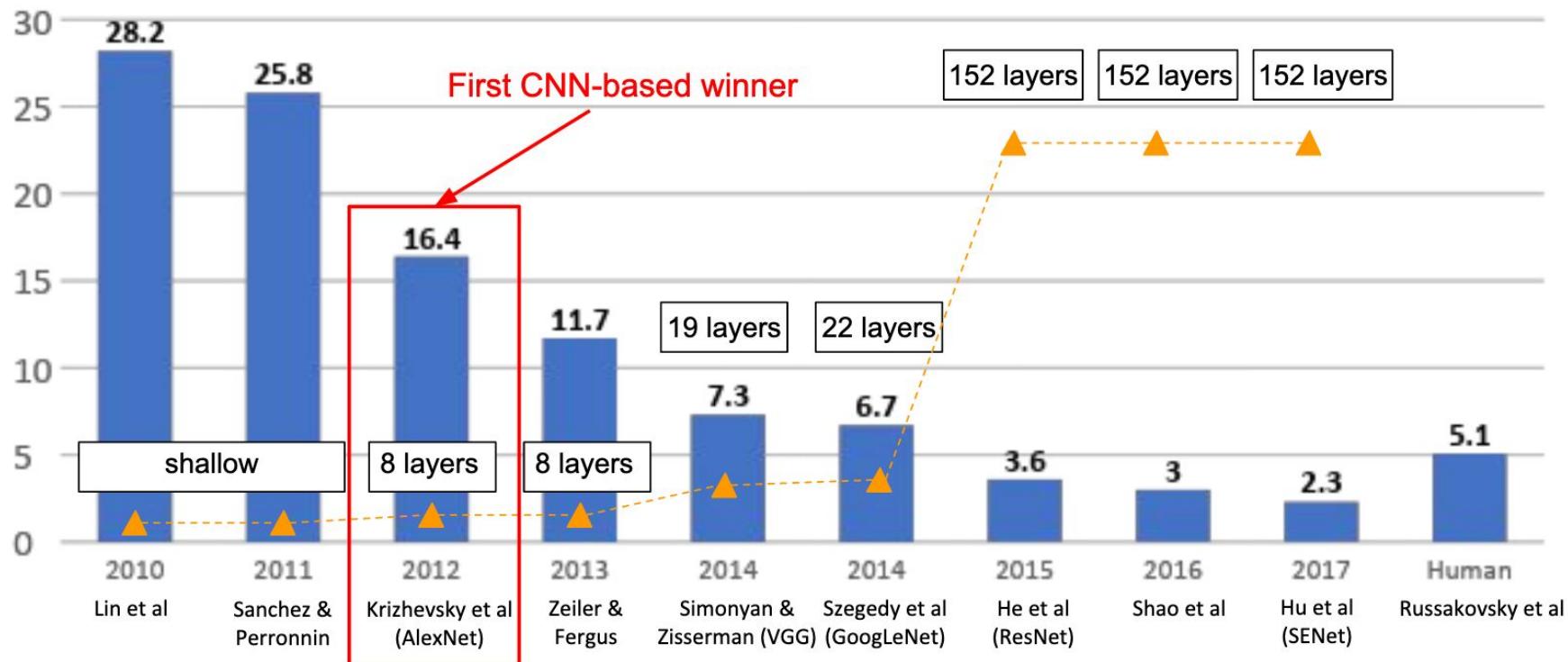


ImageNet

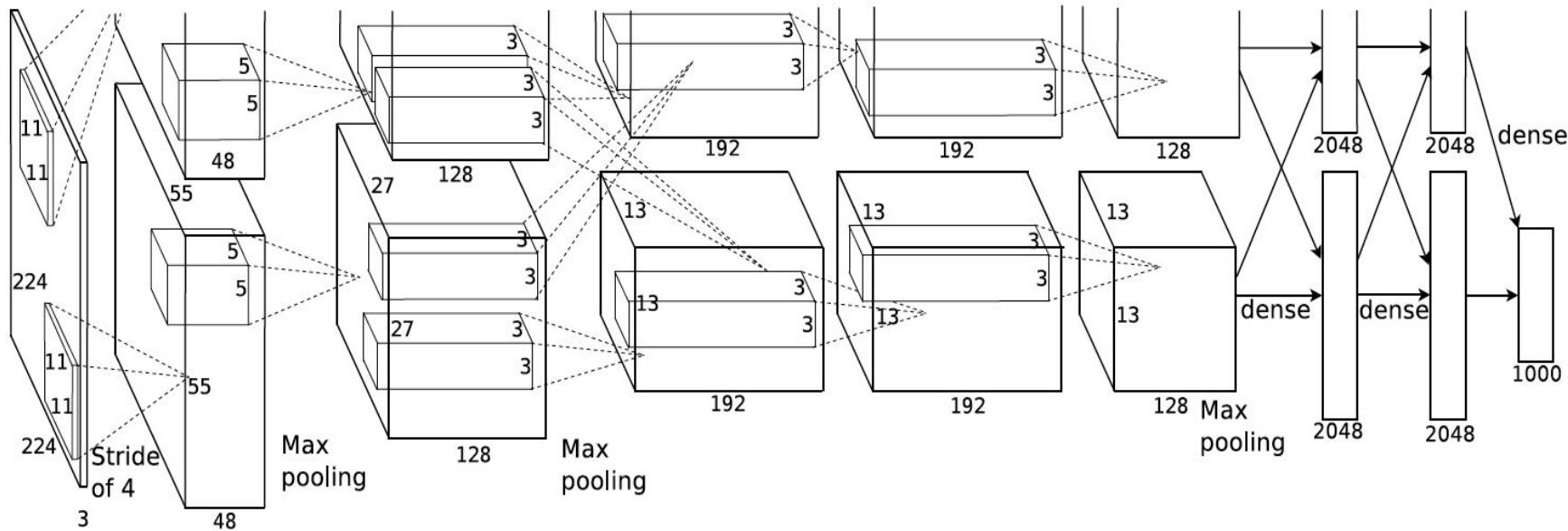
- Лейблы соответствуют словам и синсетам из WordNet
- Для аннотации использовали Amazon MT



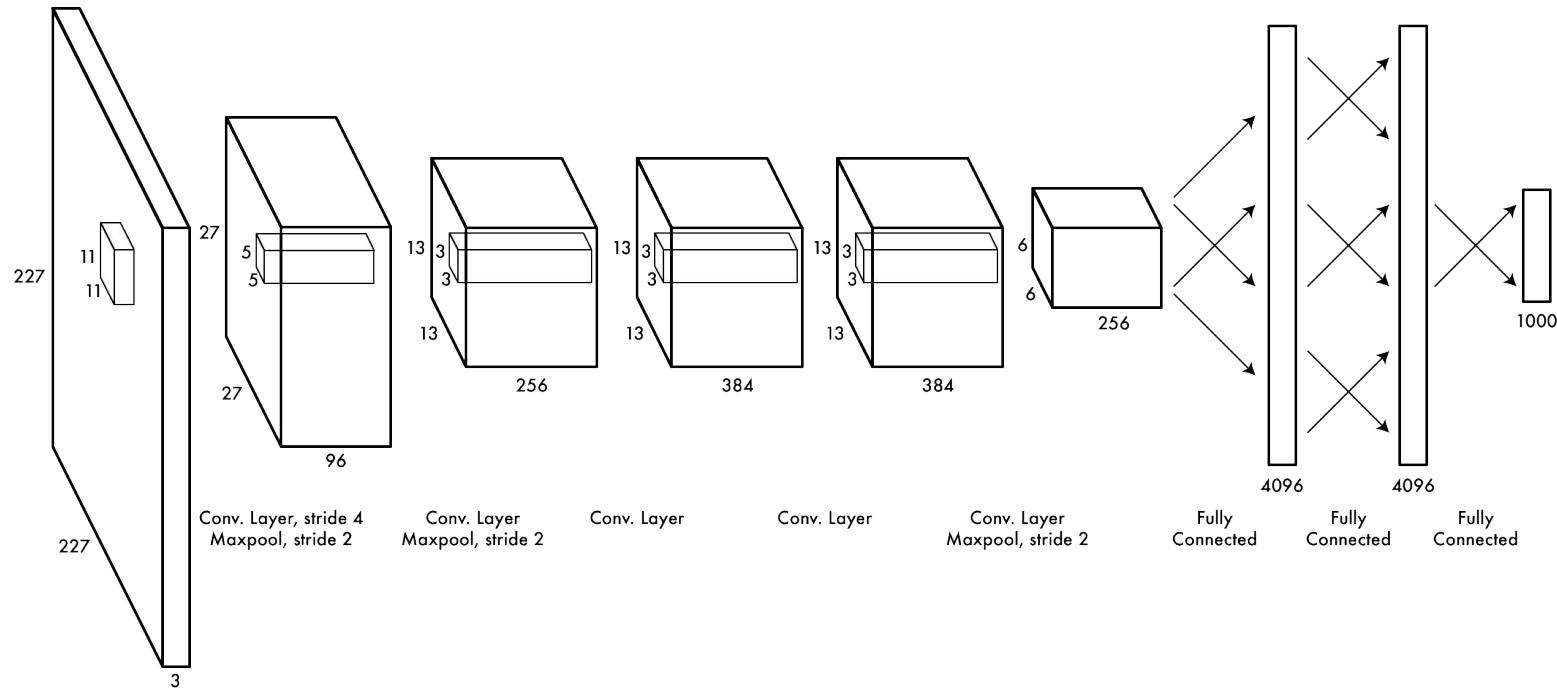
- ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



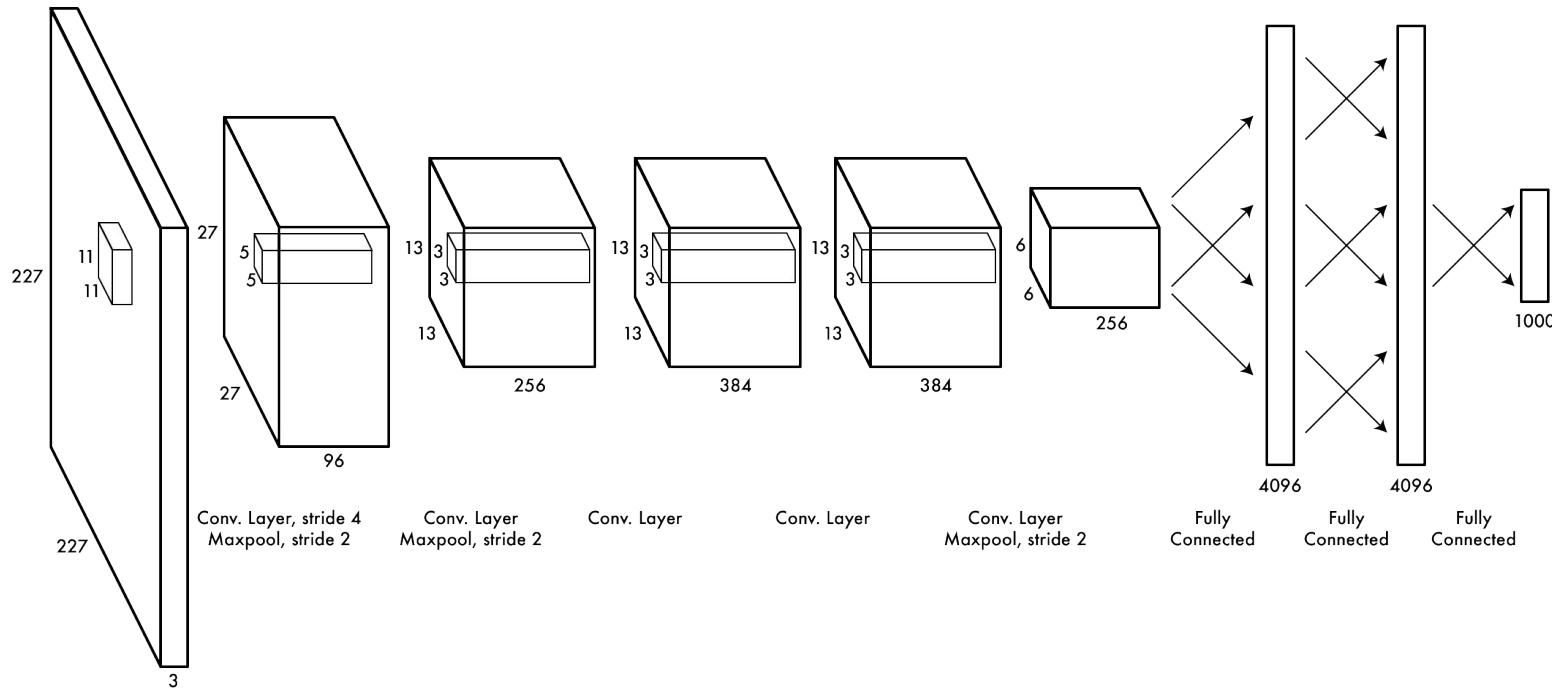
AlexNet: первое свёрточное решение ImageNet challenge



AlexNet: версия с 1 GPU



AlexNet: версия с 1 GPU

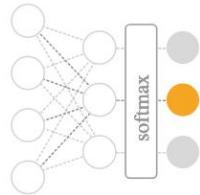
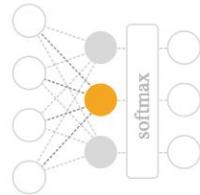


Следующее улучшение на ImageNet -- ZFNet -- уменьшение размера фильтров, увеличение их количества

<https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

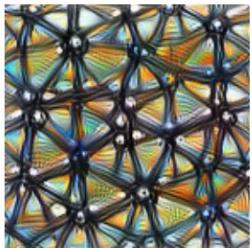
Что выучивают фильтры?

