

академия  
больших  
данных

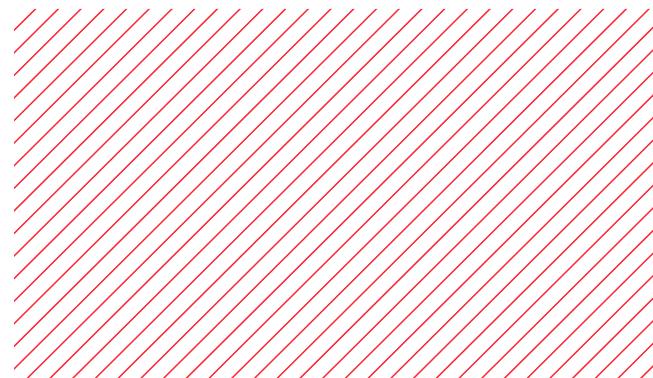


mail.ru  
group

# Архитектуры сверточных нейросетей

Даниил Лысухин

Программист-исследователь в команде машинного  
зрения





# Recap

# Recap: свертка

Входной сигнал  $X_{ij}$

X00	X01	X02	X03
X10	X11	X12	X13
X20	X21	X22	X23
X30	X31	X32	X33

$$Y_{ij} = (X * W)_{ij} = \sum_{u=0}^{K-1} \sum_{v=0}^{K-1} X_{i+u, j+v} W_{uv}$$

Результат свертки  $Y_{ij}$

Y00	Y01
Y10	Y11

Ядро свертки  $W$   
Размер **3x3**  
Веса  $W_{ij}$



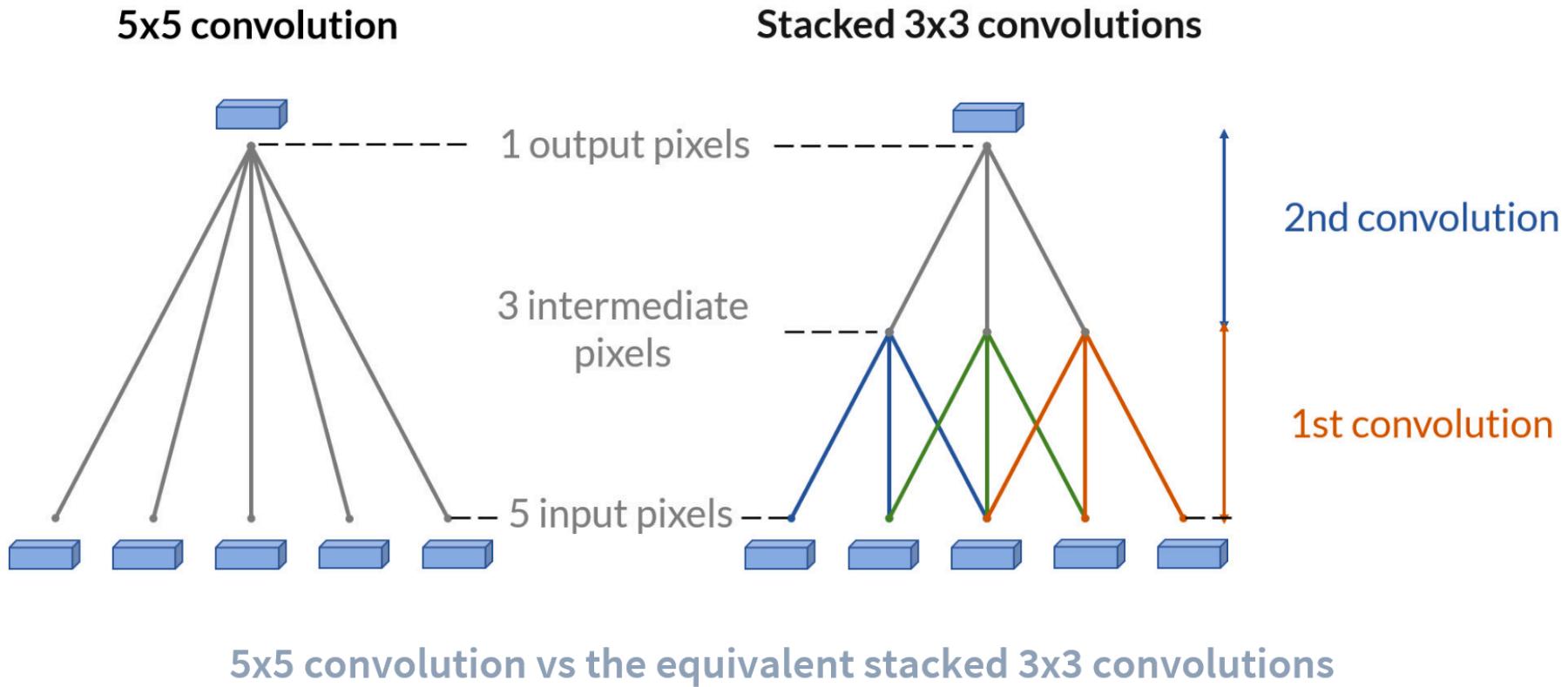
## Recap: свертка

---

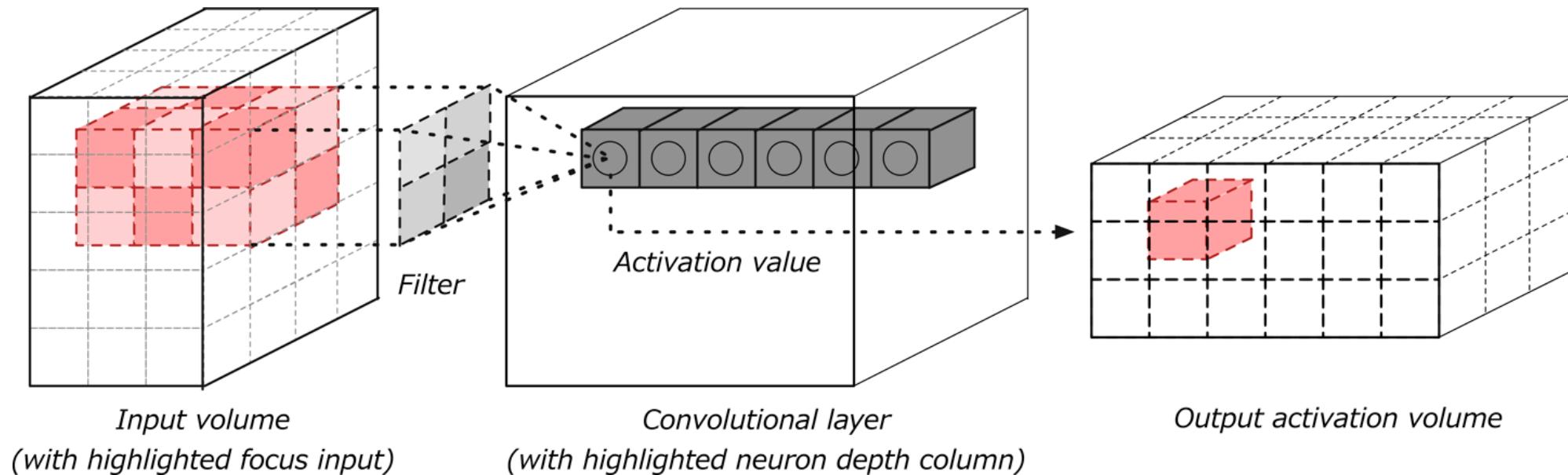
- Чему равно рецептивное поле нейрона после двух сверток  $3 \times 3$ ?

# Recap: свертка

- Чему равно рецептивное поле нейрона после двух сверточ 3 x 3?



# Recap: сверточный слой





# Recap: сверточный слой

---

- Подсчитать число параметров сверточного слоя:
  - Размер ядра  $3 \times 3$
  - Глубина входного тензора 64
  - Глубина выходного тензора 128