Optimization objectives



Neuron

`layer_n[x, y, z]`



Channel

`layer_n[:, :, :, z]`



Layer/DeepDream

`layer_n[:, :, :, :]2`



Class Logits

`pre_softmax[k]`

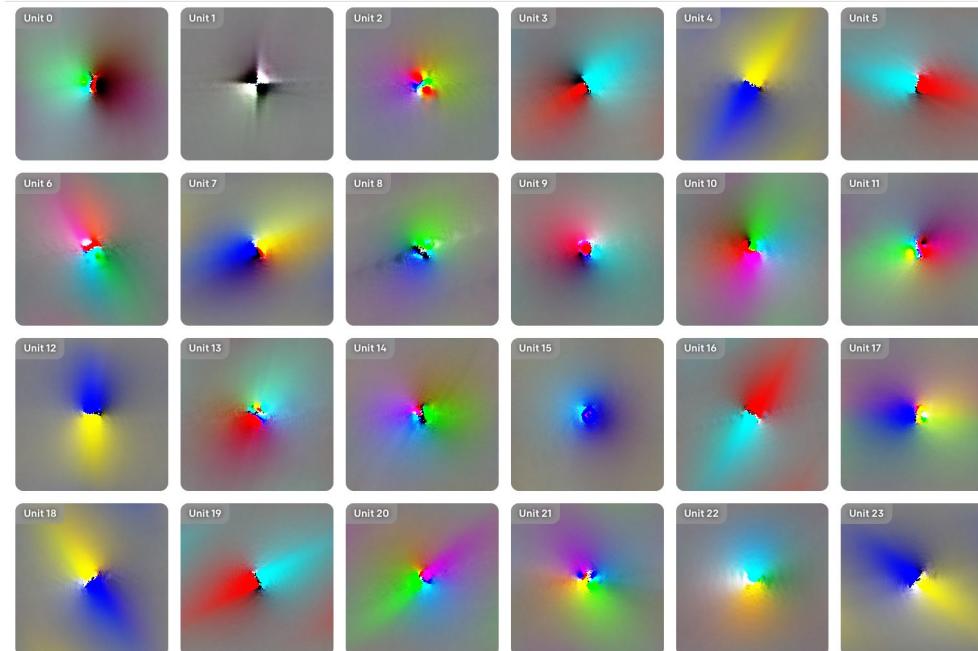


Class Probability

`softmax[k]`

Что выучивают фильтры?

Визуализация нейронов первого слоя AlexNet

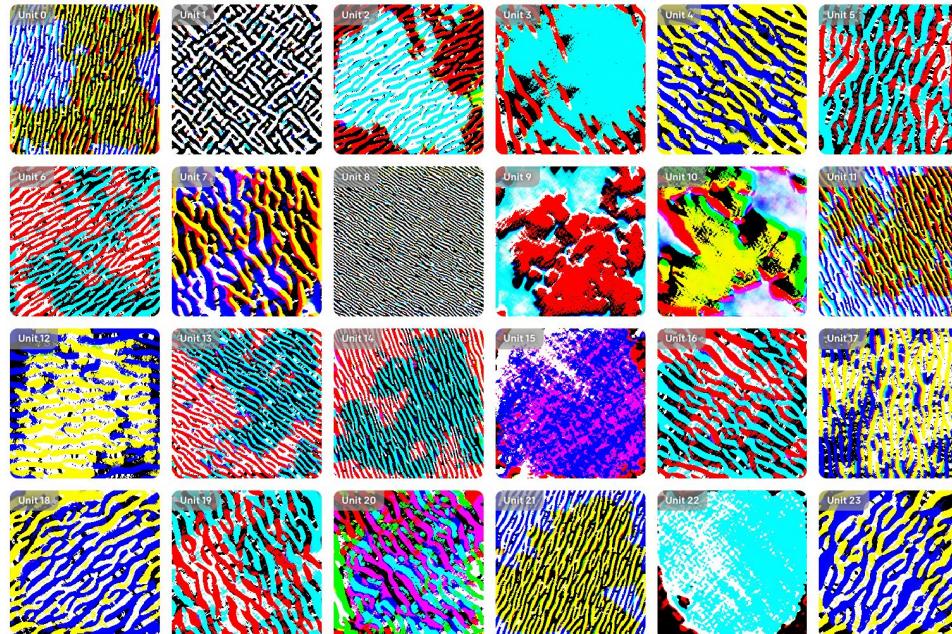


<https://microscope.openai.com/models>

<https://distill.pub/2017/feature-visualization/>

Что выучивают фильтры?

Визуализация каналов первого слоя AlexNet

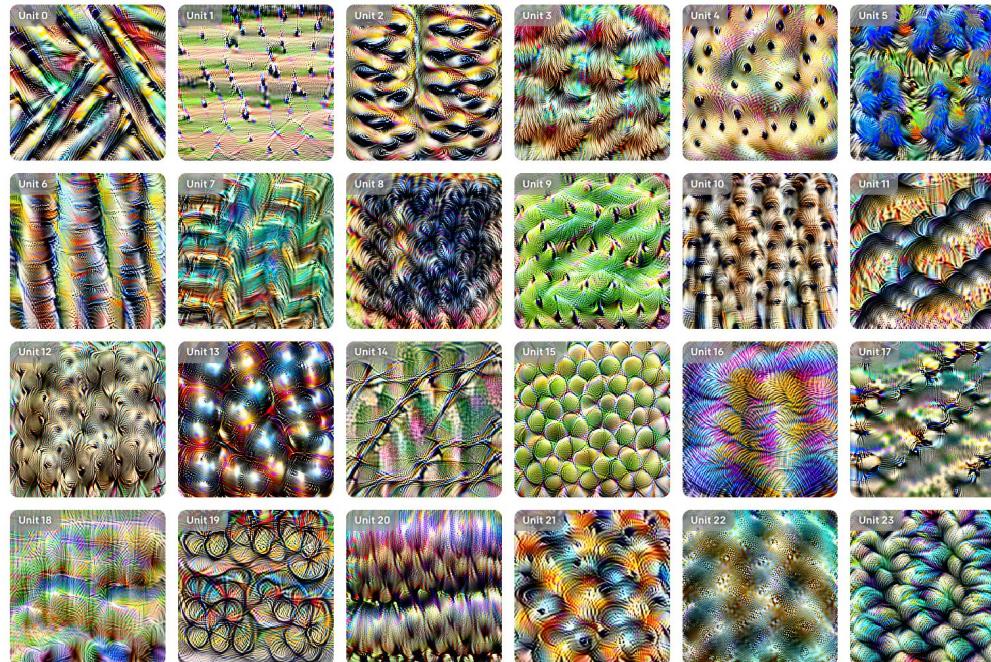


<https://microscope.openai.com/models>

<https://distill.pub/2017/feature-visualization/>

Что выучивают фильтры?

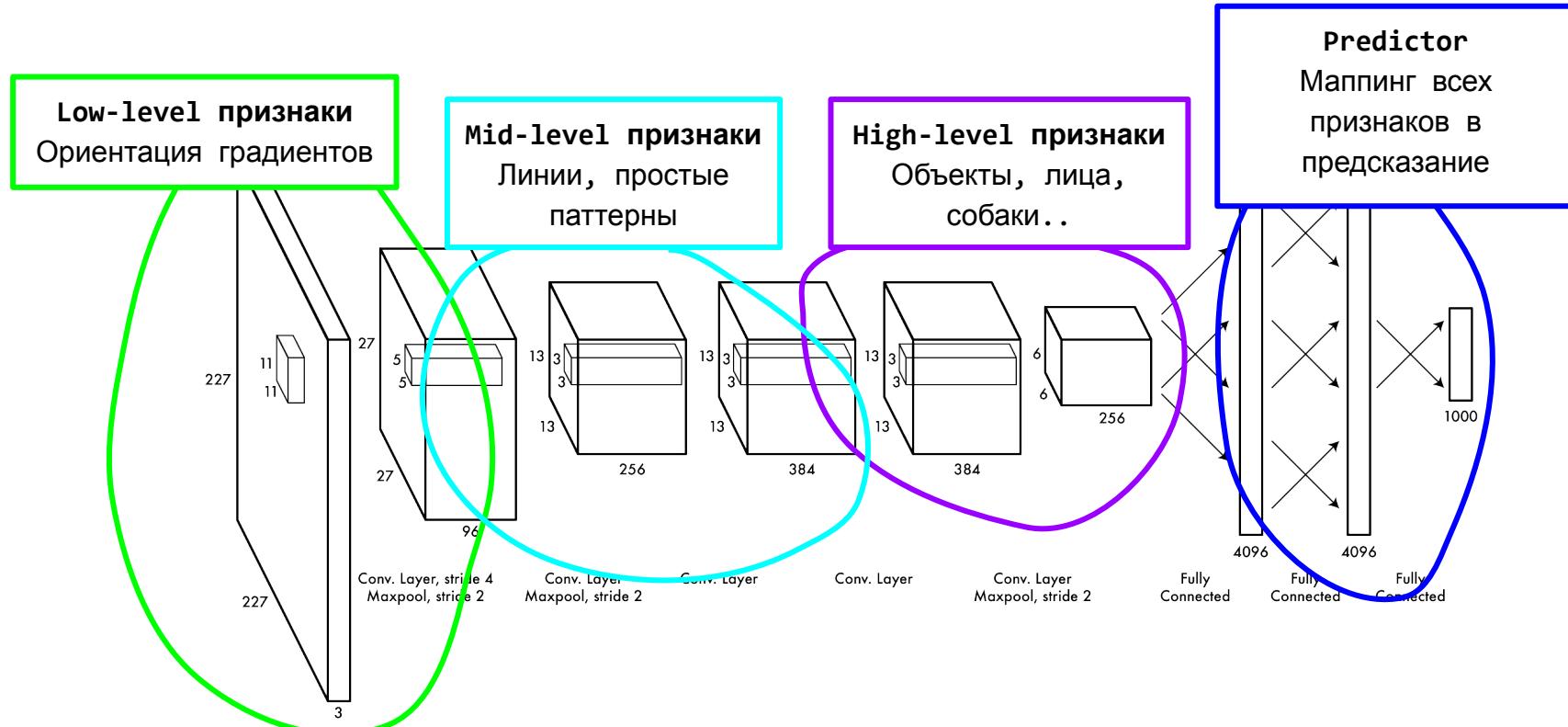
Визуализация каналов 5-ого сверточного слоя AlexNet



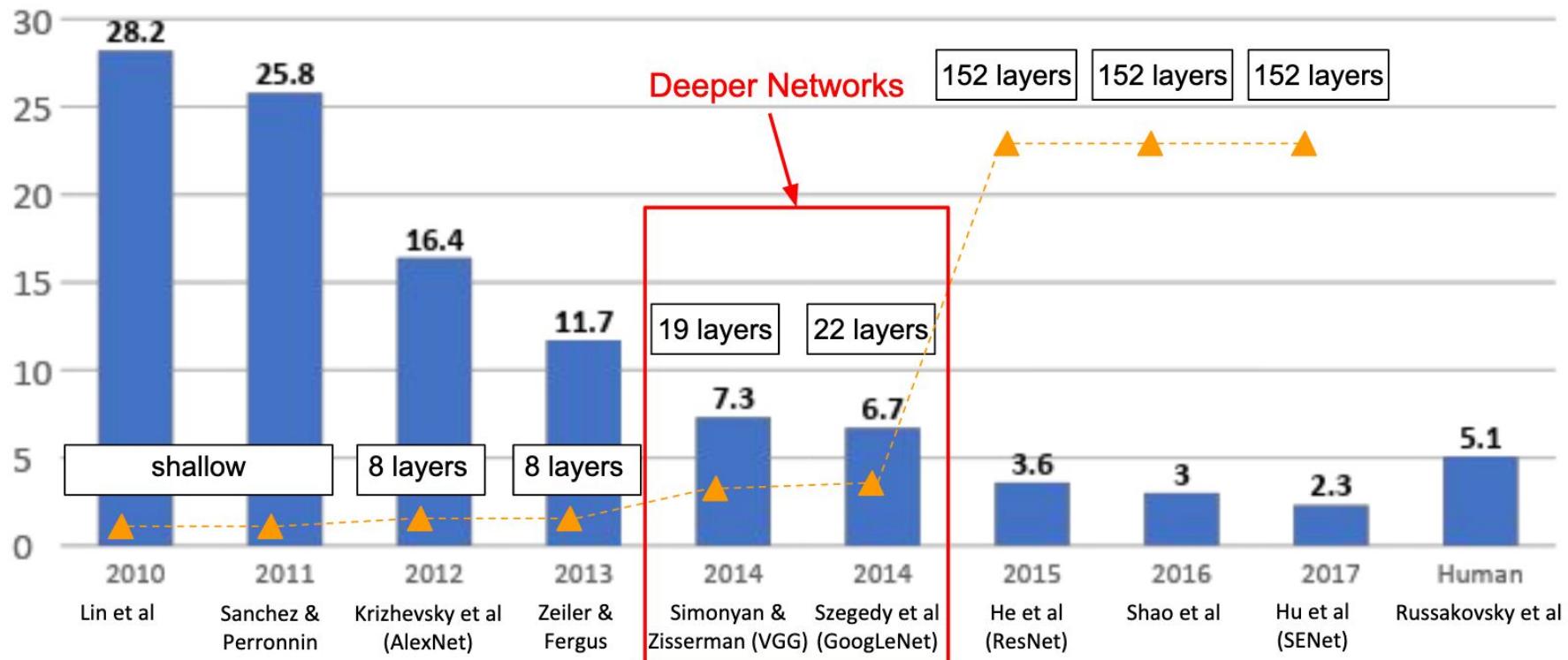
<https://microscope.openai.com/models>

<https://distill.pub/2017/feature-visualization/>

Полная картинка

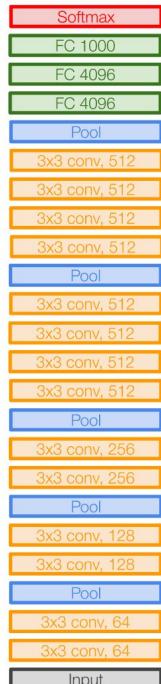
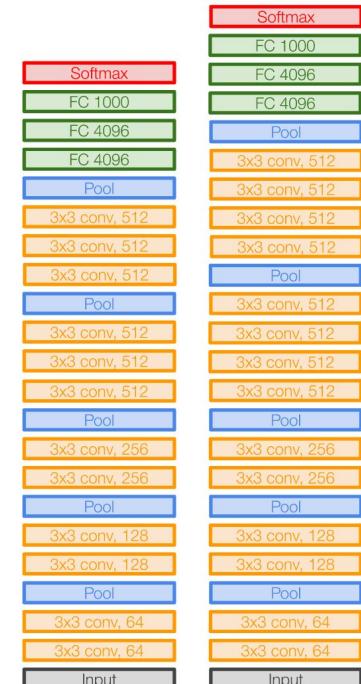
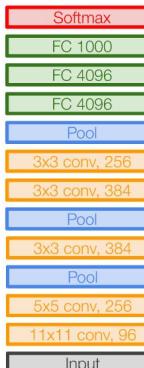
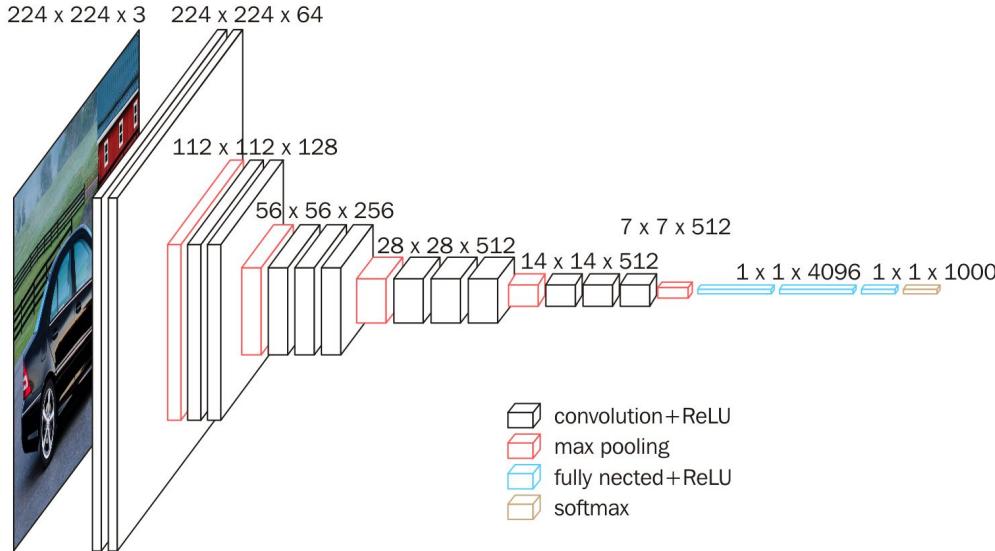


ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



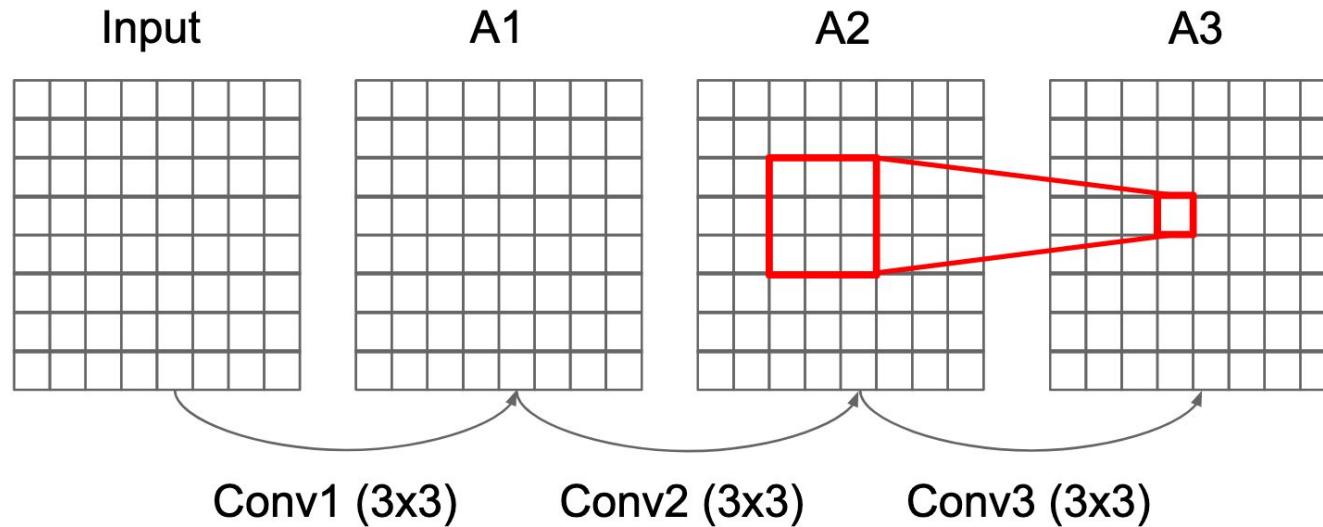
VGG: нейронки становятся глубже, фильтры меньше

Visual Geometry Group, Oxford, 2014



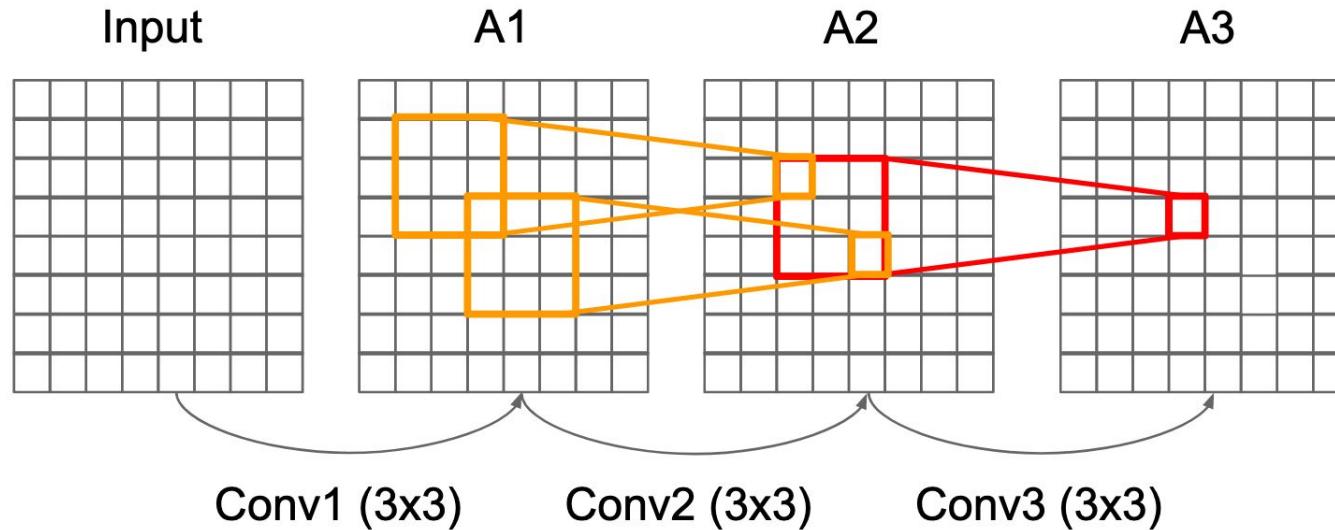
Почему так популярны свёртки 3x3 ?

- Рецептивное поле у 3х подряд идущих свёрток 3x3 такое же как у одной 7x7



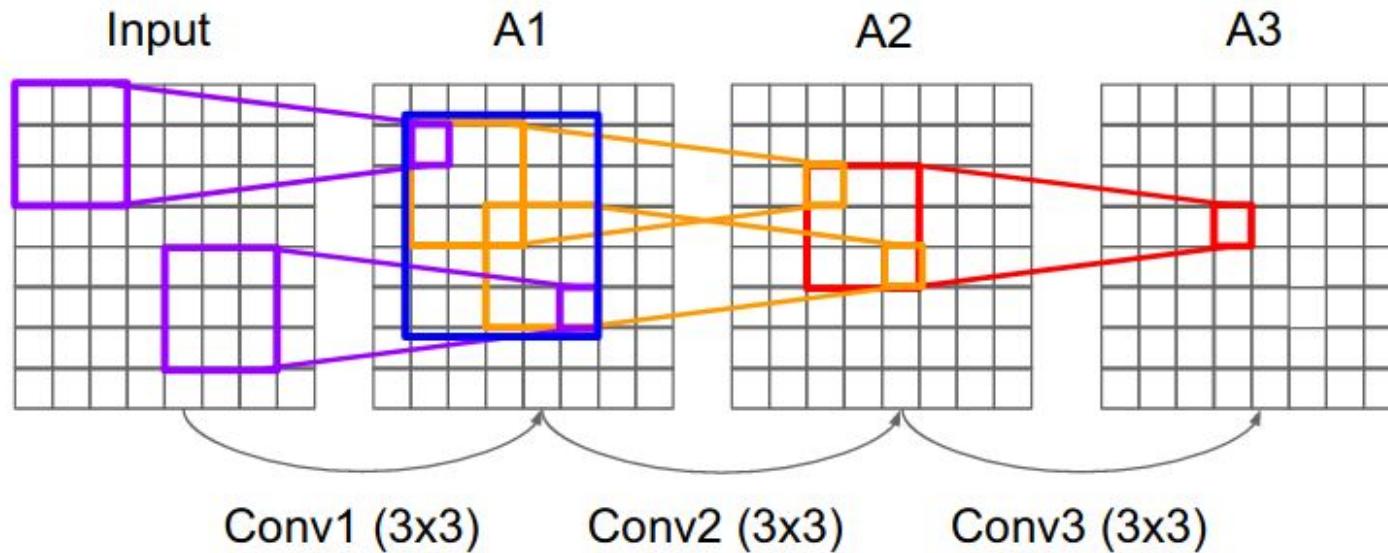
Почему так популярны свёртки 3x3 ?

- Рецептивное поле у 3х подряд идущих свёрток 3x3 такое же как у одной 7x7



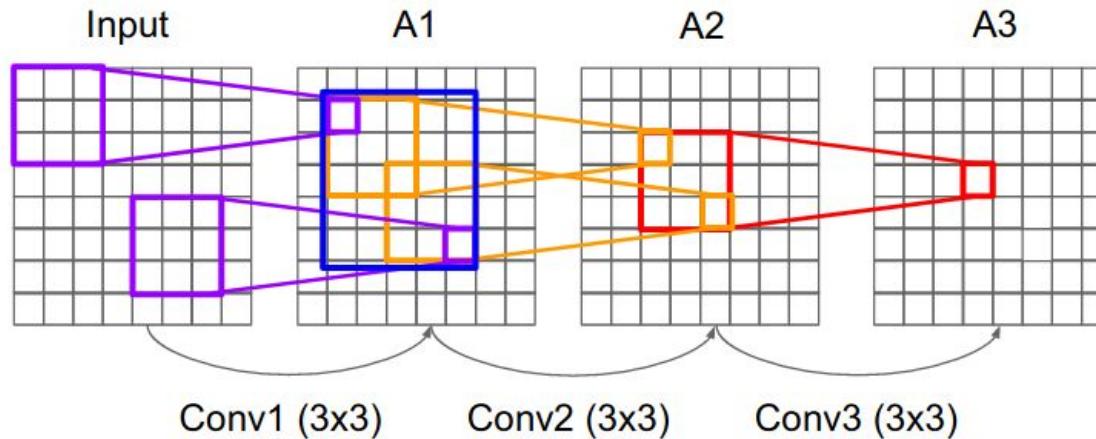
Почему так популярны свёртки 3x3 ?

- Рецептивное поле у 3х подряд идущих свёрток 3x3 такое же как у одной 7x7



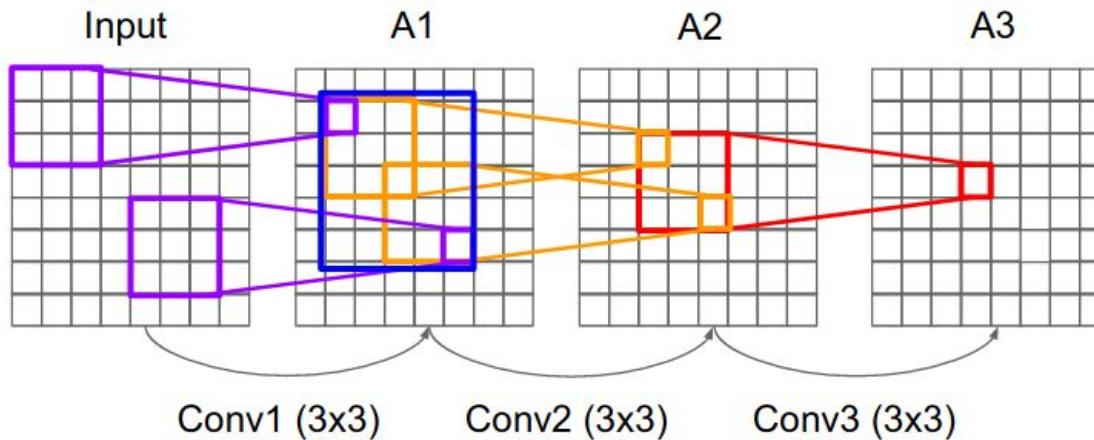
Почему так популярны свёртки 3x3 ?

- Рецептивное поле у 3х подряд идущих свёрток 3x3 такое же как у одной 7x7
- Сколько параметров мы экономим?



Почему так популярны свёртки 3x3 ?

- Рецептивное поле у 3х подряд идущих свёрток 3x3 такое же как у одной 7x7
- Сколько параметров мы экономим?
- $3*(3*C)^2$ vs $(7*C)^2$, где C - число каналов



Почему VGG не эффективно реализована?

INPUT: [224x224x3] memory: **224*224*3=150K** params: 0 (not counting biases)
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: $(3*3*3)*64 = 1,728$
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: $(3*3*64)*64 = 36,864$
POOL2: [112x112x64] memory: $112*112*64=800\text{K}$ params: 0
CONV3-128: [112x112x128] memory: $112*112*128=1.6\text{M}$ params: $(3*3*64)*128 = 73,728$
CONV3-128: [112x112x128] memory: $112*112*128=1.6\text{M}$ params: $(3*3*128)*128 = 147,456$
POOL2: [56x56x128] memory: $56*56*128=400\text{K}$ params: 0
CONV3-256: [56x56x256] memory: $56*56*256=800\text{K}$ params: $(3*3*128)*256 = 294,912$
CONV3-256: [56x56x256] memory: $56*56*256=800\text{K}$ params: $(3*3*256)*256 = 589,824$
CONV3-256: [56x56x256] memory: $56*56*256=800\text{K}$ params: $(3*3*256)*256 = 589,824$
POOL2: [28x28x256] memory: $28*28*256=200\text{K}$ params: 0
CONV3-512: [28x28x512] memory: $28*28*512=400\text{K}$ params: $(3*3*256)*512 = 1,179,648$
CONV3-512: [28x28x512] memory: $28*28*512=400\text{K}$ params: $(3*3*512)*512 = 2,359,296$
CONV3-512: [28x28x512] memory: $28*28*512=400\text{K}$ params: $(3*3*512)*512 = 2,359,296$
POOL2: [14x14x512] memory: $14*14*512=100\text{K}$ params: 0
CONV3-512: [14x14x512] memory: $14*14*512=100\text{K}$ params: $(3*3*512)*512 = 2,359,296$
CONV3-512: [14x14x512] memory: $14*14*512=100\text{K}$ params: $(3*3*512)*512 = 2,359,296$
CONV3-512: [14x14x512] memory: $14*14*512=100\text{K}$ params: $(3*3*512)*512 = 2,359,296$
POOL2: [7x7x512] memory: $7*7*512=25\text{K}$ params: 0
FC: [1x1x4096] memory: 4096 params: $7*7*512*4096 = \mathbf{102,760,448}$
FC: [1x1x4096] memory: 4096 params: $4096*4096 = 16,777,216$
FC: [1x1x1000] memory: 1000 params: $4096*1000 = 4,096,000$

Note:

Most memory is in early CONV

Most params are in late FC

TOTAL memory: 24M * 4 bytes \approx 96MB / image (only forward! ~ 2 for bwd)

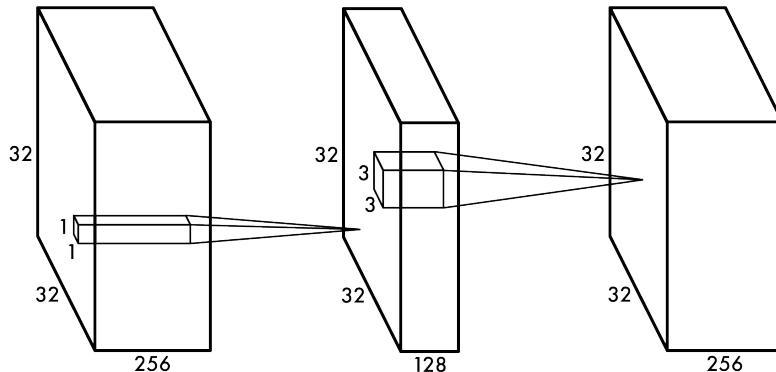
TOTAL params: 138M parameters

Почему VGG не эффективно реализована?