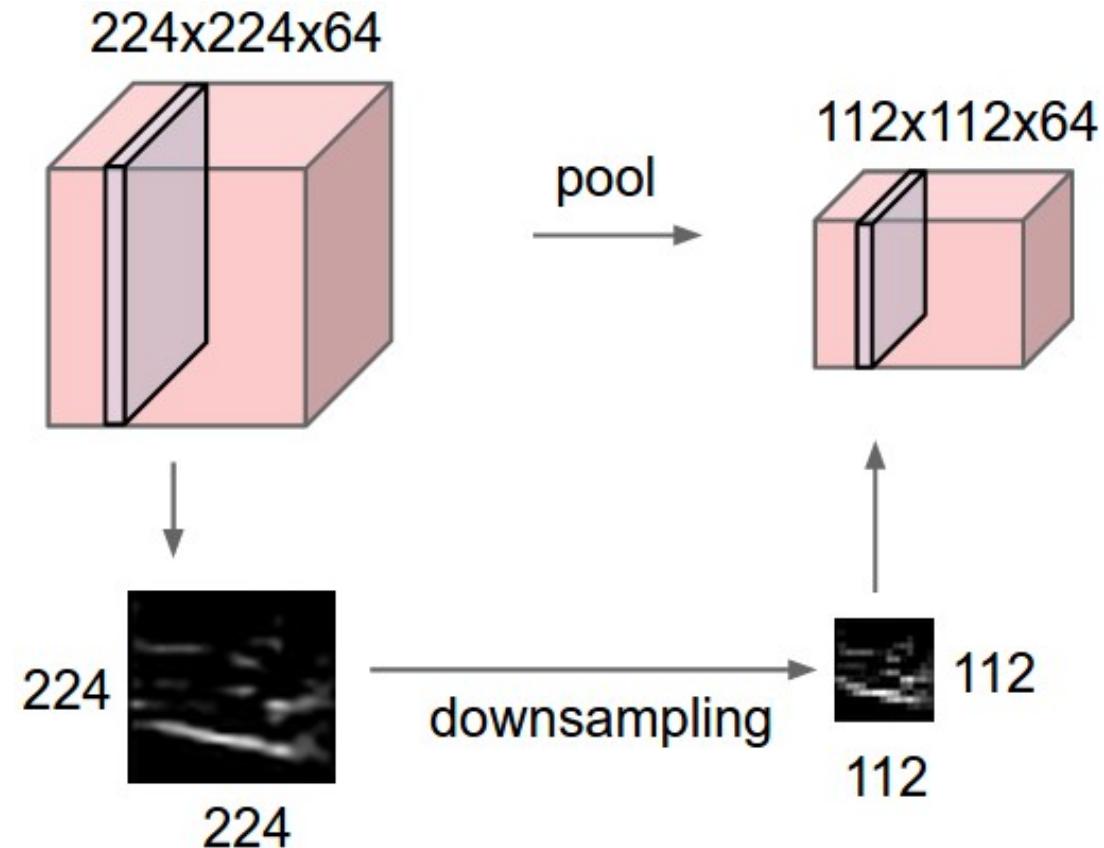


# Recap: сверточный слой

---

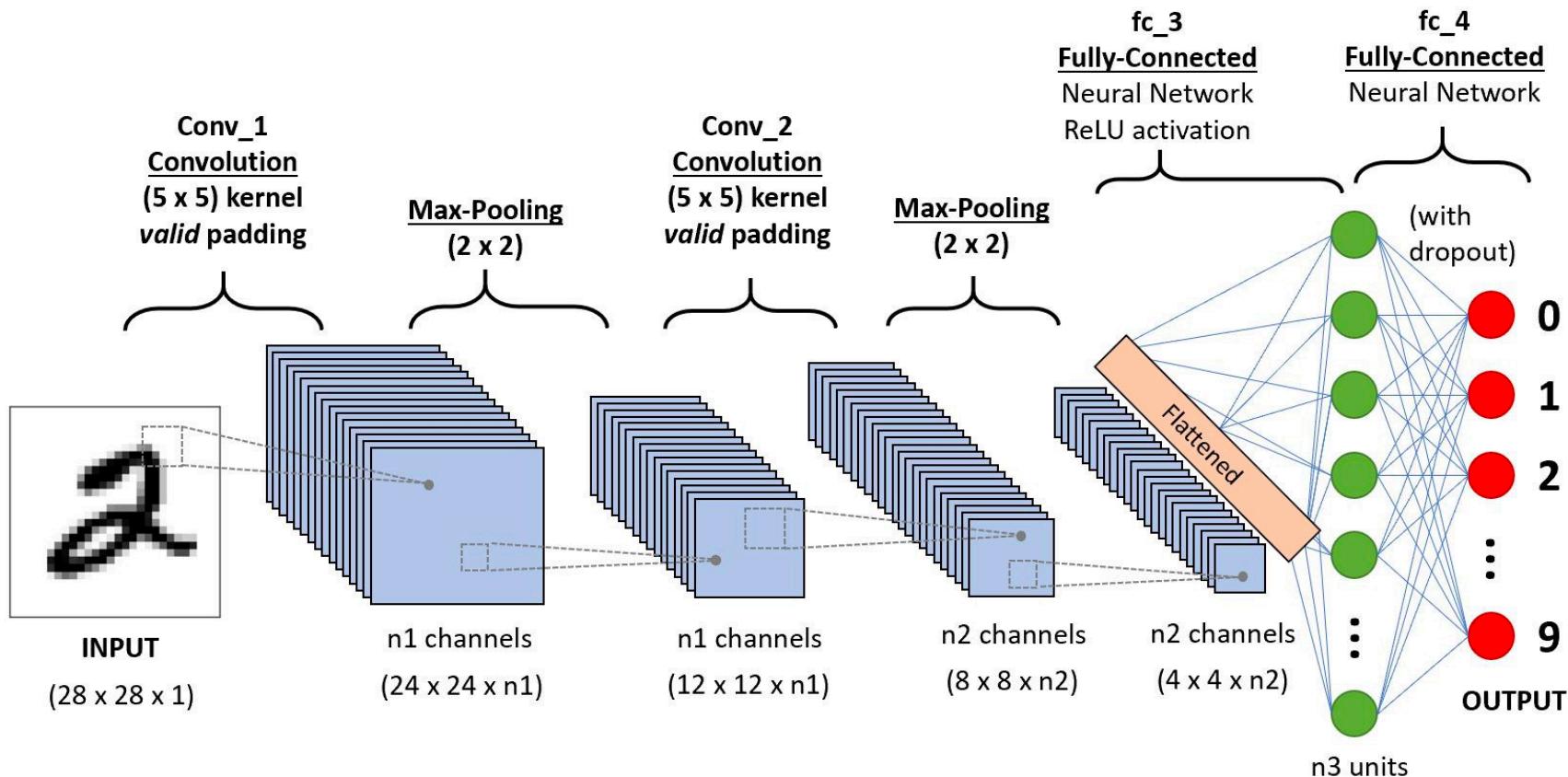
- Подсчитать число параметров сверточного слоя:
  - Размер ядра  $3 \times 3$
  - Глубина входного тензора 64
  - Глубина выходного тензора 128
- Число весов для 1 ядра =  $3 \times 3 \times 64 + 1$  (bias) = 577
- Для 512 ядер будет  $577 \times 128 = 73\,856$

# Recap: пулинг



# Recap: CNN

---



# ImageNet





# ImageNet

---

- База изображений
- Организована на основе иерархии WordNet
- > 15M изображений
- > 20k категорий
- Есть разметка положения объектов (bounding box localization)



mite



container ship



motor scooter



leopard

mite
black widow
cockroach
tick
starfish

container ship
lifeboat
amphibian
fireboat
drilling platform

motor scooter
go-kart
moped
bumper car
golfcart

leopard
jaguar
cheetah
snow leopard
Egyptian cat



grille



mushroom



cherry



Madagascar cat

convertible
grille
pickup
beach wagon
fire engine

agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers

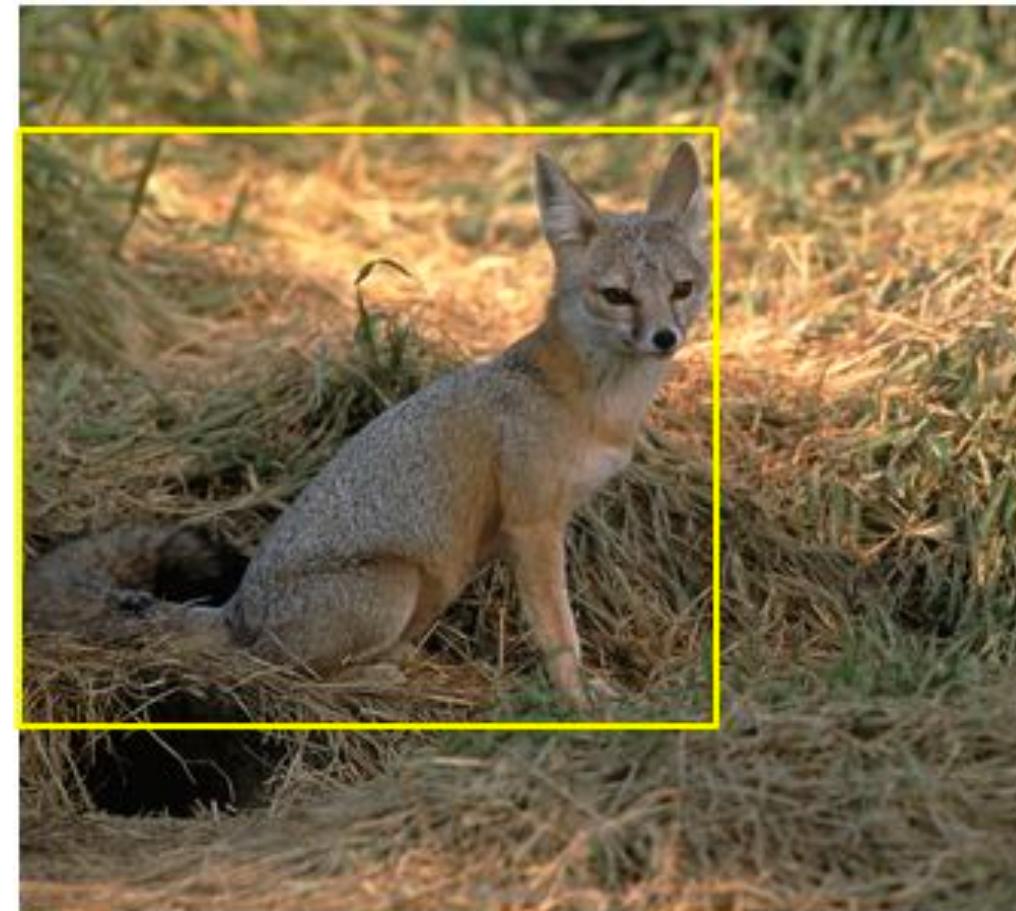
dalmatian
grape
elderberry
ffordshire bullterrier
currant

squirrel monkey
spider monkey
titi
indri
howler monkey

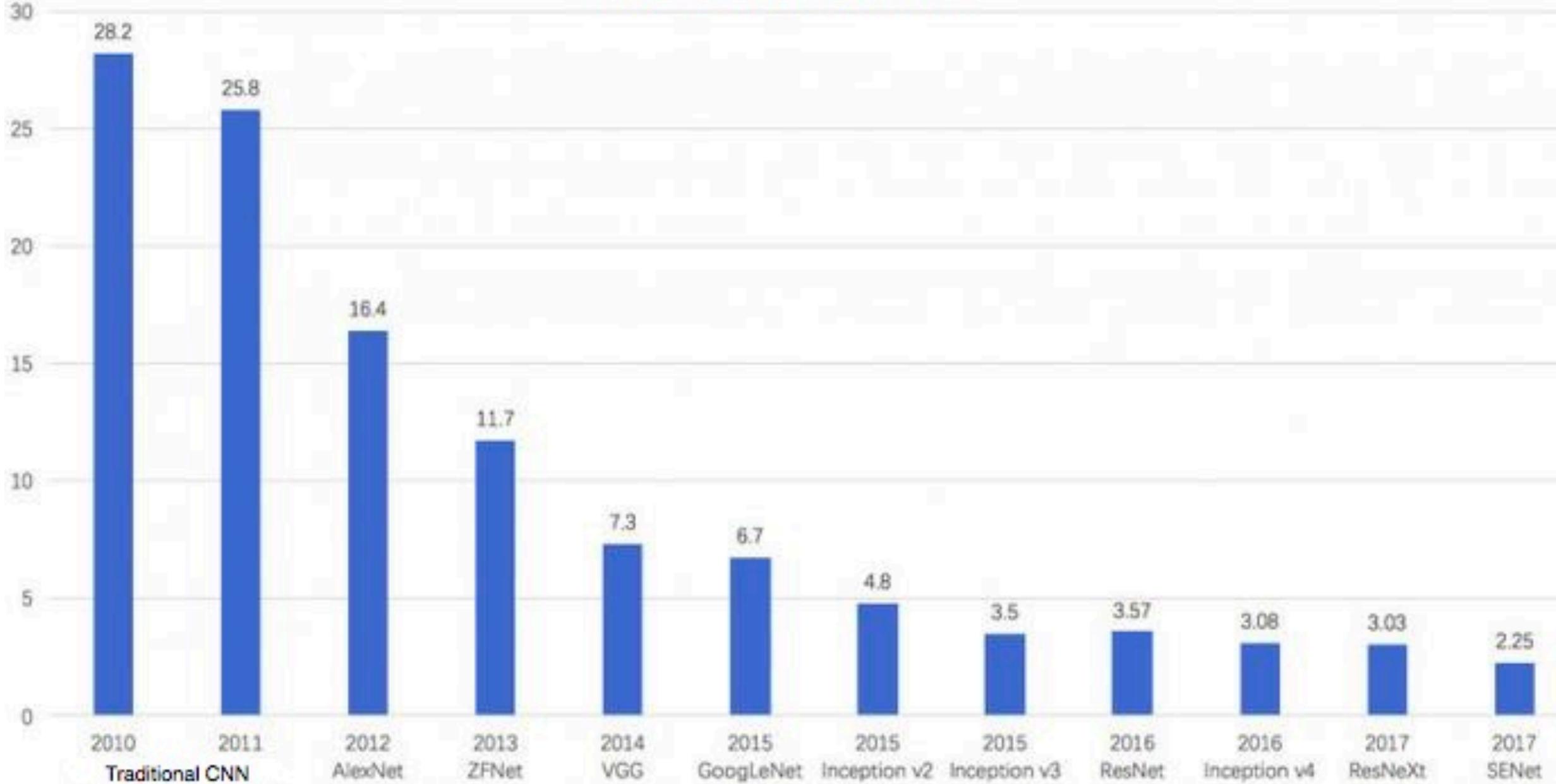
# ILSVRC

---

- ImageNet Large-Scale Visual Recognition Challenge
  - Соревнование на основе ImageNet
  - Подзадачи:
    - Классификация на 1000 классов
    - Локализация



## ImageNet Top 5 Error Rate



# Эволюция DL



# LeNet-5 (1998)

---

- <http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf>
- Задача - распознавание рукописных цифр (MNIST)
- **60k** параметров
- Активация: **tanh**
- RBF-слой в конце



# LeNet-5 (1998)

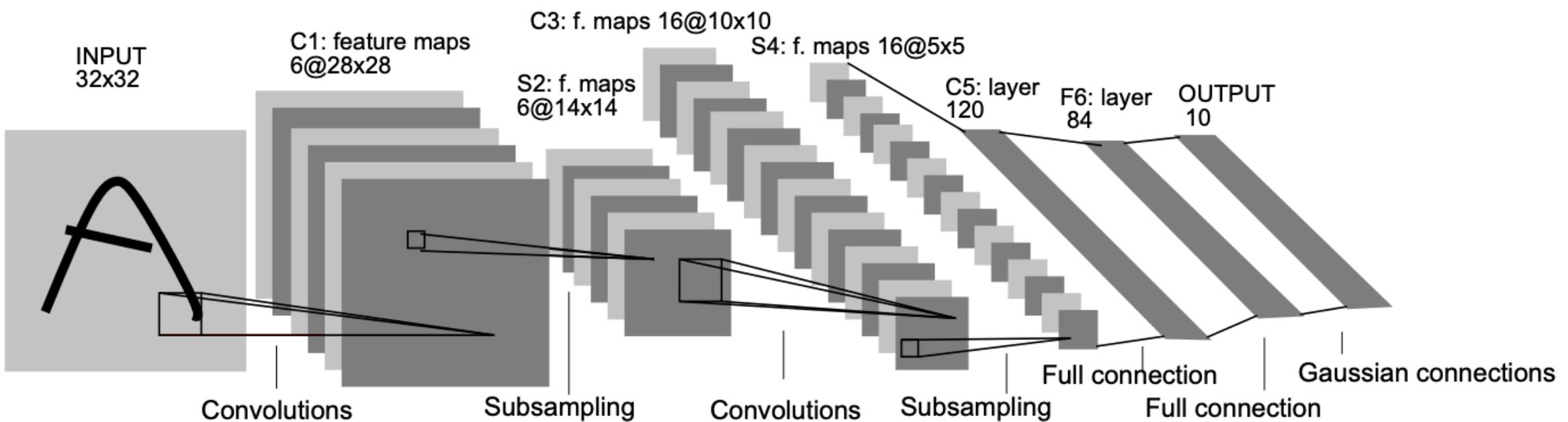


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

# LeNet-5 (1998)

- Нейроны сверточного слоя C3 “подключены” не ко всем выходам предыдущего слоя S2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X			X	X	X			X	X	X	X		X	X	
1	X	X			X	X	X			X	X	X	X		X	
2	X	X	X			X	X	X			X		X	X	X	
3		X	X	X		X	X	X	X			X		X	X	
4			X	X	X		X	X	X	X		X	X		X	
5				X	X	X		X	X	X	X		X	X	X	