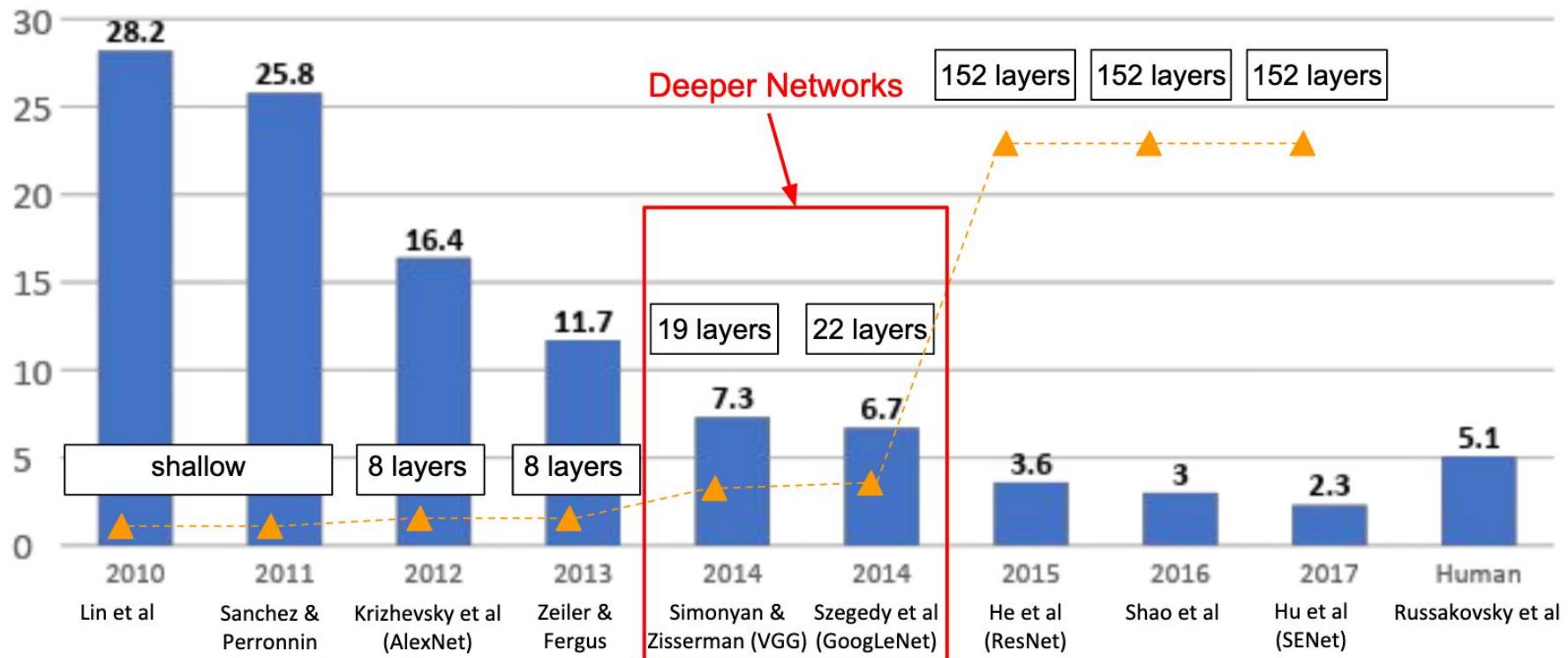
Потому что стакать подряд сверточные слои 3x3 не эффективно

Идея компрессии сверточного слоя по каналам (Lin et al.):

- Используем свертку 3x3 для извлечения пространственной информации
- Сжимаем информацию о каналах сверткой 1x1



ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

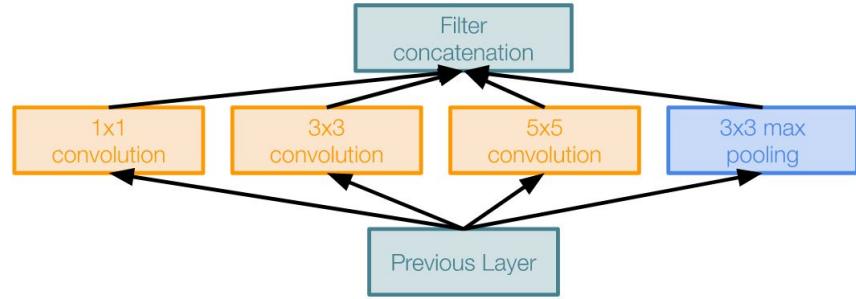


GoogleNet (Inception-v1)

Ключевые улучшения:

1. Применения разного размера фильтров на одном уровне процессинга

$1 \times 1, 3 \times 3, 5 \times 5$

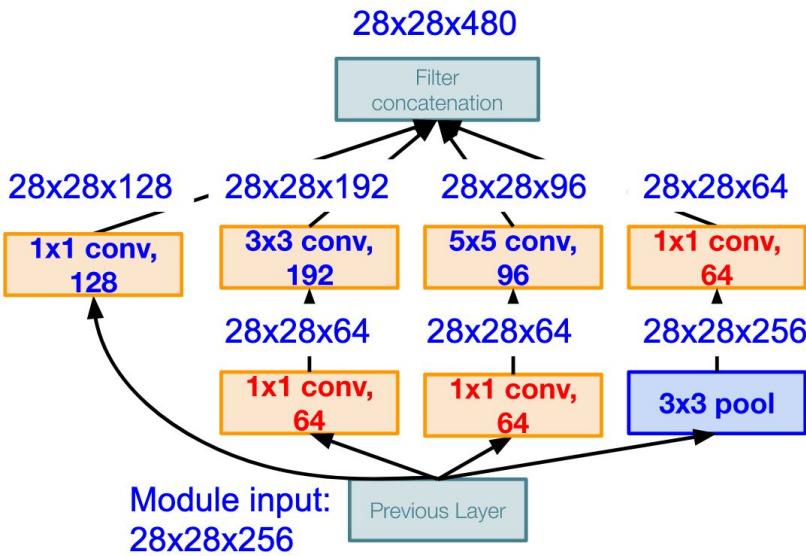


Naive Inception module

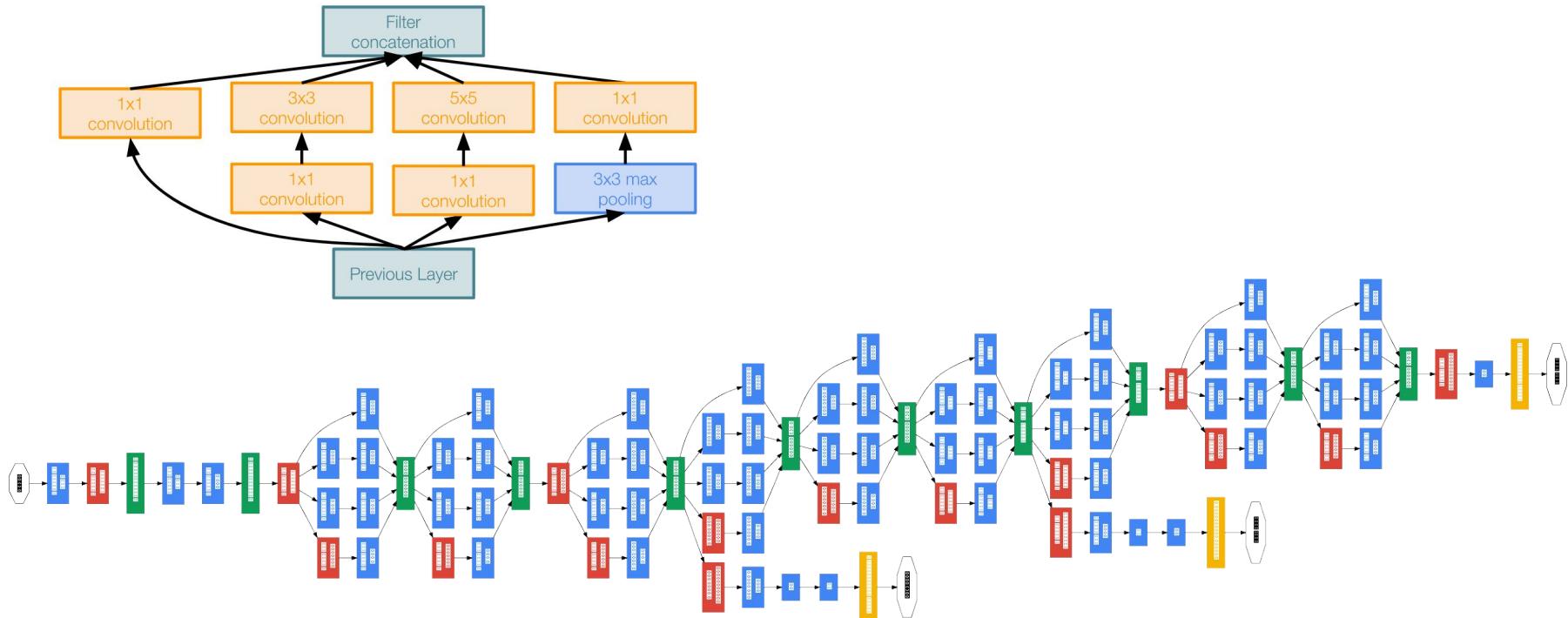
GoogleNet (Inception-v1)

Ключевые улучшения:

1. Применения разного размера фильтров на одном уровне процессинга
 1×1 , 3×3 , 5×5
2. 1×1 свёртки для компрессии
3. Average Pooling вместо FC
4. Всего 5 млн параметров
(138 млн у VGG16;
~60 млн у AlexNet)



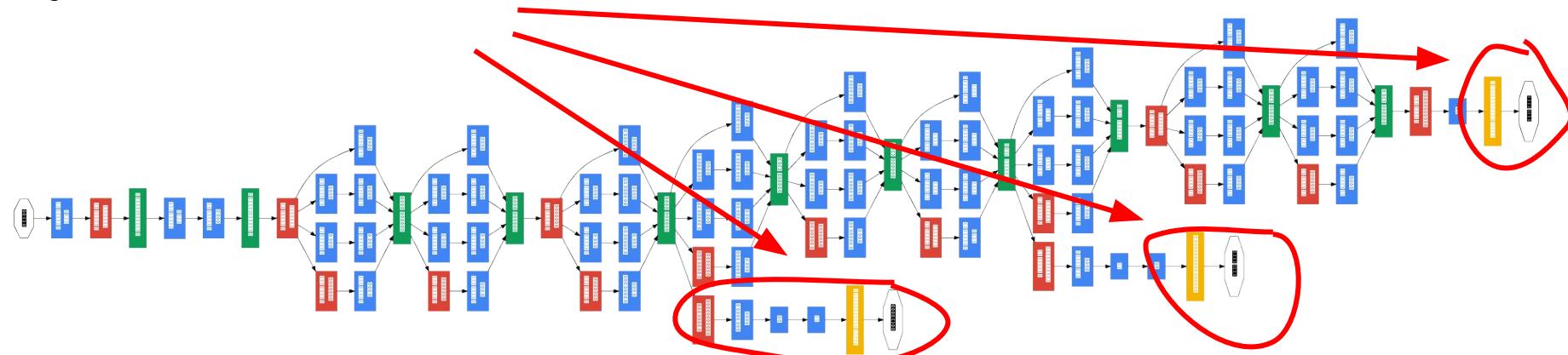
GoogleNet (Inception-v1)



GoogleNet (Inception-v1)

Несколько выходов для борьбы
с затухающими/взрывающимися градиентами

AvgPool-1x1Conv-FC-FC-Softmax

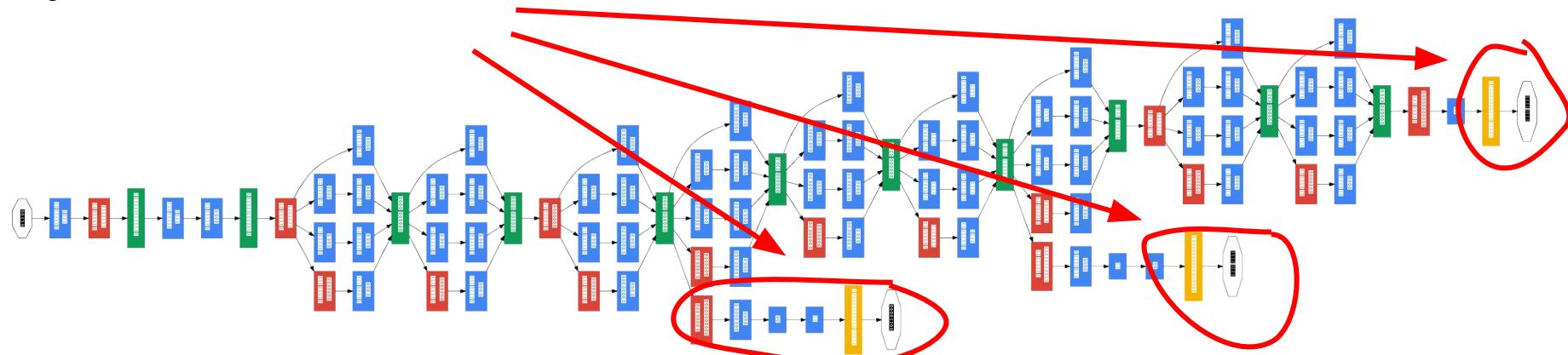


GoogleNet (Inception-v1)