TABLE I

EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED  
BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

# LeNet-5 (1998)

---

- Последний слой (RBF) вычисляет MSE между картой активаций и векторами символов

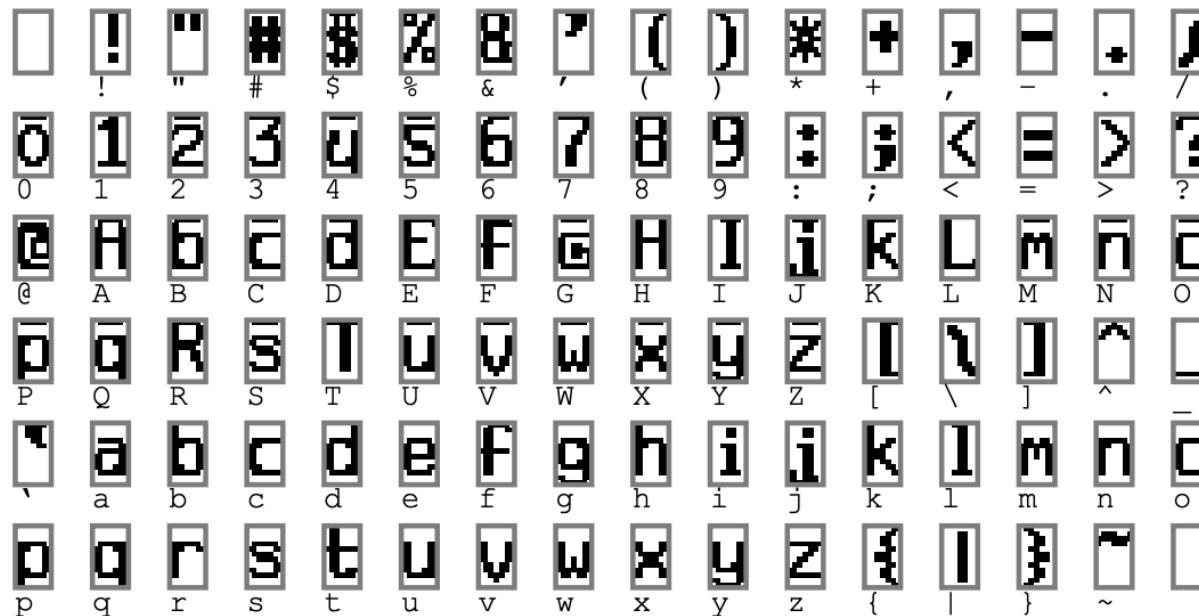
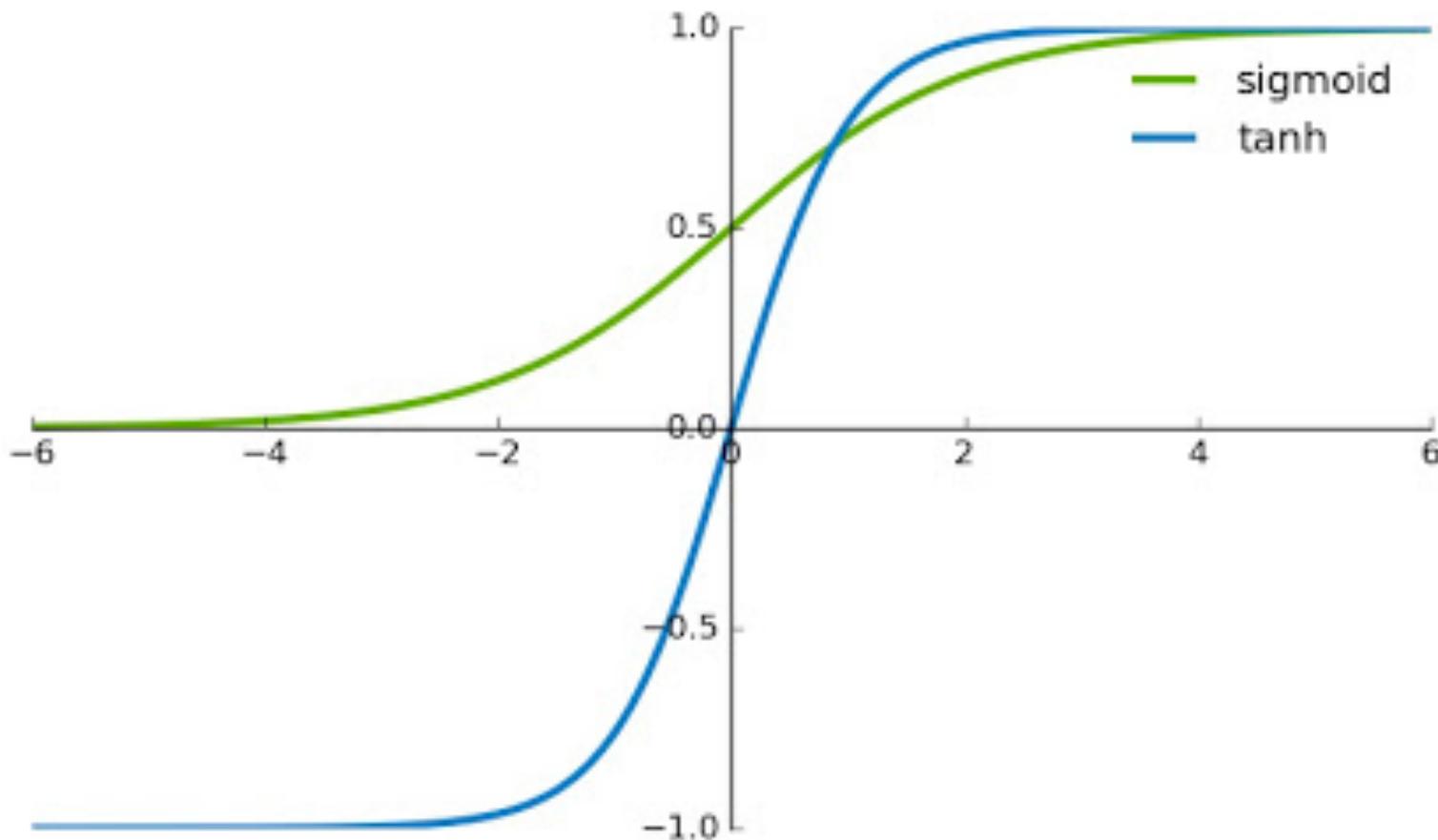


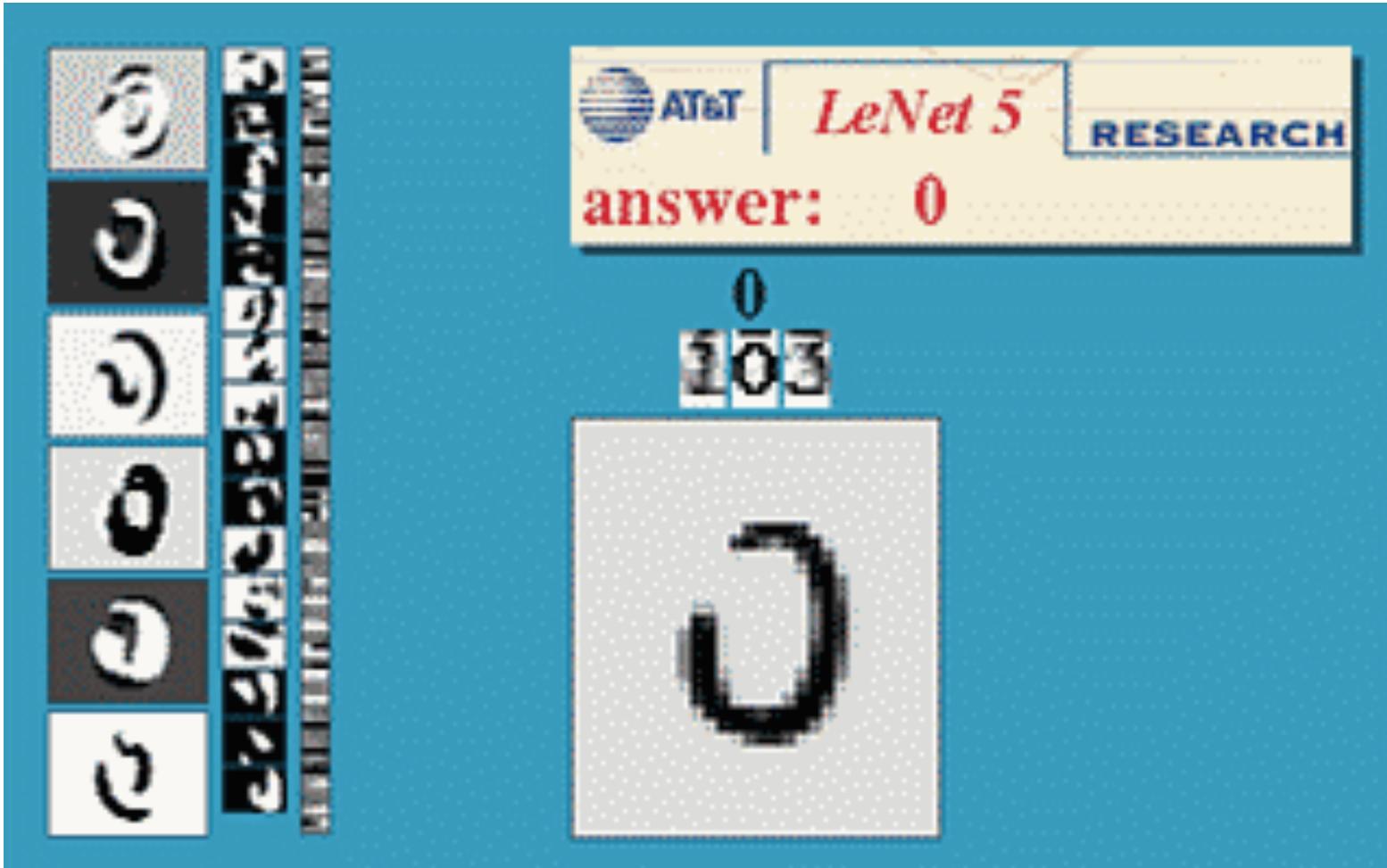
Fig. 3. Initial parameters of the output RBFs for recognizing the full ASCII set.