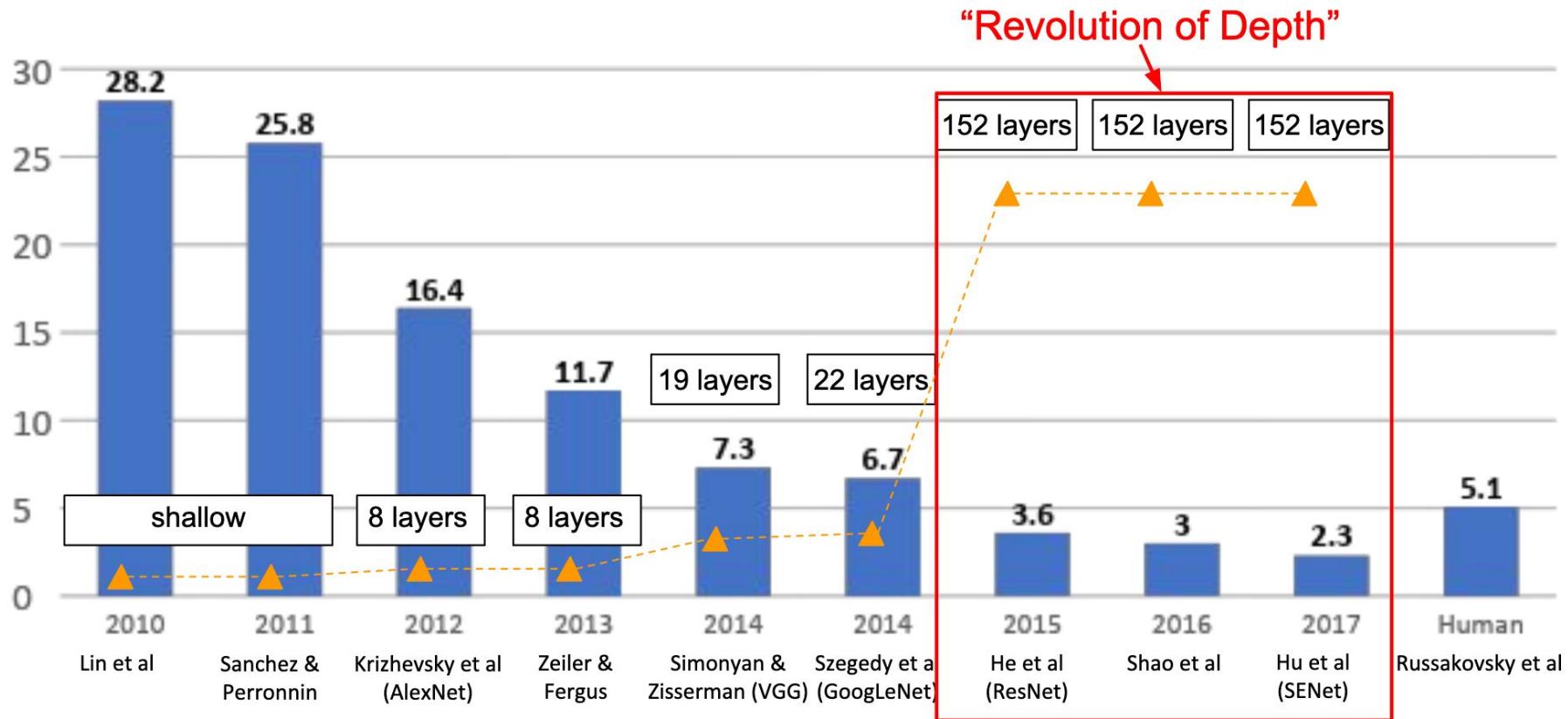
Итого 22 слоя
(считая Inception блок за один слой, не считая дополнительных выходов)

Несколько выходов для борьбы с затухающими/взрывающимися градиентами

AvgPool-1x1Conv-FC-FC-Softmax

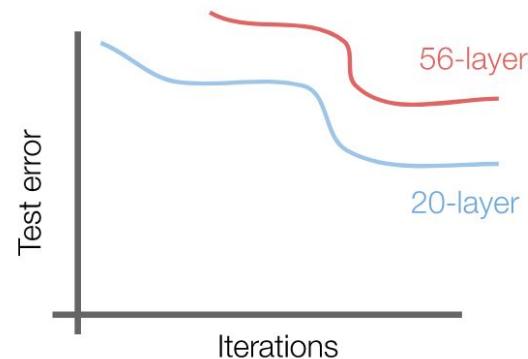
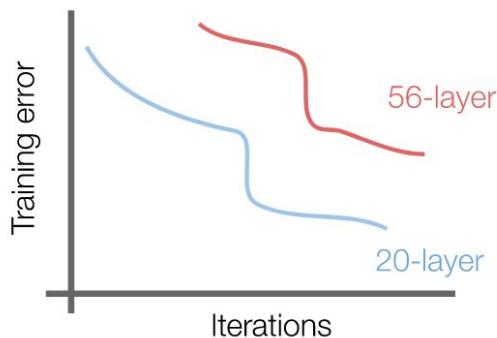


ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



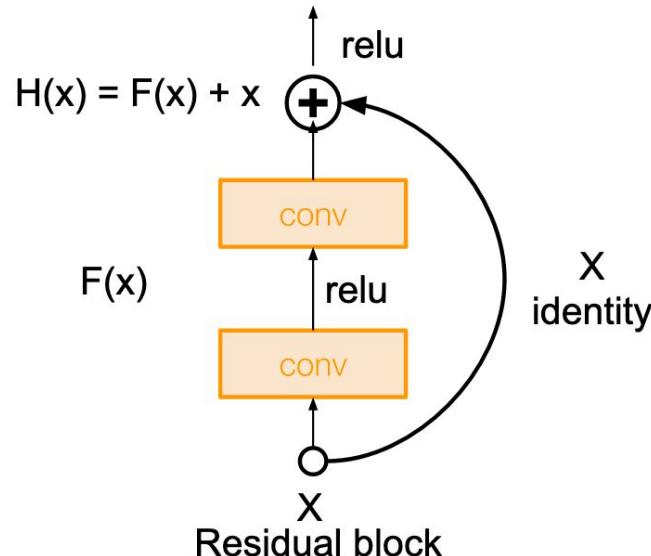
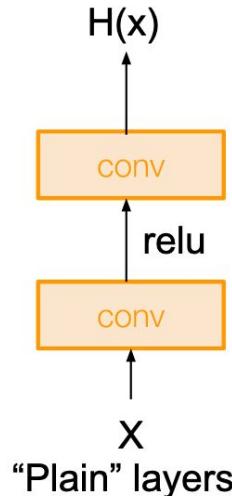
Предпосылки появления ResNet

- Результат на этих графиках train намекает, что плохой результат на test *не от переобучения 56-слойной сетки*



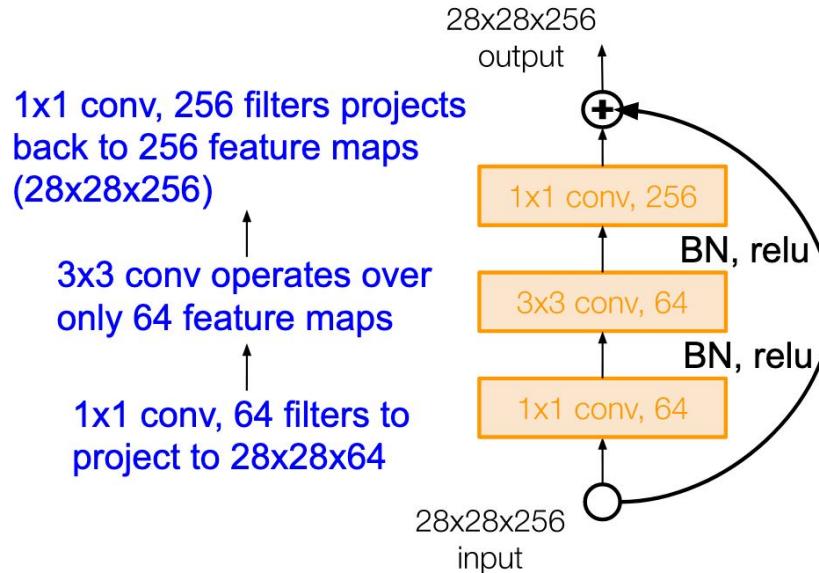
ResNet: для по-настоящему глубоких сетей

- В остаточных (residual) блоках сеть учит $F(x)$ - остаток, который необходимо прибавить к X



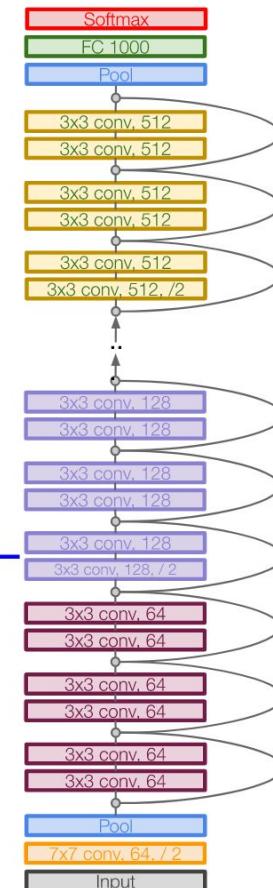
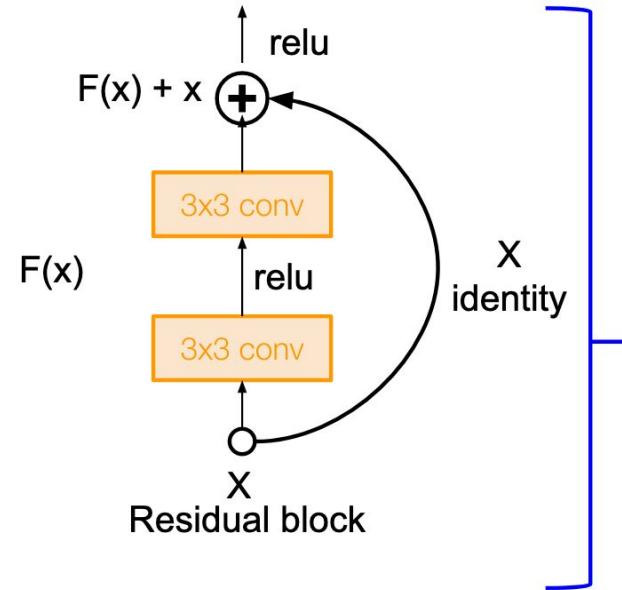
ResNet: для по-настоящему глубоких сетей

- ResNet 50+ использует трюк с компрессией из GoogleNet
- Размерность входа и выхода одна и та же

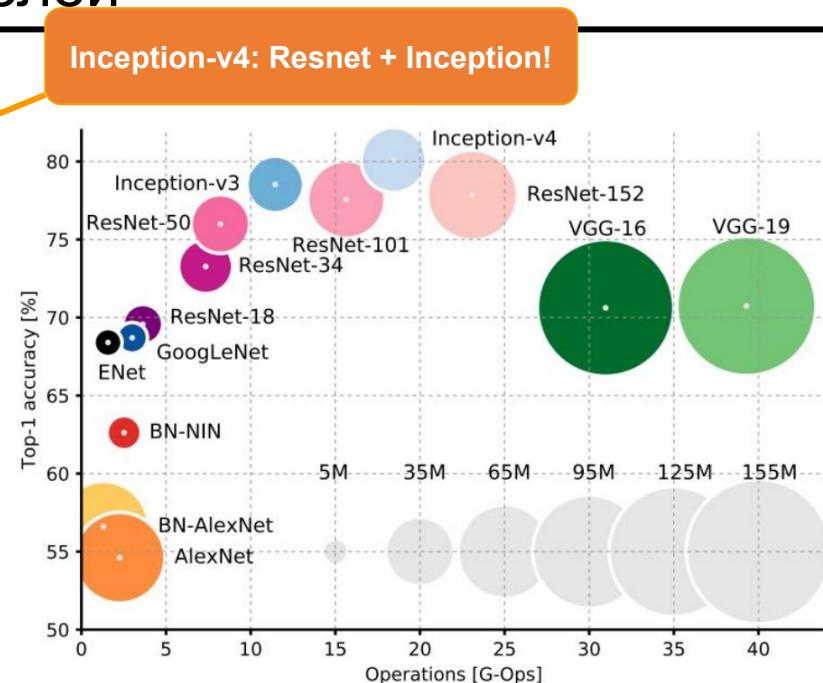
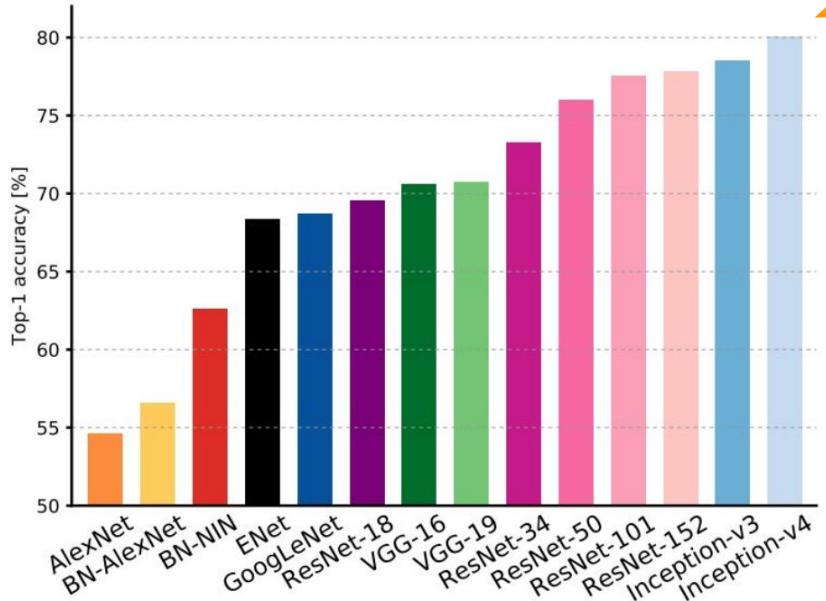


ResNet: для по-настоящему глубоких сетей

- Стакаем residual блоки
- Иногда удваиваем количество фильтров в них
- Иногда используем stride 2, чтобы уменьшить размер изображения ~ в 2 раза
- Всего один полно связанный слой в конце
- Одна большая дополнительная свертка 7x7 в самом начале
- Используем He инициализацию
- BN после каждой свертки