# LeNet-5 (1998)



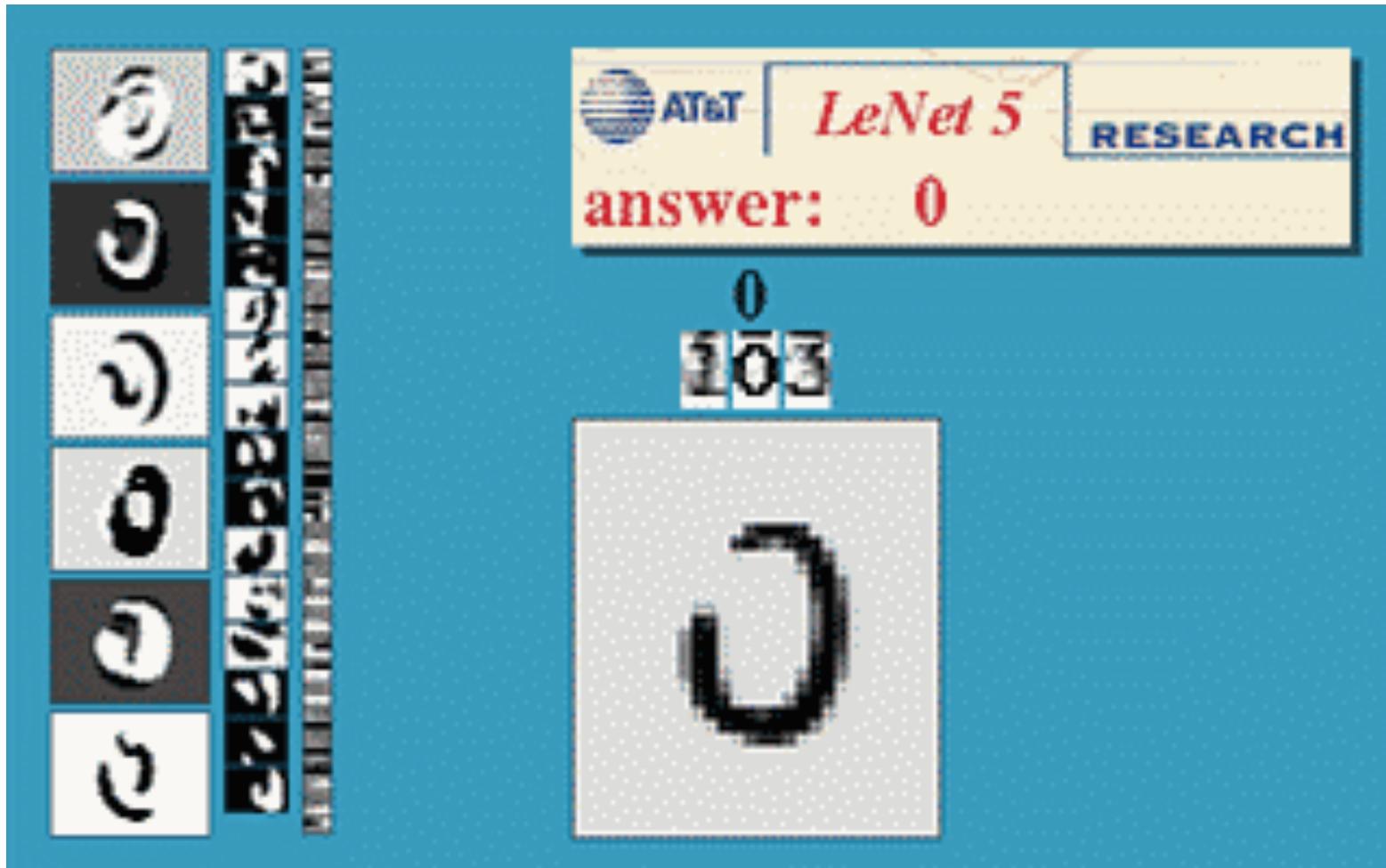
# LeNet-5 (1998)

---



# LeNet-5 (1998)

---



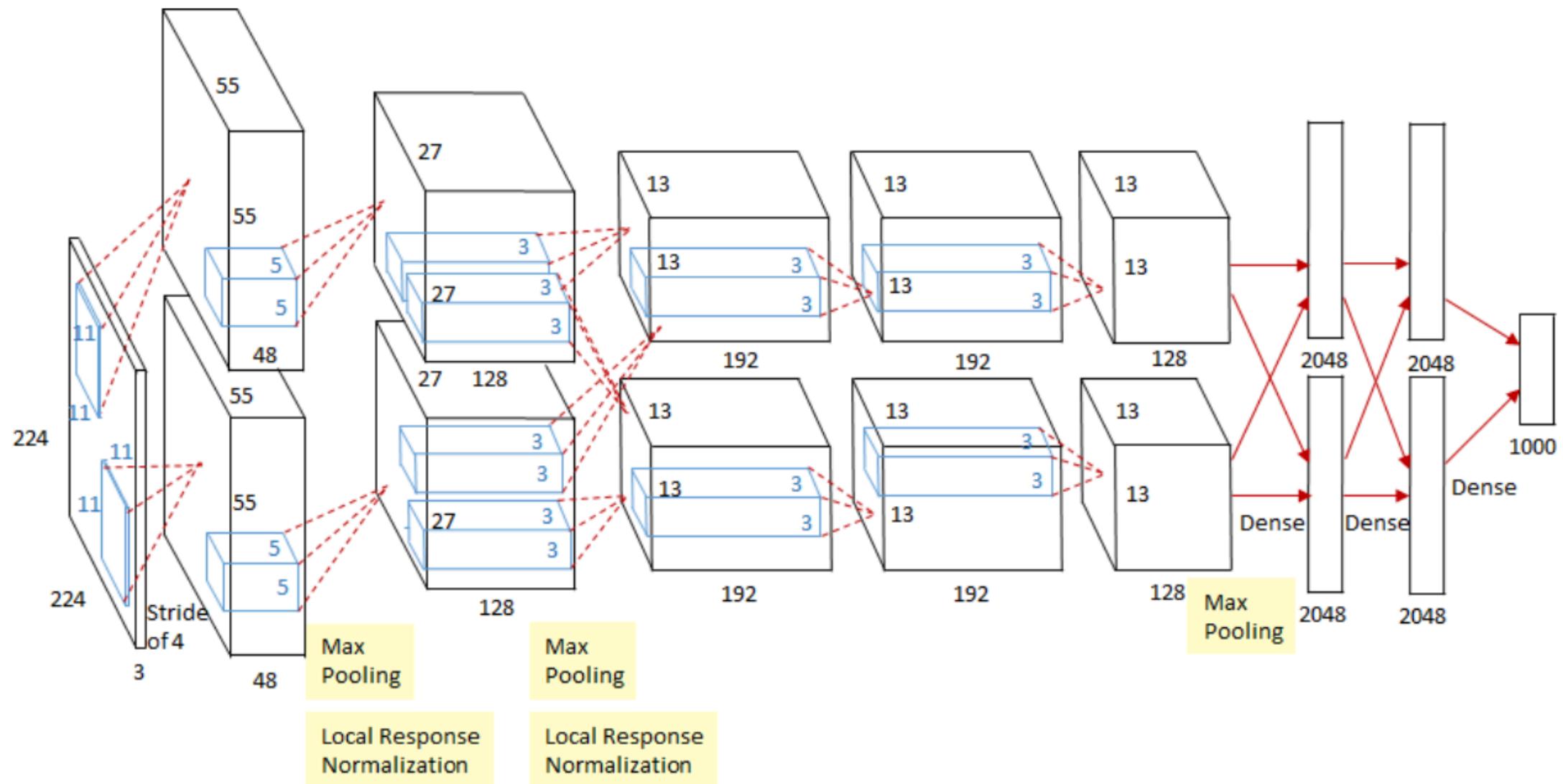


# AlexNet (2012)

---

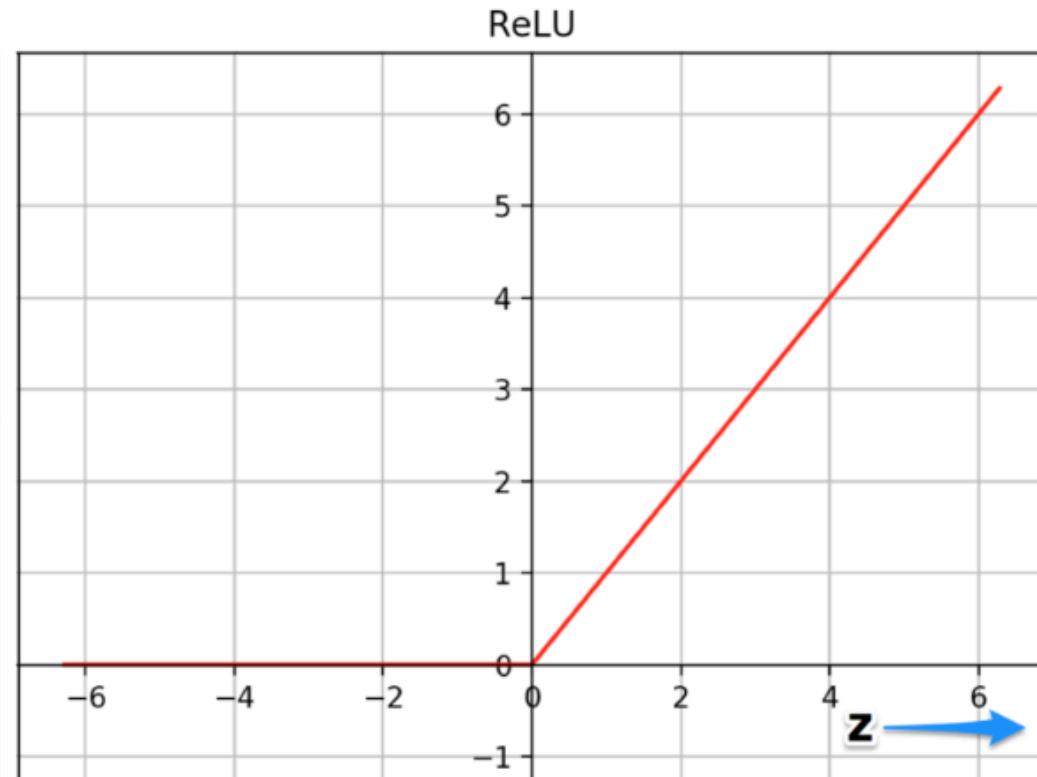
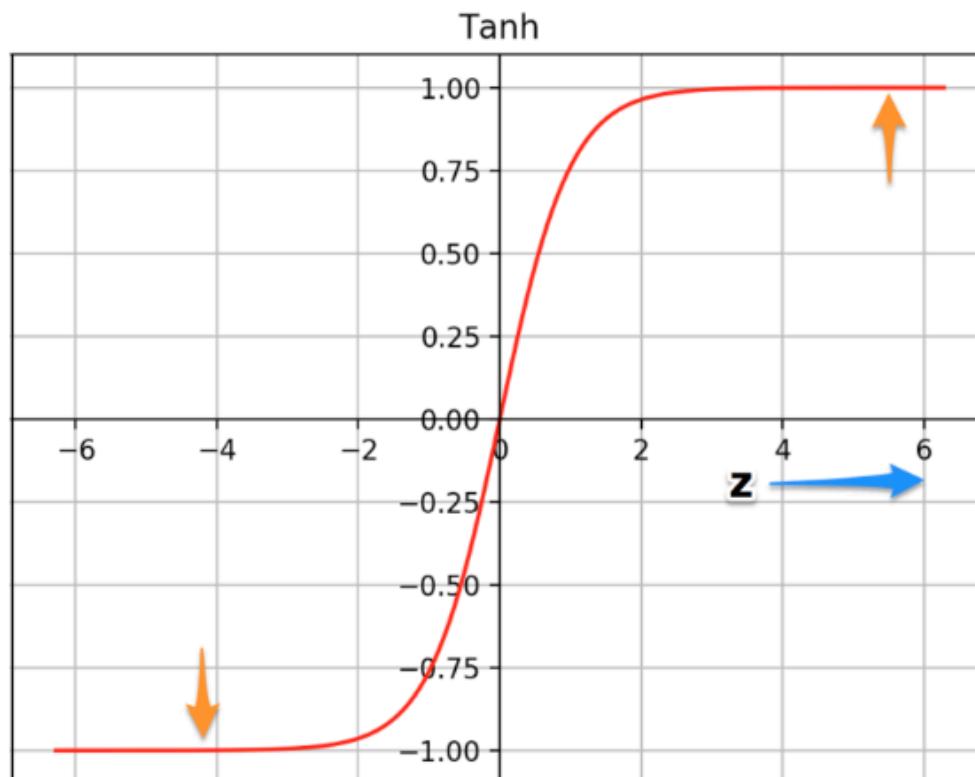
- <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
- Первая CNN-модель, победившая в ILSVRC (2012)
- **60М** параметров
- Размеры сверток:  $3 \times 3$ ,  $5 \times 5$ ,  $11 \times 11$
- Активация: **ReLU**
- Обучение на **2 GPU**
- Использование **аугментаций** для обучения и предсказания