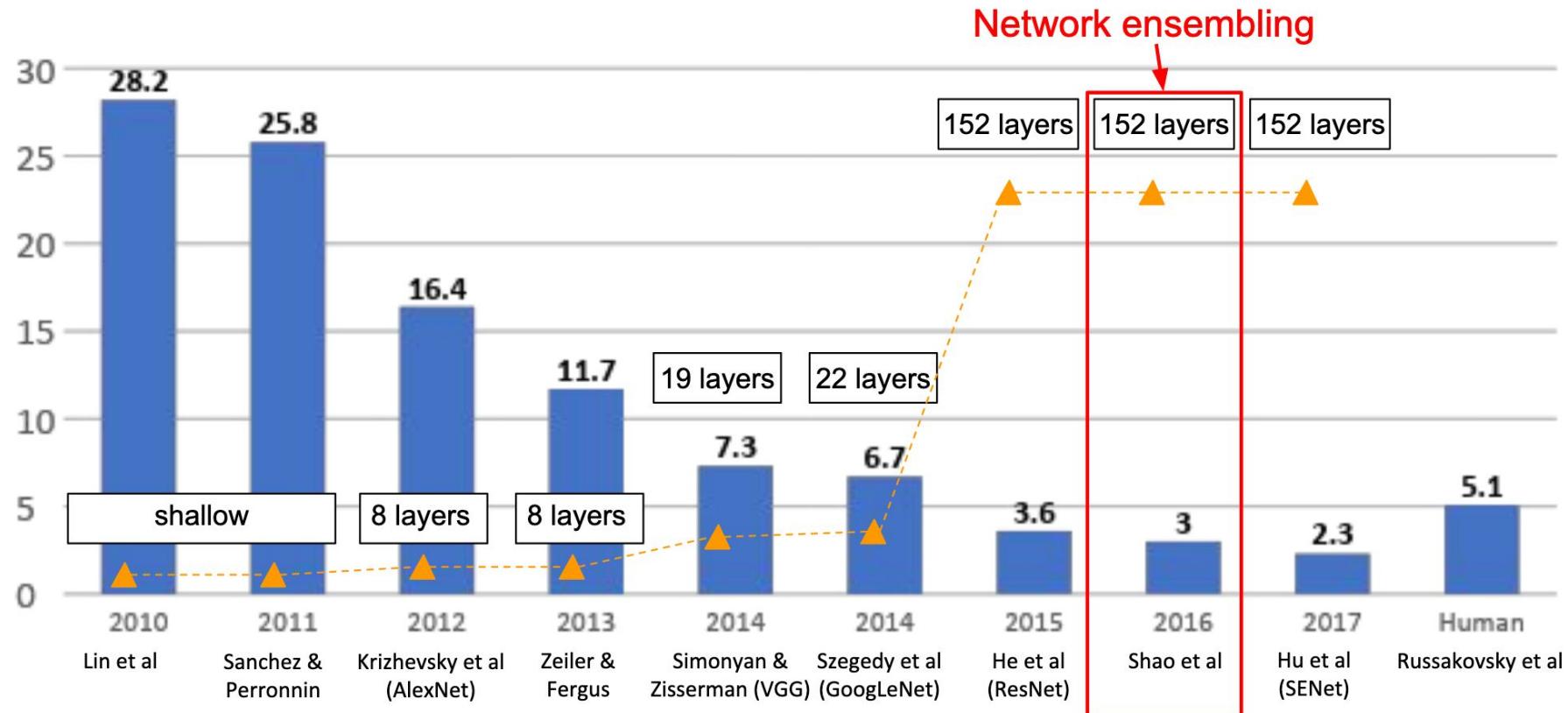


Сравнение эффективности моделей

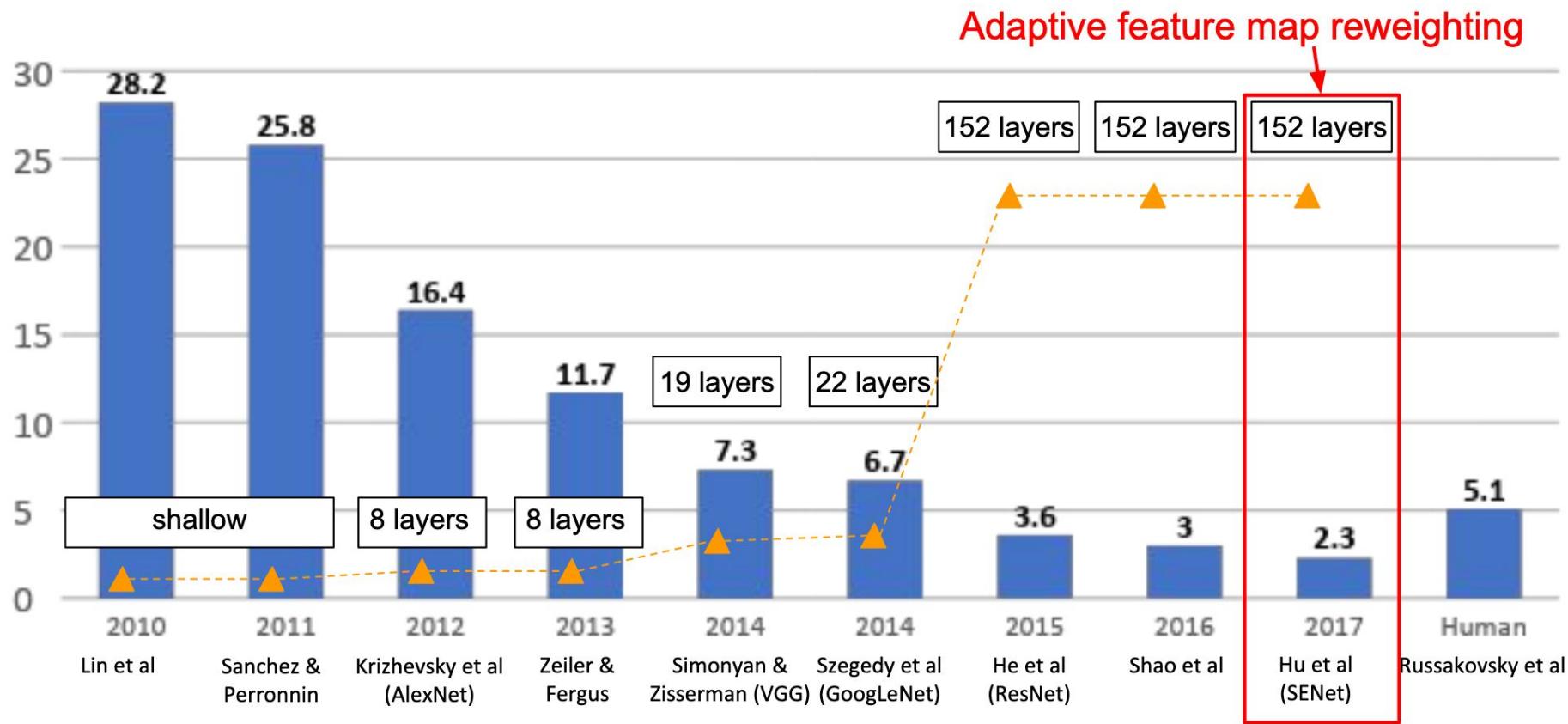


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



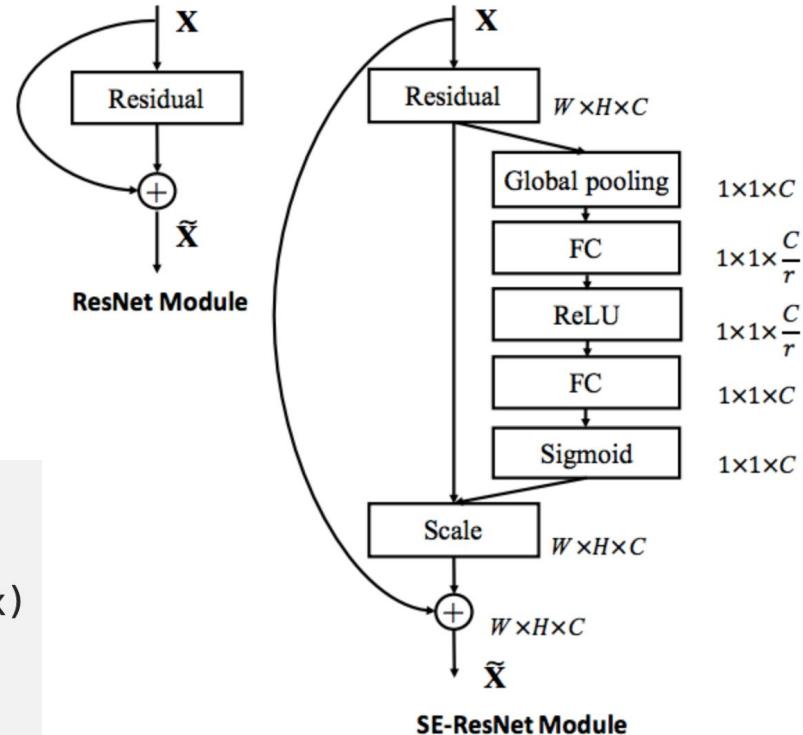
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Squeeze-and-Excitation Networks (SENet)

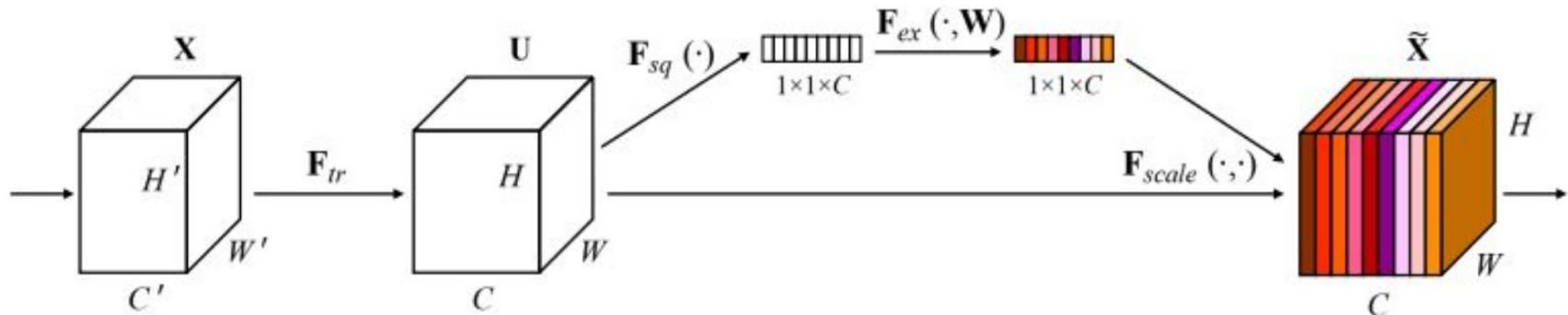
- Простая идея добавить обучаемые веса каждому каналу на выходе резнет блока
- ЭТОТ ТРЮК ПРИМЕНИМ И К Inception блоку, к любым другим блокам.

```
def se_block(in_block, ch, ratio=16):
    x = GlobalAveragePooling2D()(in_block)
    x = Dense(ch//ratio, activation='relu')(x)
    x = Dense(ch, activation='sigmoid')(x)
    return multiply()([in_block, x])
```



Vanilla ResNet Module vs the proposed SE-ResNet Module

Squeeze-and-Excitation Networks (SENet)



Как этот трюк улучшает существующие архитектуры

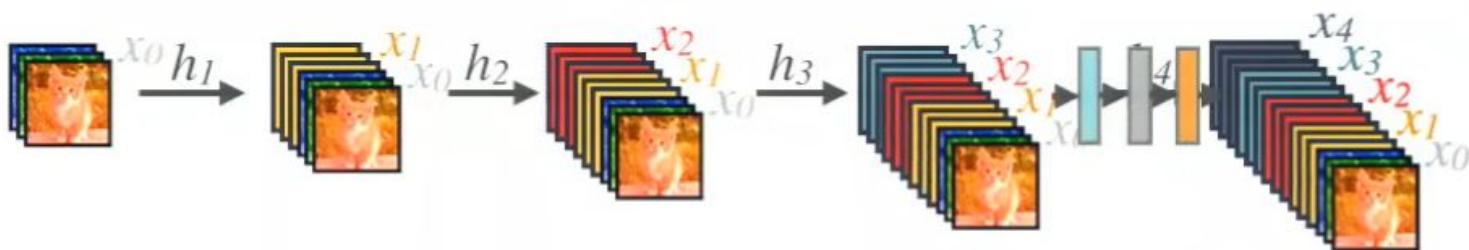
	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [9]	24.7	7.8	24.80	7.48	3.86	23.29 _(1.51)	6.62 _(0.86)	3.87
ResNet-101 [9]	23.6	7.1	23.17	6.52	7.58	22.38 _(0.79)	6.07 _(0.45)	7.60
ResNet-152 [9]	23.0	6.7	22.42	6.34	11.30	21.57 _(0.85)	5.73 _(0.61)	11.32
ResNeXt-50 [43]	22.2	-	22.11	5.90	4.24	21.10 _(1.01)	5.49 _(0.41)	4.25
ResNeXt-101 [43]	21.2	5.6	21.18	5.57	7.99	20.70 _(0.48)	5.01 _(0.56)	8.00
BN-Inception [14]	25.2	7.82	25.38	7.89	2.03	24.23 _(1.15)	7.14 _(0.75)	2.04
Inception-ResNet-v2 [38]	19.9 [†]	4.9 [†]	20.37	5.21	11.75	19.80 _(0.57)	4.79 _(0.42)	11.76