# AlexNet (2012)



# AlexNet (2012)

- Использовали ReLU для улучшения сходимости





# AlexNet (2012)

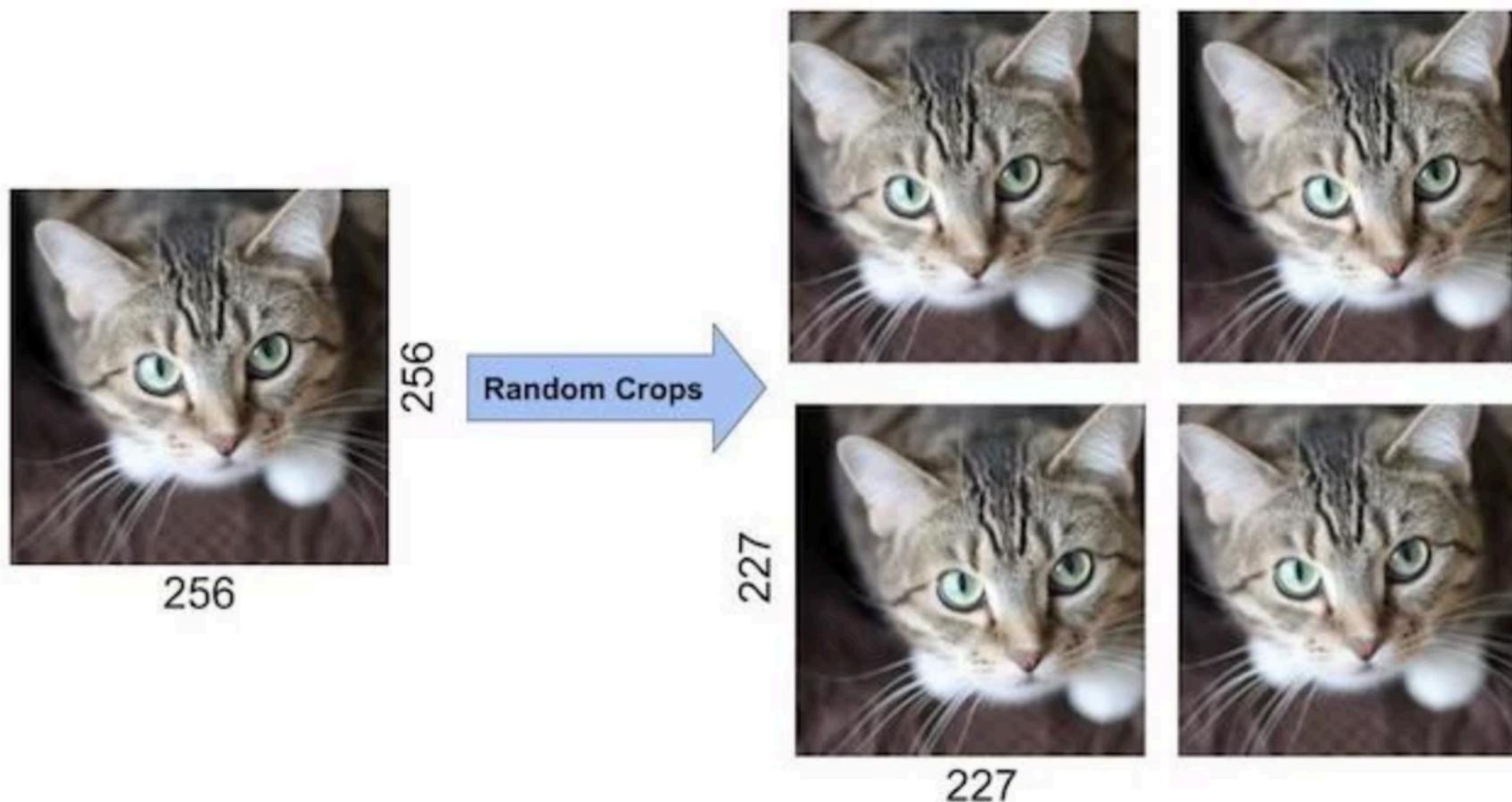
---

- Исходные картинки 256 x 256 px (центральные кропы изображений из ImageNet)
- Обучение:
  - Случайные кропы 227 x 227
  - Использование отраженных примеров

# AlexNet (2012)

---

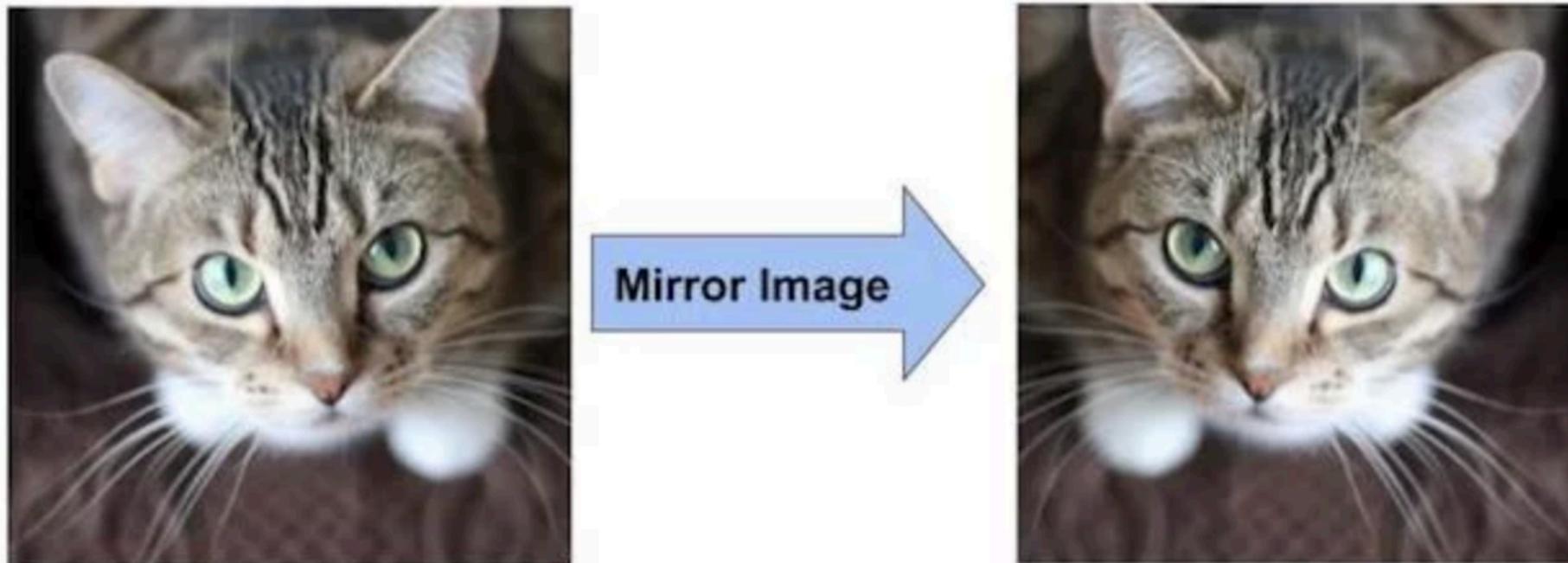
- Случайные кропы 227 x 227



# AlexNet (2012)

---

- Использование отраженных примеров





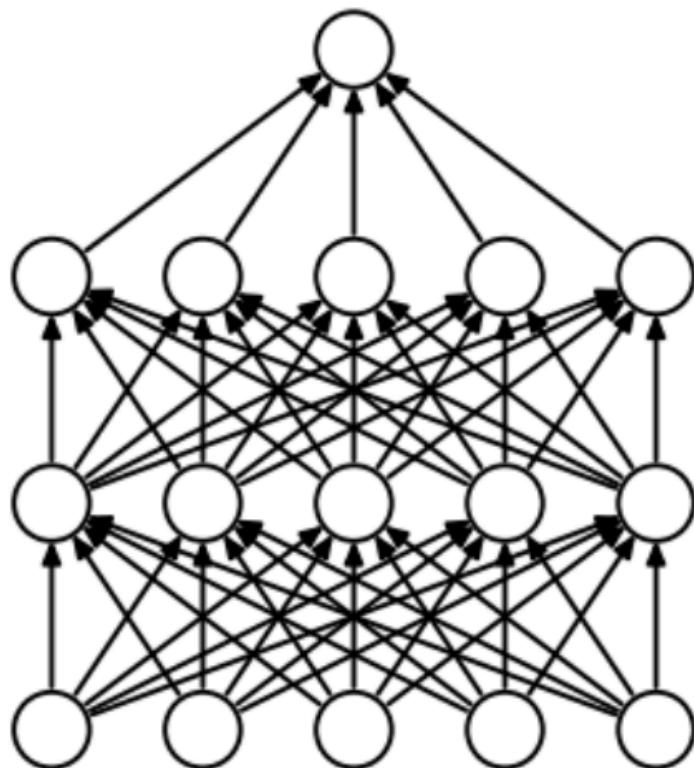
# AlexNet (2012)

---

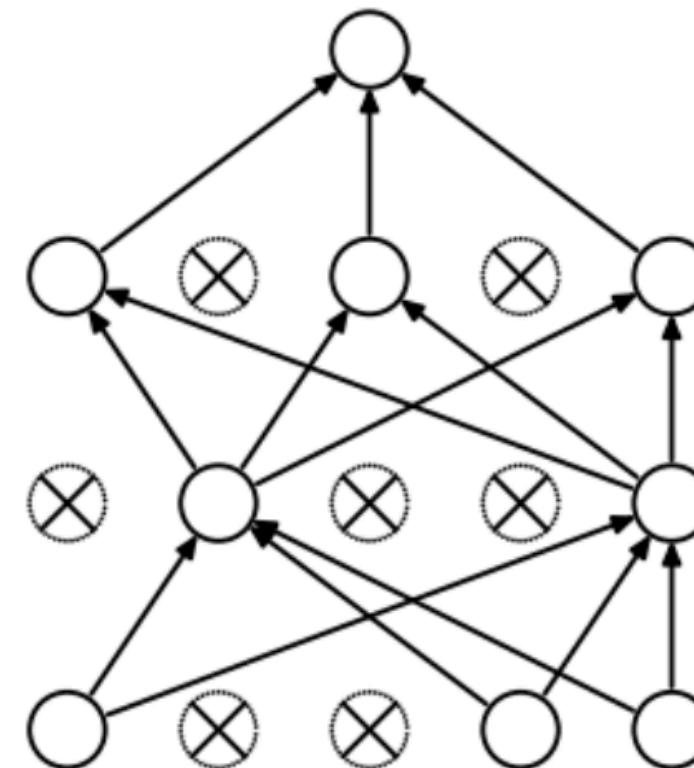
- **Test-time augmentation:** на этапе предсказания усредняют результаты по 10 примерам:
  - 5 кропов (4 угловых + 1 центральный)
  - $\times 2$  с учетом отраженных

# AlexNet (2012)

- Использование Dropout (для FC-слоев):