How SENets improve existing architectures

DenseNet

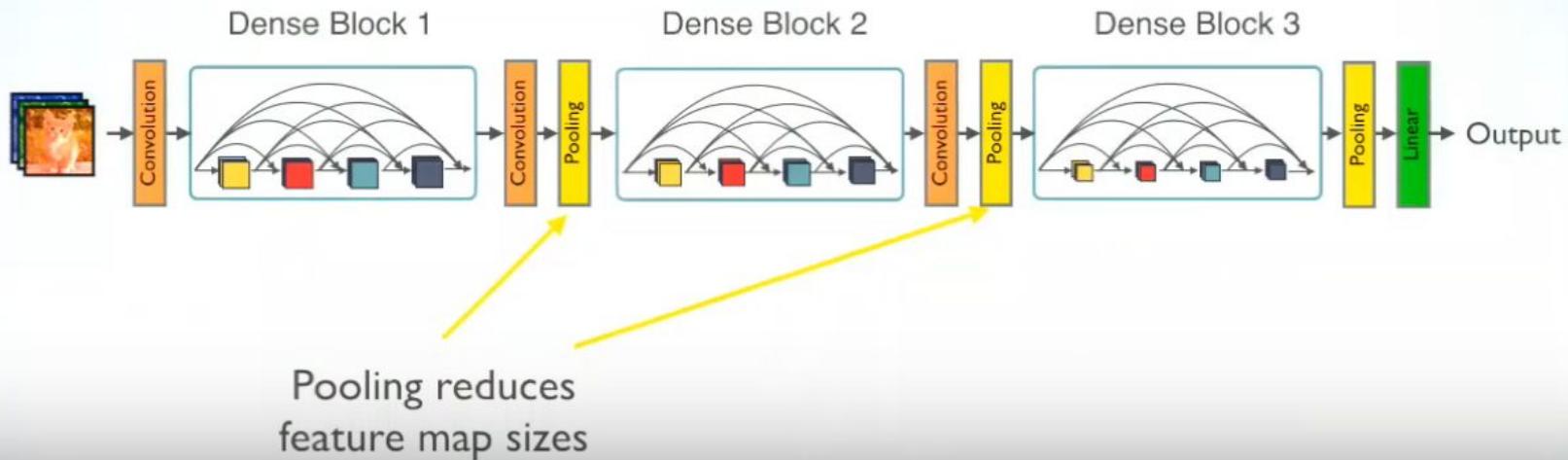


FORWARD PROPAGATION



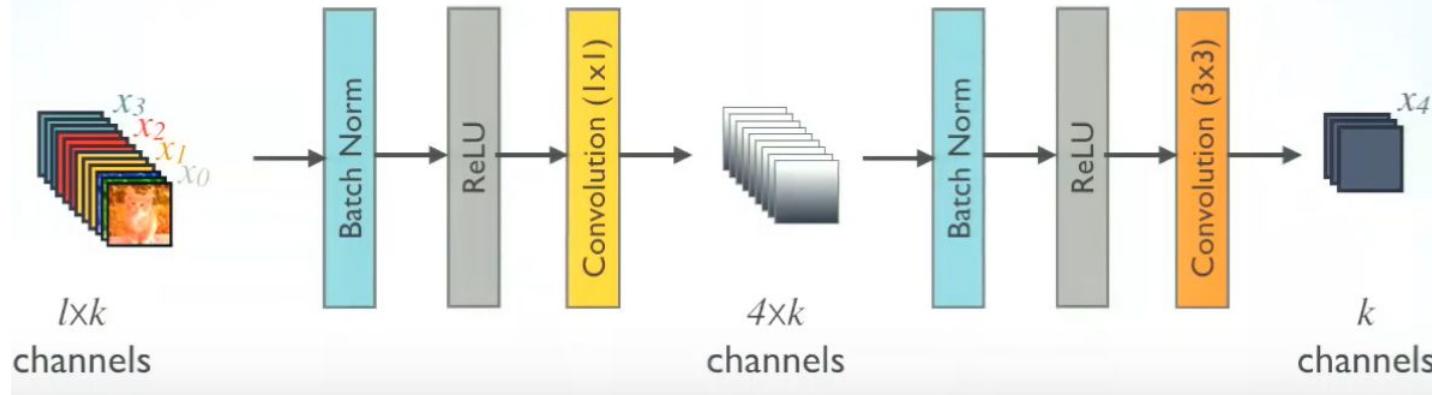
DenseNet

DENSENET

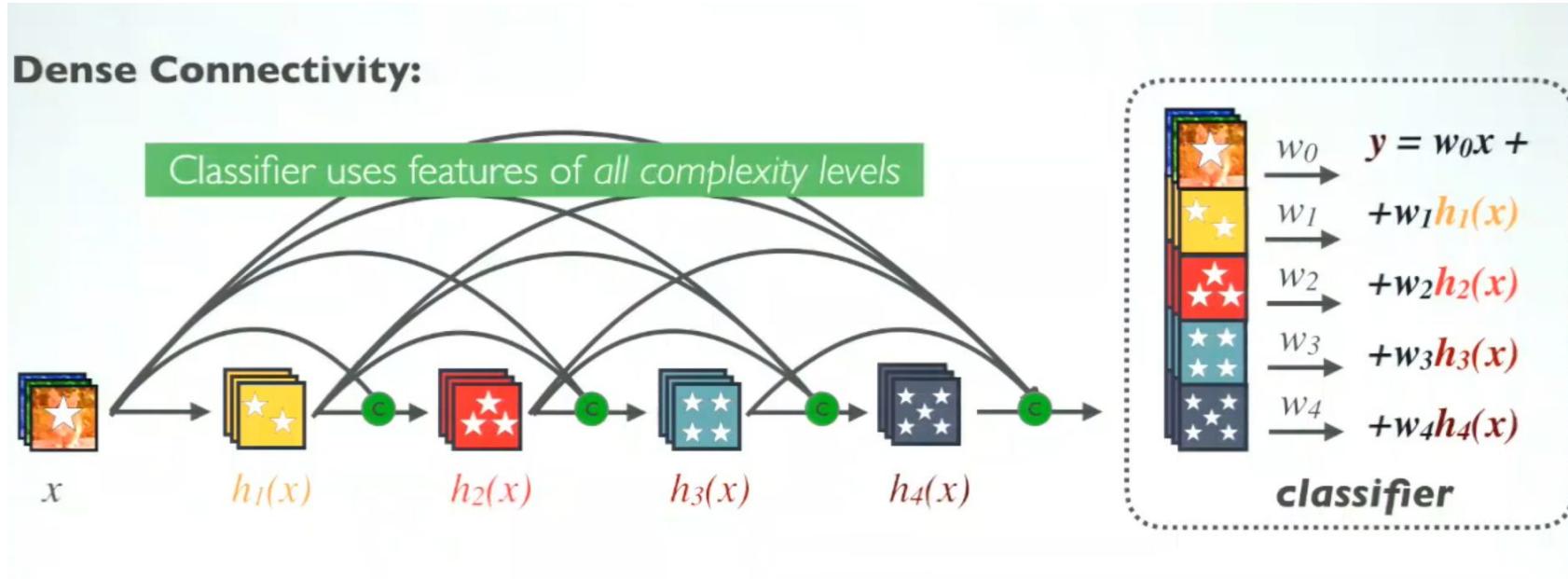


DenseNet

COMPOSITE LAYER IN DENSENET WITH BOTTLENECK LAYER



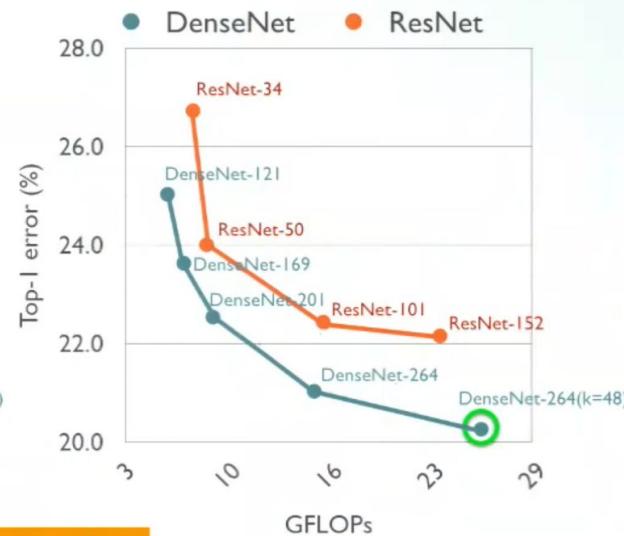
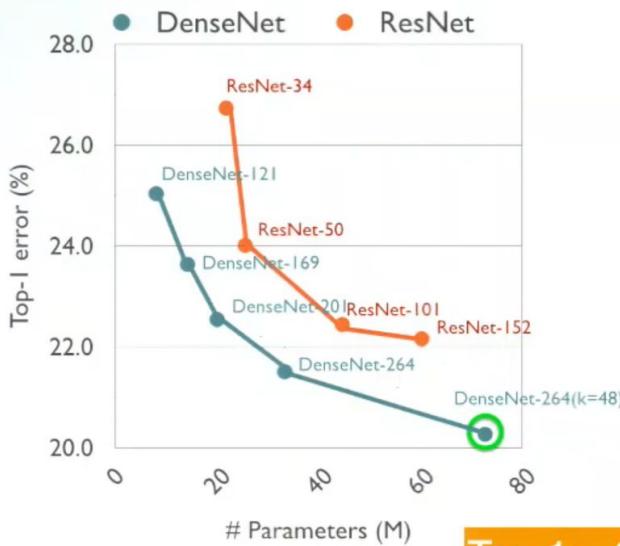
DenseNet



Works better on a smaller train data

DenseNet

RESULTS ON **IMAGENET**



Top-1: 20.27%
Top-5: 5.17%

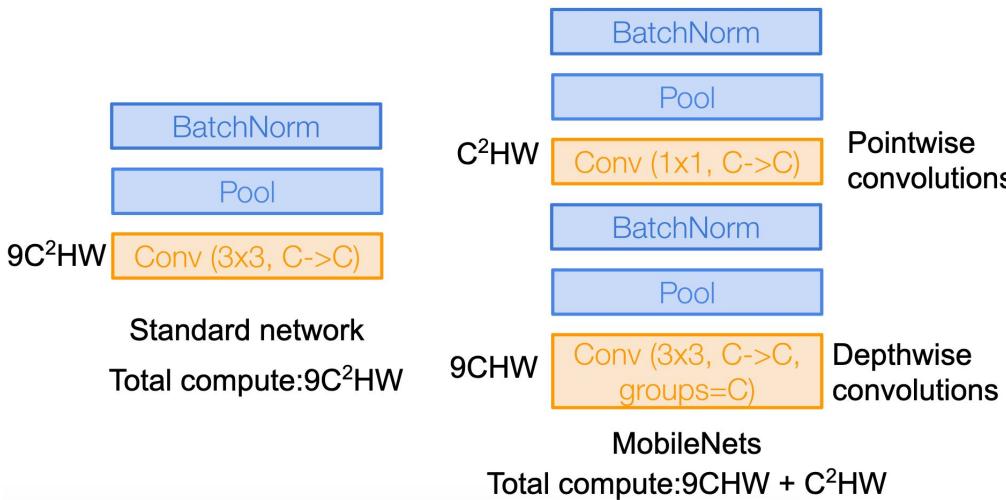


ОНЛАЙН ОБРАЗОВАНИЕ

Что модно сейчас

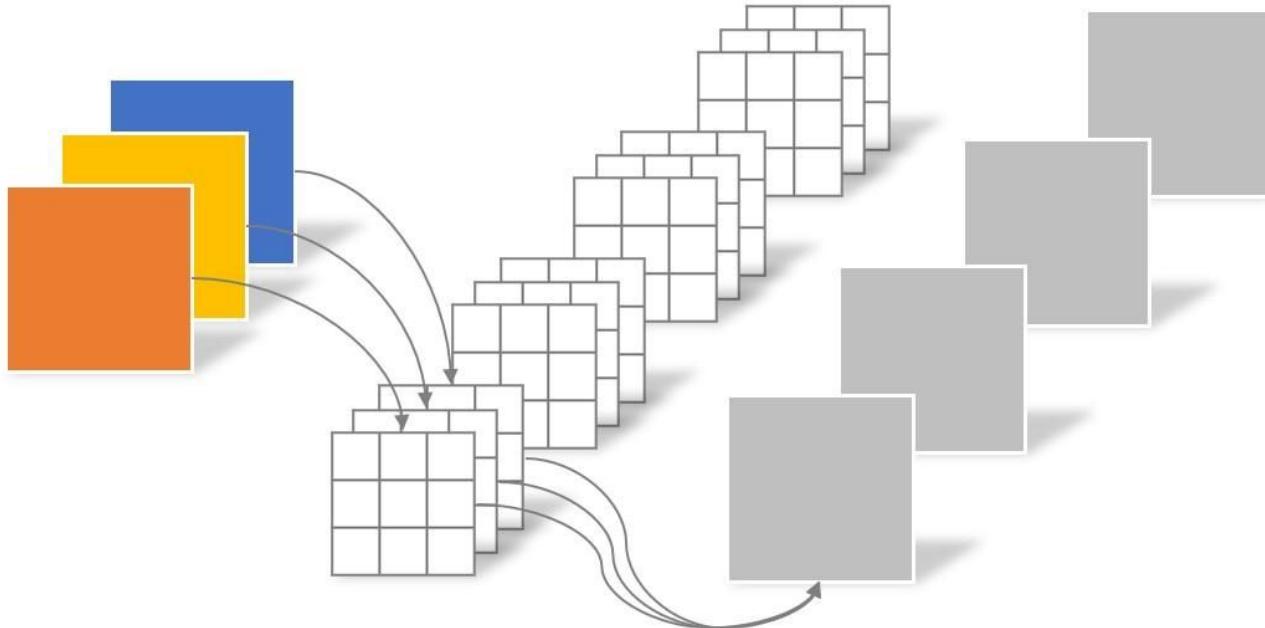
MobileNets: Efficient Convolutional Neural Networks for Mobile Applications

- Depthwise convolutions
(также Xception [2018] целиком базируется на них, ShuffleNet [2018], [Inverted Residual Block](#) [2018])
- Pointwise convolutions



Conventional convolution

3 channel Input



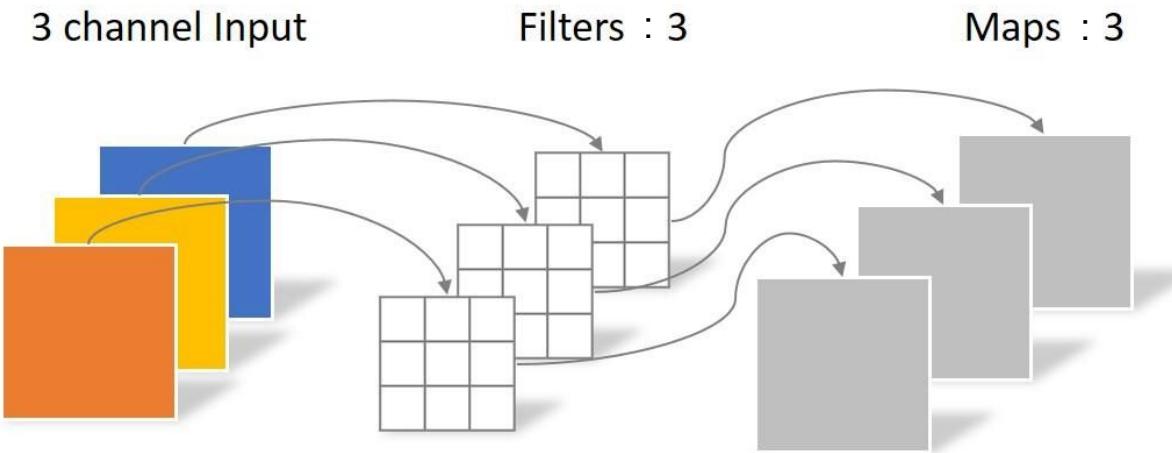
Filters : 4

Maps : 4

Чтобы получить
4х канальную
картинку из 3х
канальной
картинки на
входе

$$N_{std} = 4 \times 3 \times 3 \times 3 = 108;$$

Depthwise convolutions [first step of alternative conv operation]

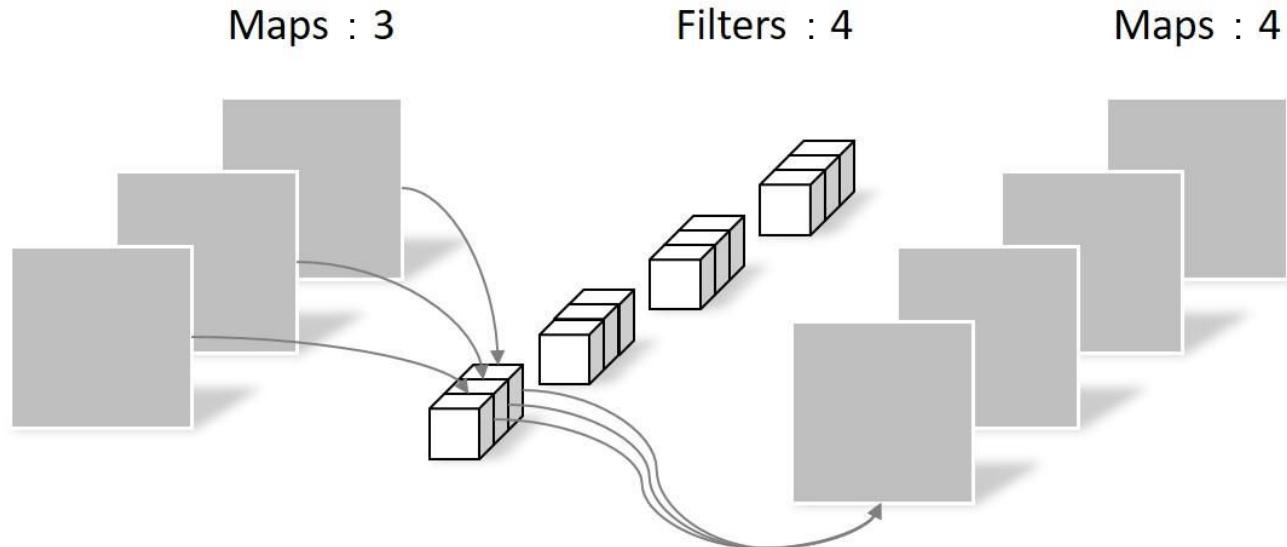


DWC всегда производит количество карт равное количеству карт на входе.