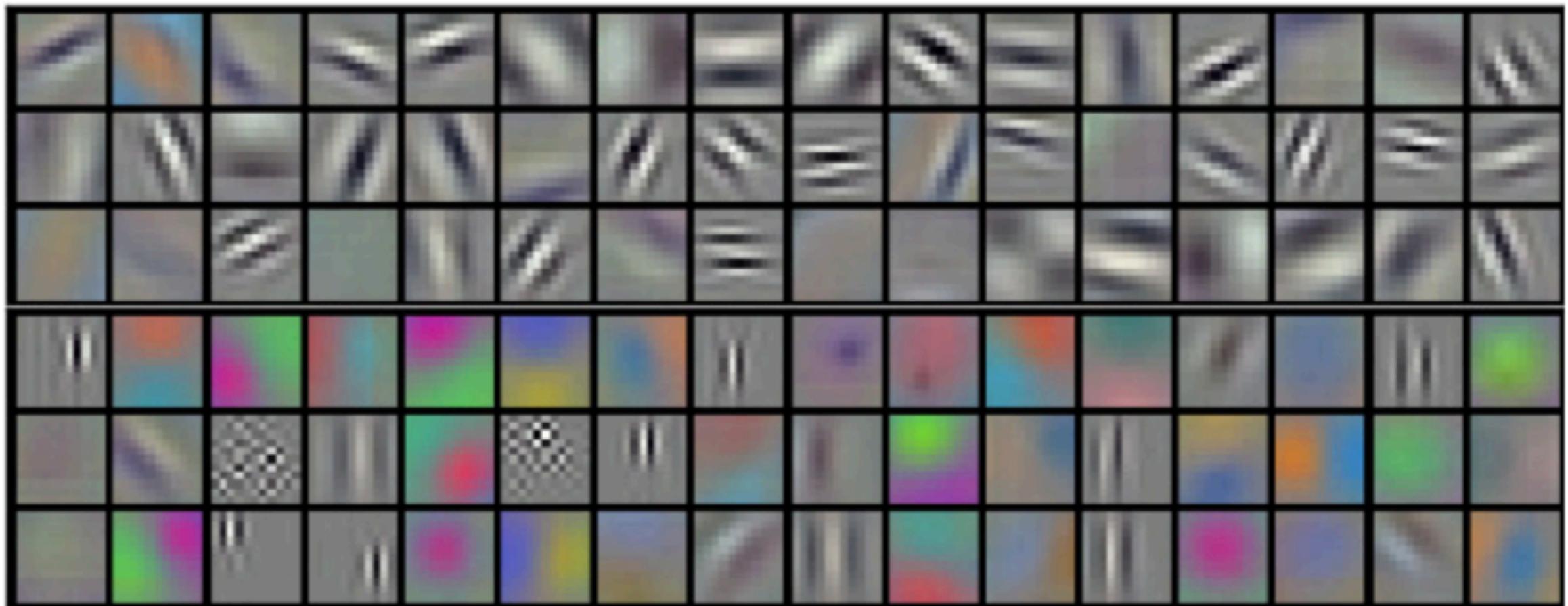
(a) Standard Neural Net



(b) After applying dropout.

# AlexNet (2012)

---



*Learnings of First convolution layer on image of size 224X224X3*



# VGG (2014)

---

- <https://arxiv.org/abs/1409.1556>
- Свертки **3 x 3** (и  $1 \times 1$  в отдельных вариациях)
- **138M** параметров (VGG-19)

# VGG (2014)

- Обучили несколько вариантов архитектур разной глубины
  - VGG-11, VGG-16, VGG-19, ...

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 <b>conv3-64</b>	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 <b>conv1-256</b>	conv3-256 conv3-256 <b>conv3-256</b>	conv3-256 conv3-256 conv3-256 <b>conv3-256</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

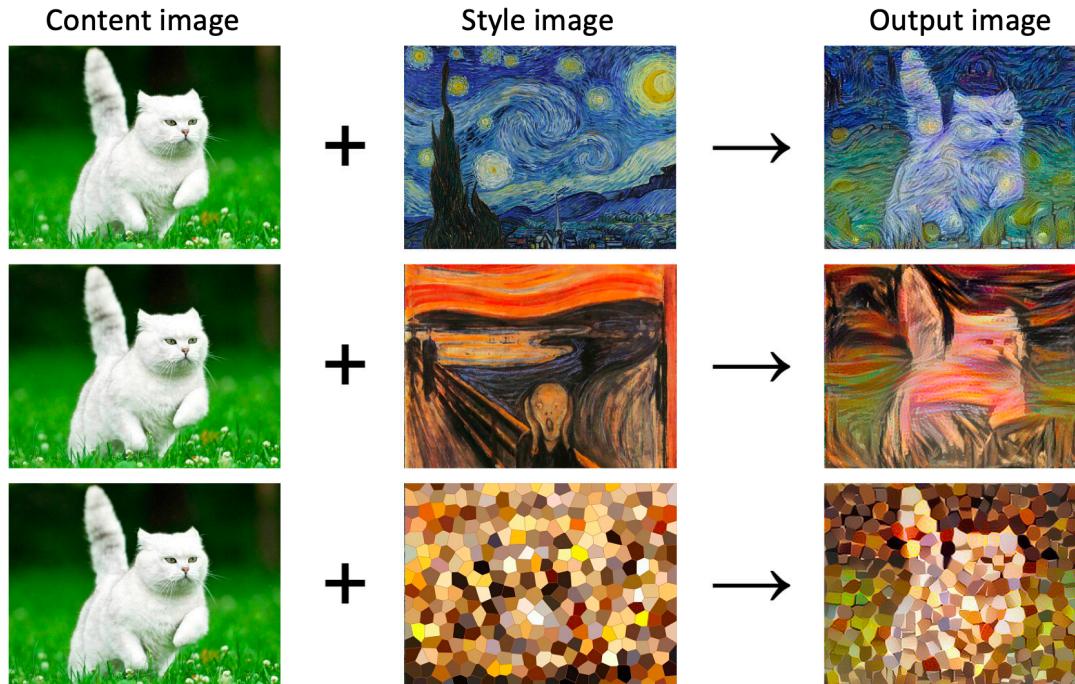
# VGG (2014)

- # весов:  $(3 \times 3 \times 512) \times 512 = 2\ 359\ 296$
- # весов:  $(7 \times 7 \times 512) \times 4096 = 102\ 760\ 448$

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 <b>conv3-64</b>	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 <b>conv1-256</b>	conv3-256 conv3-256 <b>conv3-256</b>	conv3-256 conv3-256 conv3-256 <b>conv3-256</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

# VGG (2014)

- Предобученные (на ImageNet) VGG стали использовать для извлечения признаков (feature extraction) в других задачах, например Style Transfer (будет на следующих лекциях)





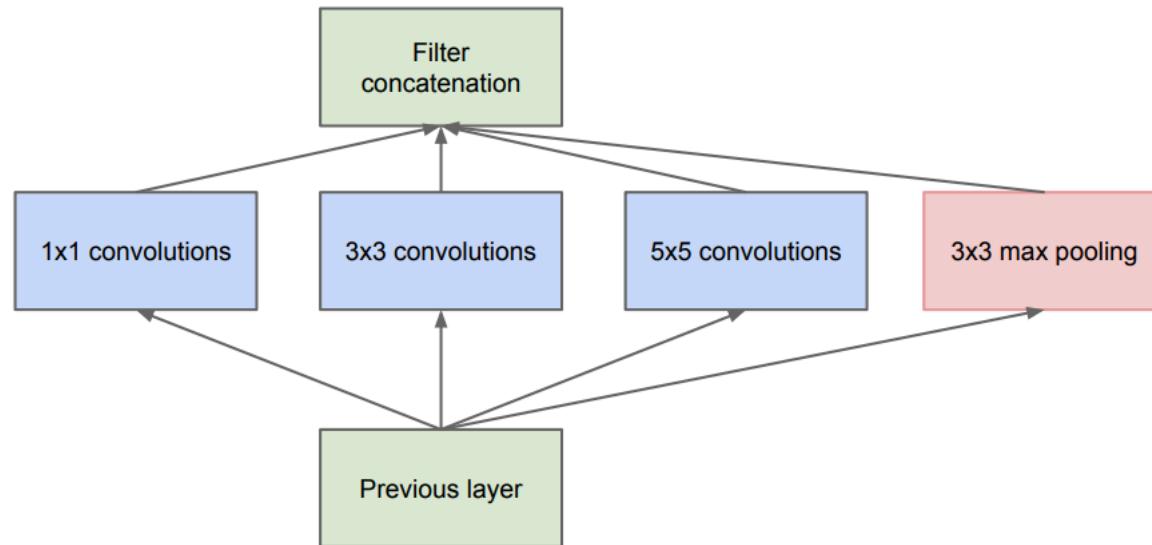
# Inception / GoogLeNet (2014)

---

- <https://arxiv.org/abs/1409.4842>
- Новая единица - **inception-блок**
- Использование сверток разного размера для учета **разного масштаба признаков**
- Пара **дополнительных “голов”** для обучения
- **Global Average Pooling (GAP)** перед FC-классификатором
- **5M (!) параметров**

# Inception / GoogLeNet (2014)

- Идея: использовать “параллельные” каналы для вычисления признаков

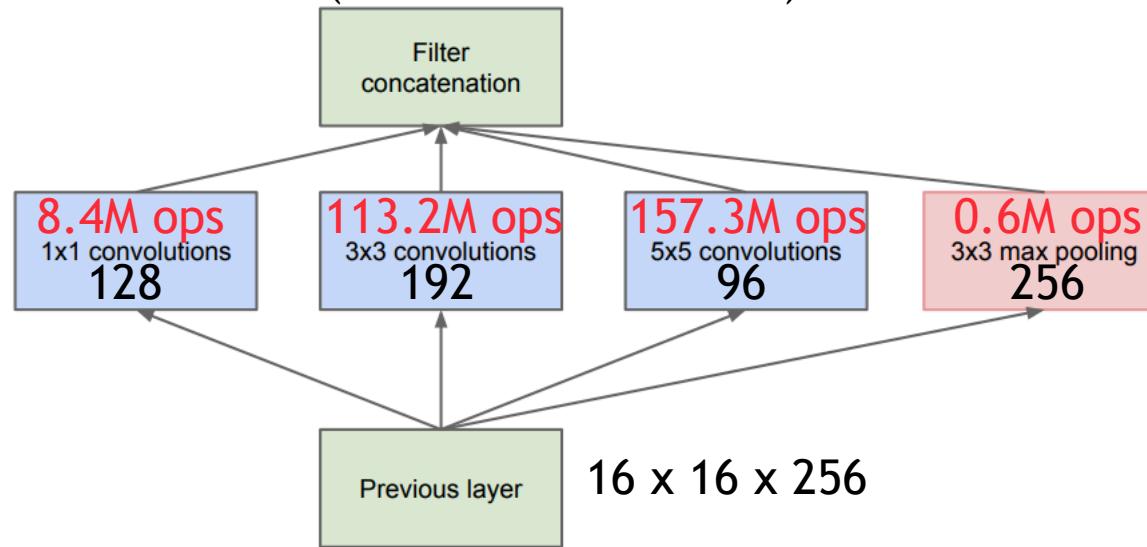


(a) Inception module, naïve version

# Inception / GoogLeNet (2014)

- Идея: использовать “параллельные” каналы для вычисления признаков

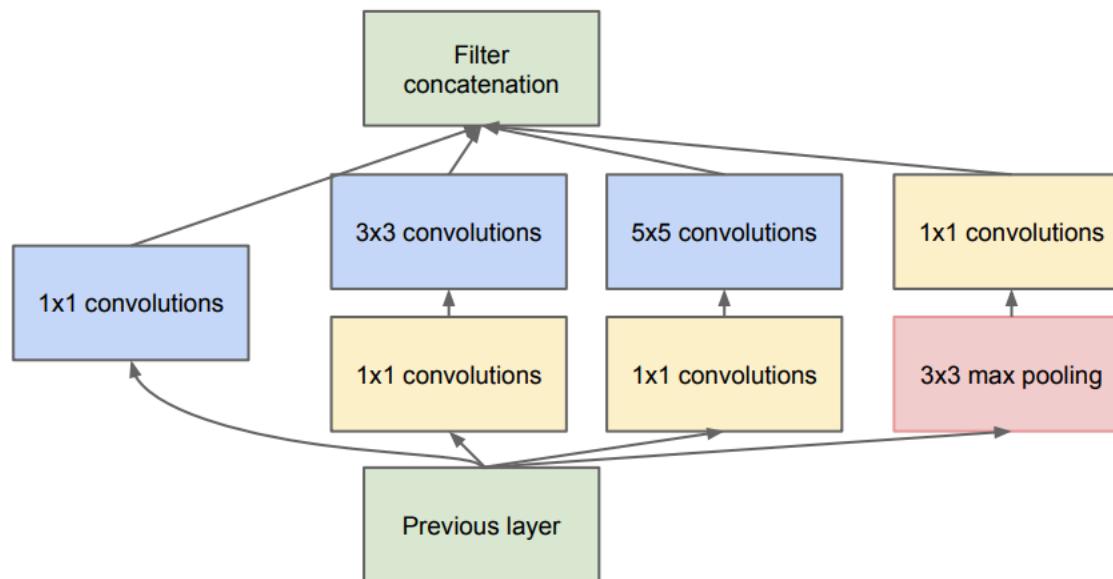
$$16 \times 16 \times (128 + 192 + 96 + 256) = 16 \times 16 \times 672 \quad 280M \text{ ops}$$



(a) Inception module, naïve version

# Inception / GoogLeNet (2014)

- Идея: понижать размерность с помощью сверток  $1 \times 1$

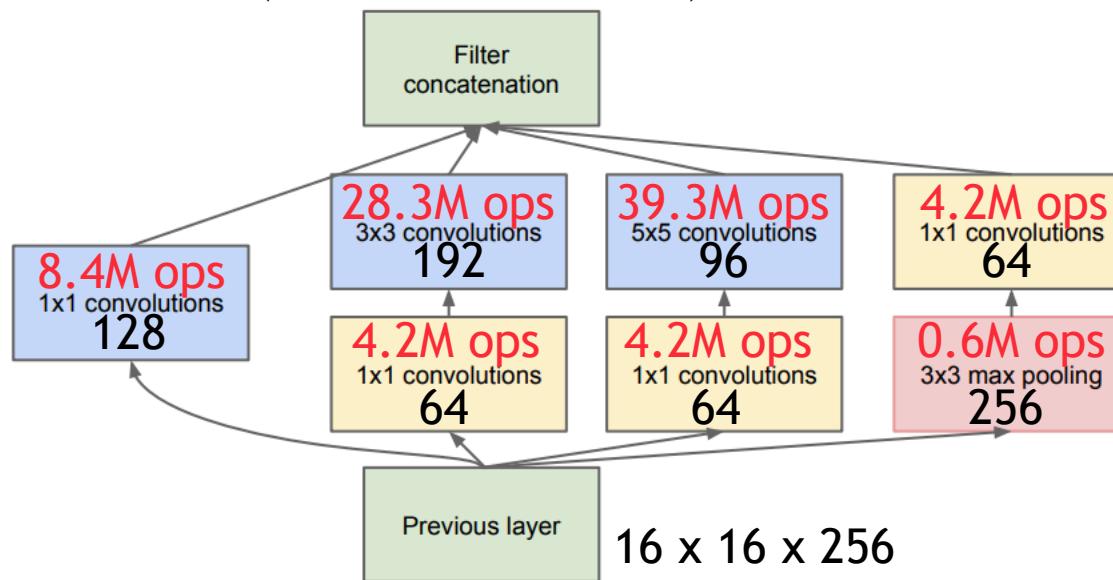


(b) Inception module with dimension reductions

# Inception / GoogLeNet (2014)

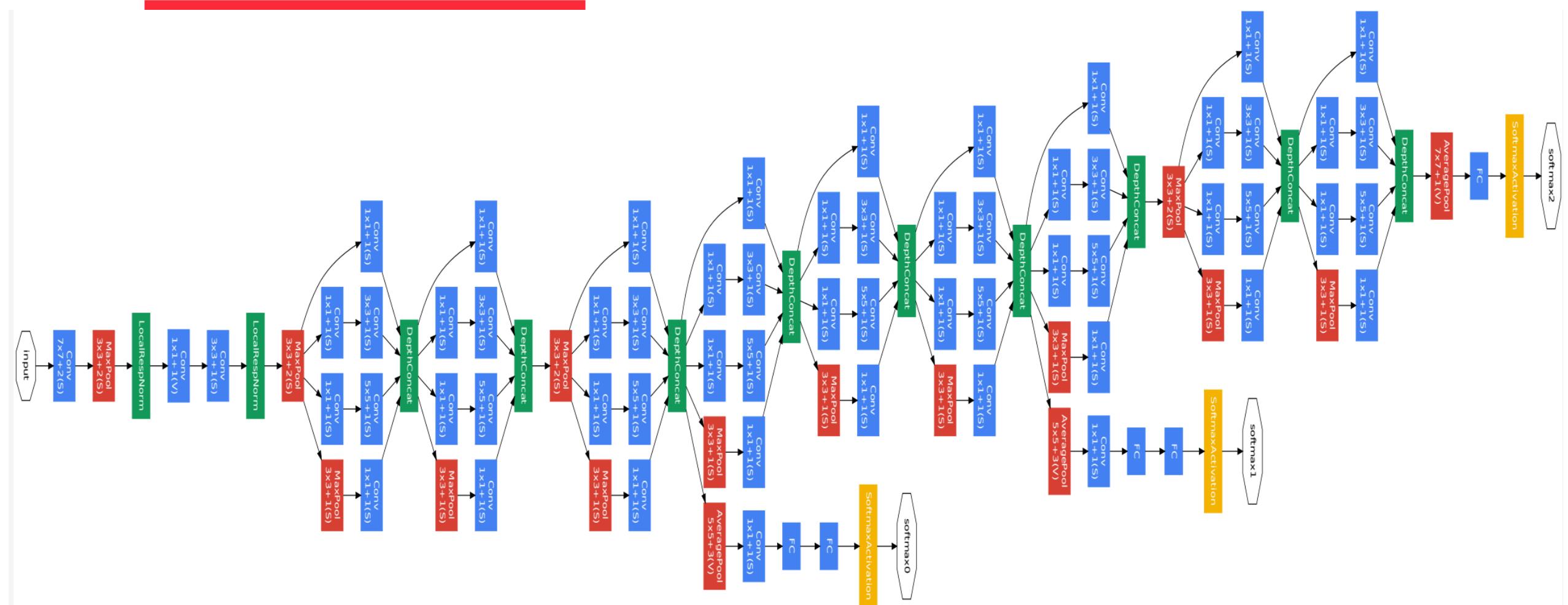
- Идея: понижать размерность с помощью сверток  $1 \times 1$

$$16 \times 16 \times (128 + 192 + 96 + 64) = 16 \times 16 \times 480 \quad 90M \text{ ops}$$

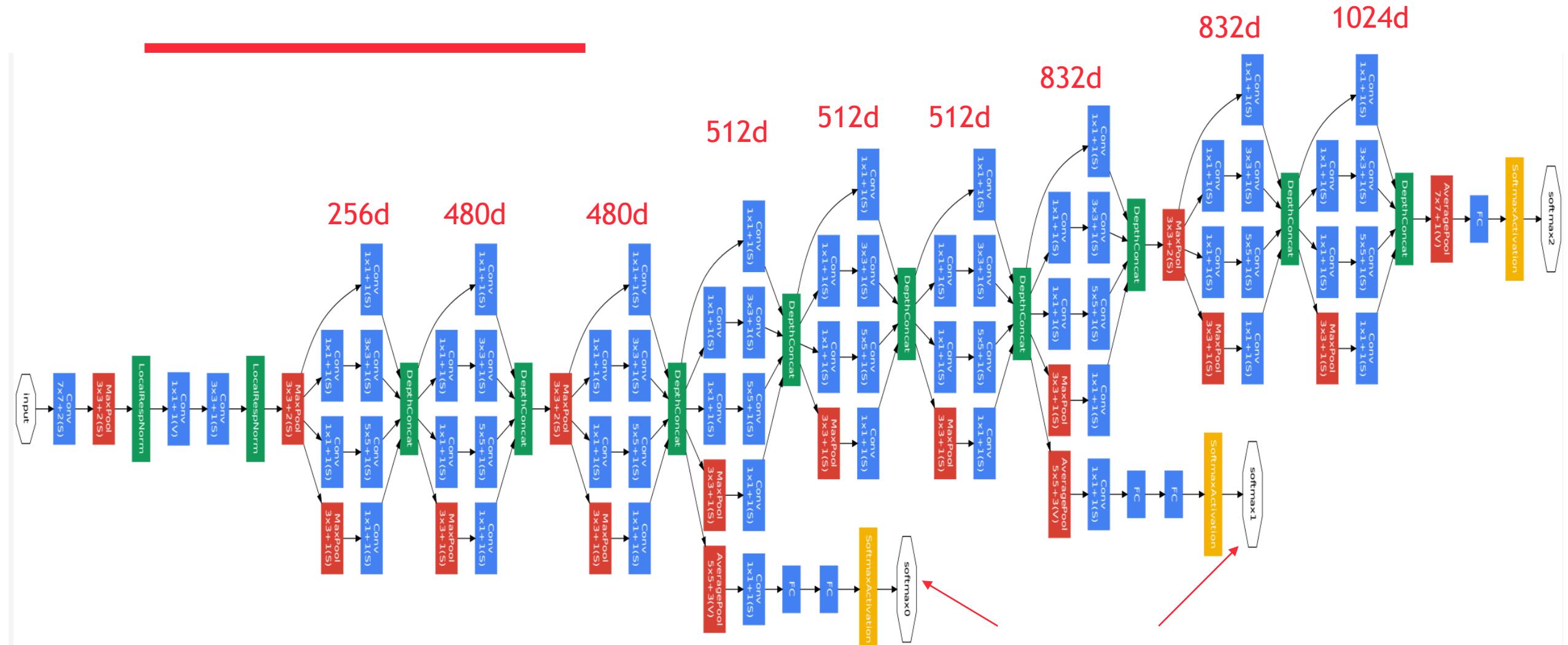


(b) Inception module with dimension reductions

# Inception / GoogLeNet (2014)



# Inception / GoogLeNet (2014)



## Дополнительные головы (только при обучении!)



# ResNet (2015)

---

- <https://arxiv.org/abs/1512.03385>
- Использование **residual-connections** для эффективного наращивание глубины сети
- Использование нормализации (batch normalization)
- До **152** слоев

# ResNet (2015)

- Наблюдение: наращивание глубины может ухудшить качество работы сети

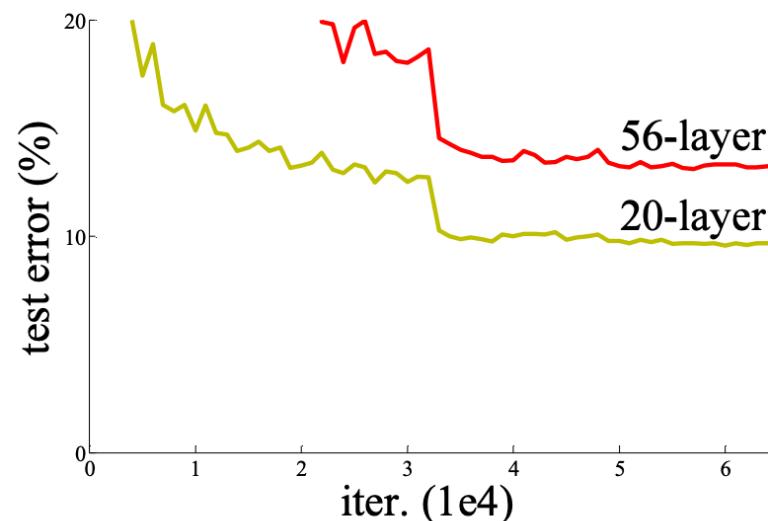
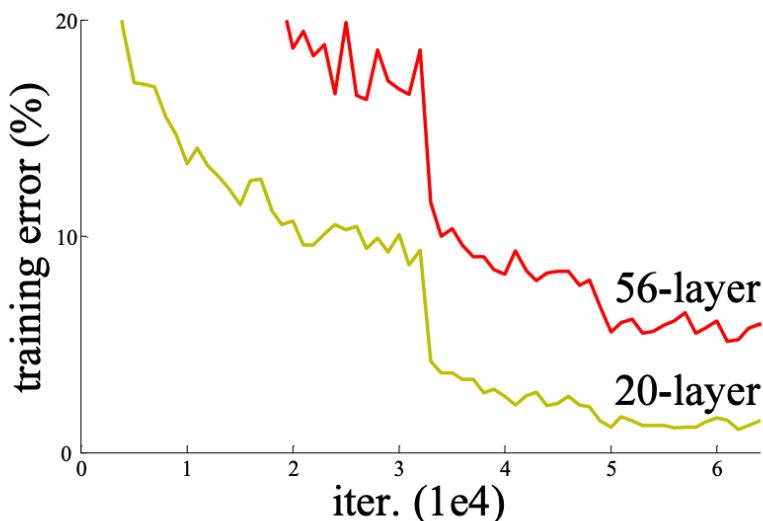


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

# ResNet (2015)

- Идея: позволить сети “пропускать” слои, если они не способствуют улучшению

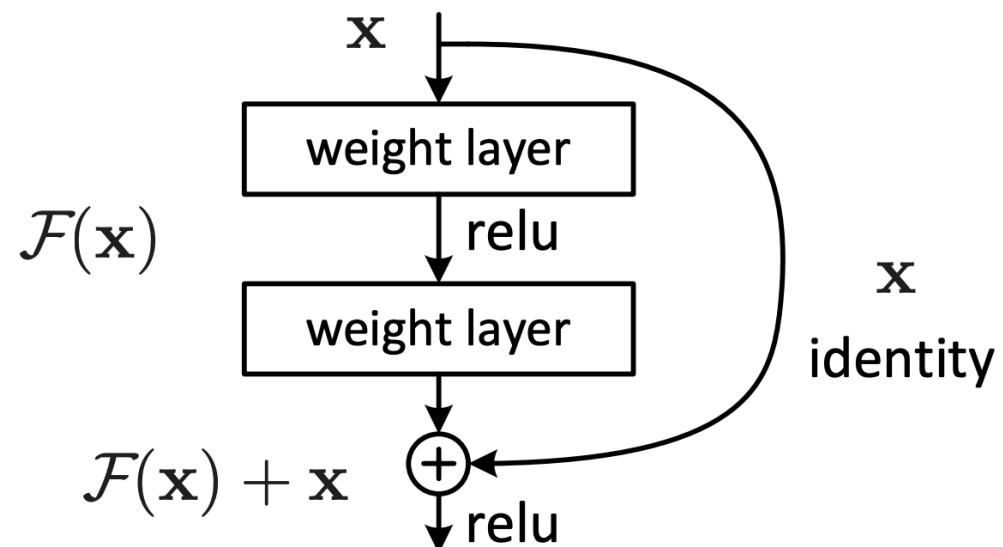


Figure 2. Residual learning: a building block.