Чтобы сделать фичи нужно применить следующий шаг.

$$N_{\text{depthwise}} = 3 \times 3 \times 3 = 27;$$

Pointwise convolutions



Каждый фильтр в РС имеет размерность $1 \times N$, где N -- количество каналов у исходной картинки (а также количество карт на выходе DWC)

$$N_{\text{pointwise}} = 1 \times 1 \times 3 \times 4 = 12;$$

Итого

$$N_{\text{depthwise}} = 3 \times 3 \times 3 = 27$$

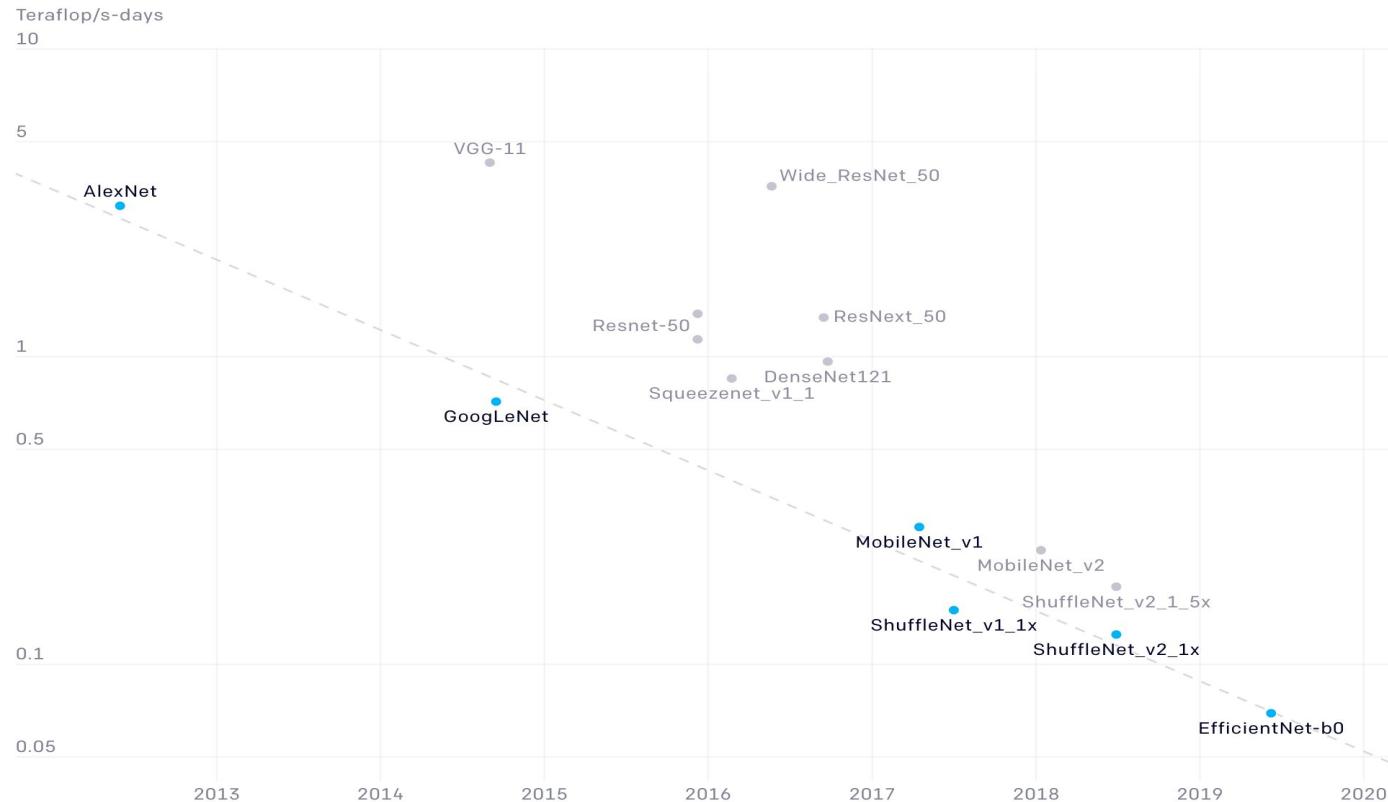
$$N_{\text{pointwise}} = 1 \times 1 \times 3 \times 4 = 12$$

$$N_{\text{separable}} = N_{\text{depthwise}} + N_{\text{pointwise}} = 39$$

Когда у стандартной вышло:

$$N_{\text{std}} = 4 \times 3 \times 3 \times 3 = 108$$

44x less compute required to get to AlexNet performance 7 years later (log scale)



EfficientNet

- Эффективно реализованный поиск архитектуры путём сэмплирования параметров скейла

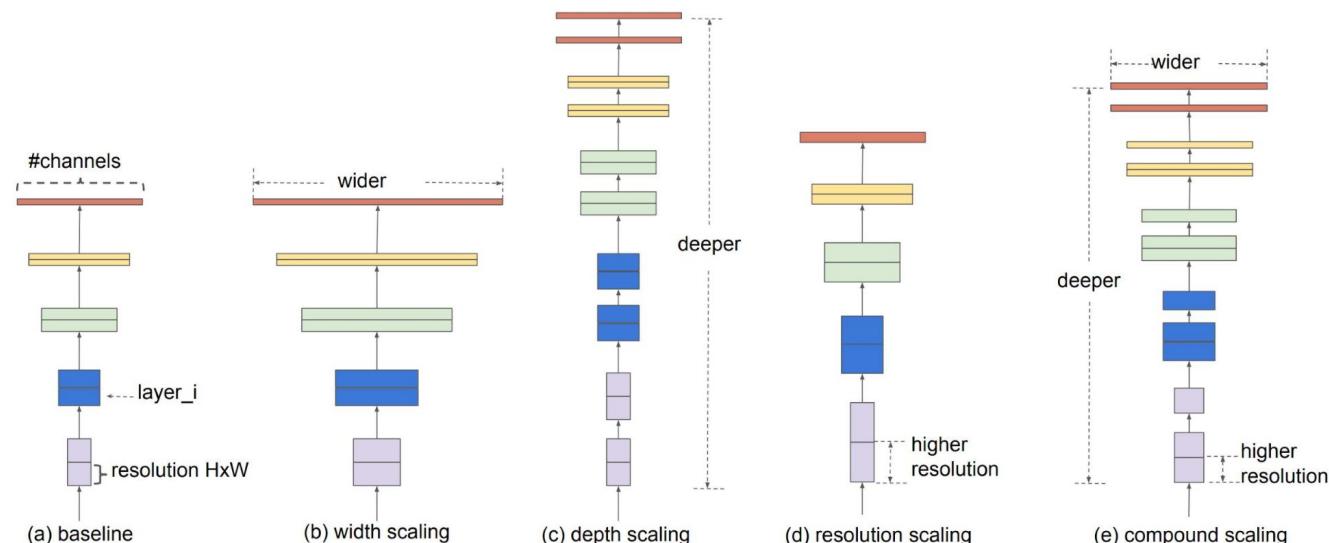
$$\text{depth: } d = \alpha^\phi$$

$$\text{width: } w = \beta^\phi$$

$$\text{resolution: } r = \gamma^\phi$$

$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$



EfficientNet

- Как получили B0 - B7 (смотреть графики в конце)

$$\text{depth: } d = \alpha^\phi$$

$$\text{width: } w = \beta^\phi$$

$$\text{resolution: } r = \gamma^\phi$$

$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

1. Fix $\phi = 1$, assuming that twice more resources are available, and do a small grid search for α , β , and γ . For baseline network **B0**, it turned out the optimal values are $\alpha = 1.2$, $\beta = 1.1$, and $\gamma = 1.15$ such that $\alpha * \beta^2 * \gamma^2 \approx 2$
2. Now fix α , β , and γ as constants (with values found in above step) and experiment with different values of ϕ . The different values of ϕ produce EfficientNets **B1-B7**.

EfficientNet

- Авторы запустили *Neural Architecture Search* чтобы найти бейзлайн архитектуру
- **Inverted Residual Block** (сегодня не смотрели, но смотреть [TUT](#))
 - Tldr; Заменяем в блоке последовательность [много_каналов -> мало_каналов -> много_каналов] на [мало_каналов -> много_каналов -> мало_каналов]; Используем DWC & relu6
- SE смотрели

$$\text{depth: } d = \alpha^\phi$$

$$\text{width: } w = \beta^\phi$$

$$\text{resolution: } r = \gamma^\phi$$

$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

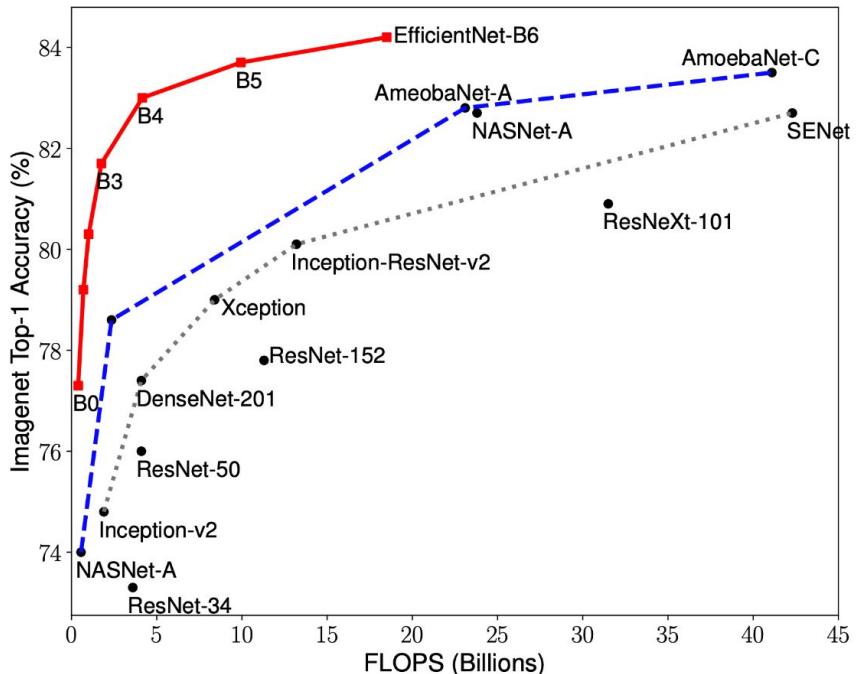
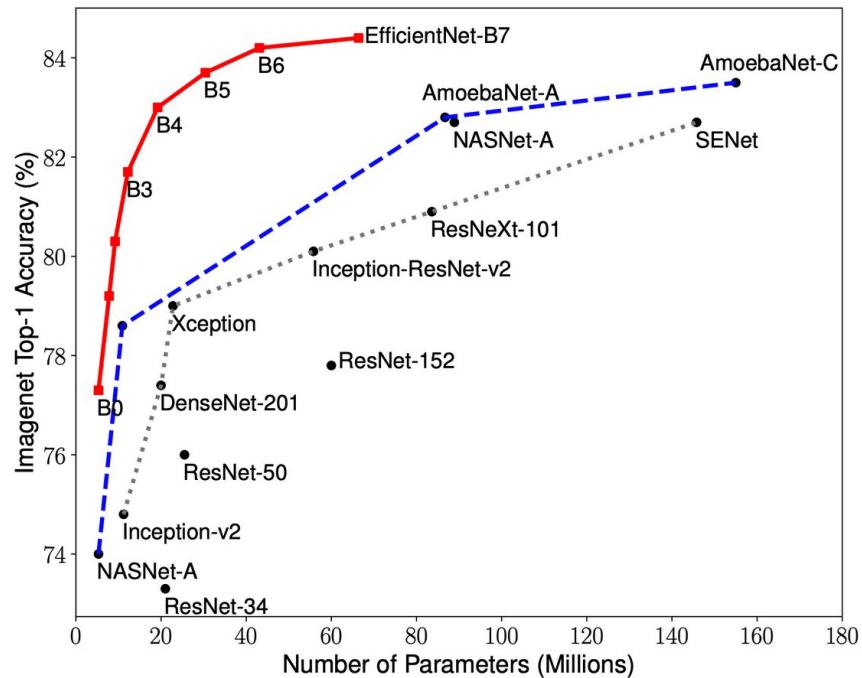
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels \hat{C}_i	#Layers \hat{L}_i
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

EfficientNet-B0 baseline network

The MBConv block is nothing fancy but an Inverted Residual Block (used in MobileNetV2) with a Squeeze and Excite block injected sometimes.

Об эффективности



Подробнее смотрите [пост](#)

Что мы узнали

- AlexNet показала, что возможно использование CNN для обучения моделей в CV.
- VGG показывает, что большие сети работают лучше
- GoogLeNet - один из первых, кто сосредоточился на эффективности, используя свертки 1x1 и Global Average Pooling вместо FC
- ResNet показал нам, как тренировать чрезвычайно глубокие сети, ограниченные только GPU и памятью
- После ResNet: фокус смещается на эффективные сети:
 - Множество крошечных сетей, нацеленных на мобильные устройства: MobileNet, ShuffleNet
 - Neural Architecture Search теперь может автоматизировать проектирование архитектуры

Материалы и ссылки

- При подготовки лекции использовались материалы
 - [Cs231n](#), Stanford 2020, Spring
 - [Ancient secrets of computer vision](#)
 - MobileNet, EfficientNet статьи
- Актуальный часто обновляемый зоопарк CV моделей
 - Pytorch: <https://github.com/rwightman/pytorch-image-models>
- [Интерактивная визуализация работы CNN](#)