# ResNet (2015)

- Идея: позволить сети “пропускать” слои, если они не способствуют улучшению

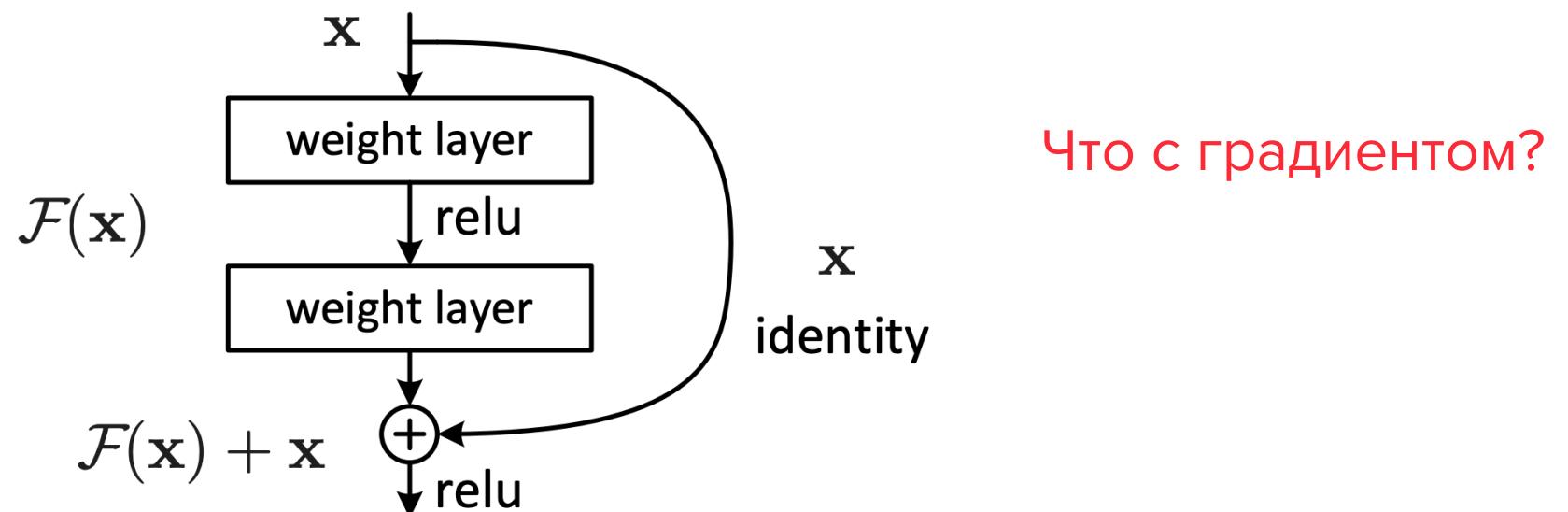
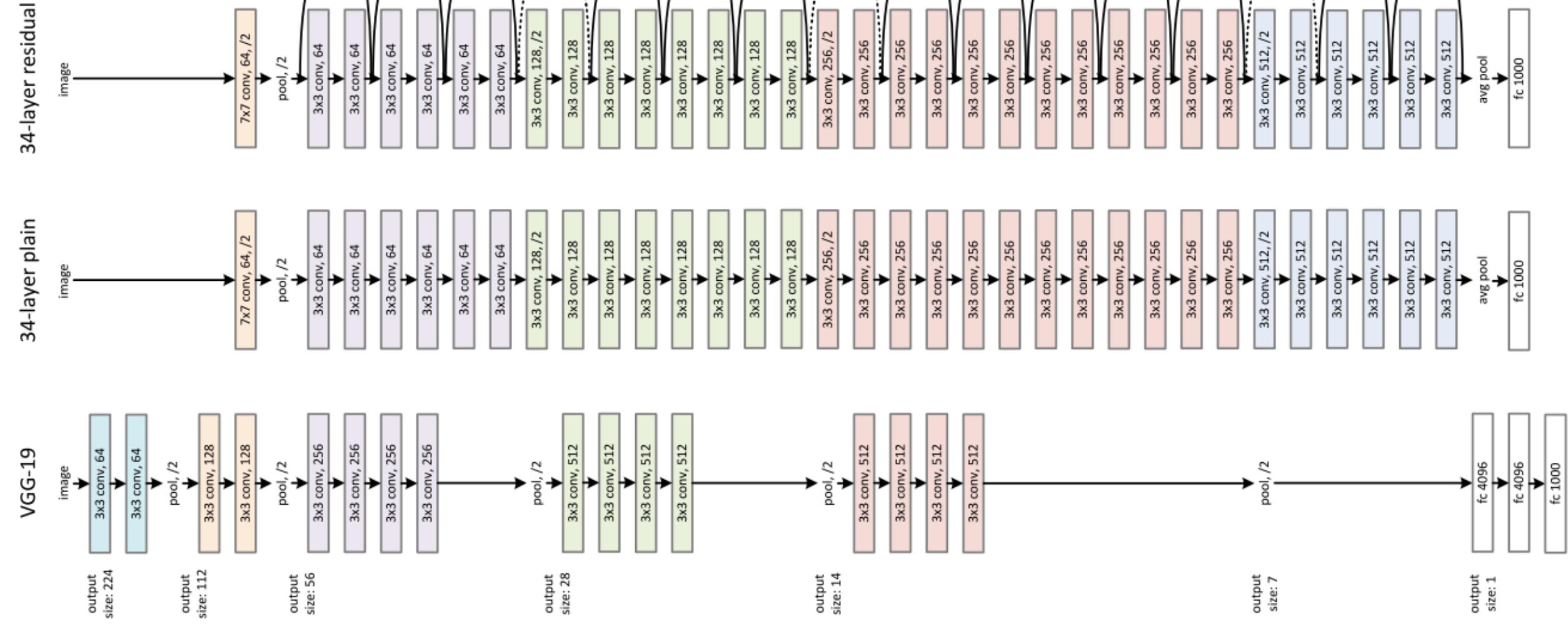


Figure 2. Residual learning: a building block.

# ResNet (2015)

## Увеличение глубины



# ResNet (2015)

- Разновидности:
  - ResNet-18, ResNet-34 (с “простым” residual-блоком)
  - ResNet-50, ResNet-101, ResNet-152 (с “bottleneck”-блоком)

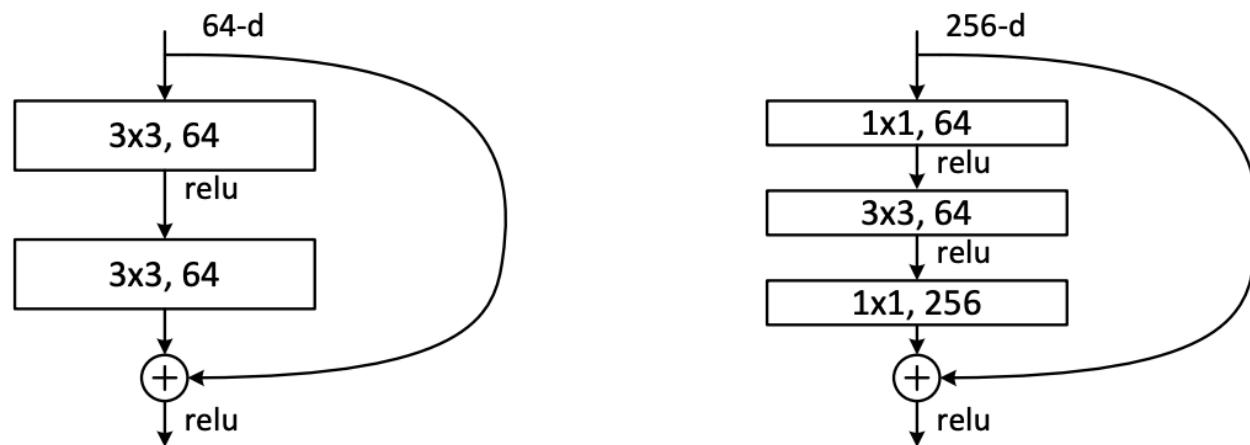


Figure 5. A deeper residual function  $\mathcal{F}$  for ImageNet. Left: a building block (on  $56 \times 56$  feature maps) as in Fig. 3 for ResNet-34. Right: a “bottleneck” building block for ResNet-50/101/152.

# ResNet (2015)

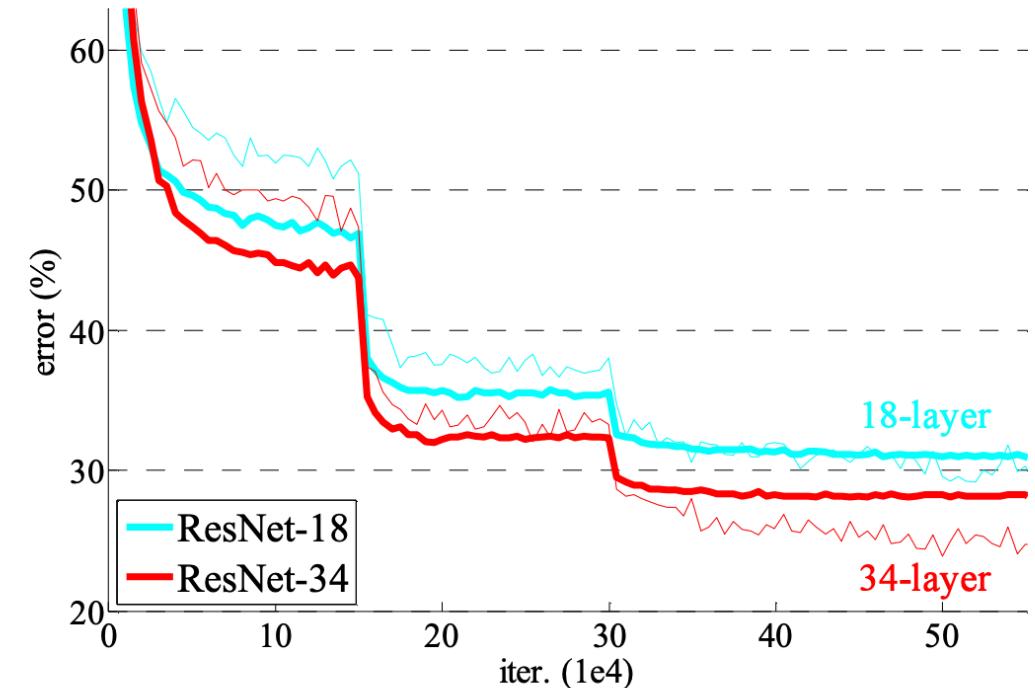
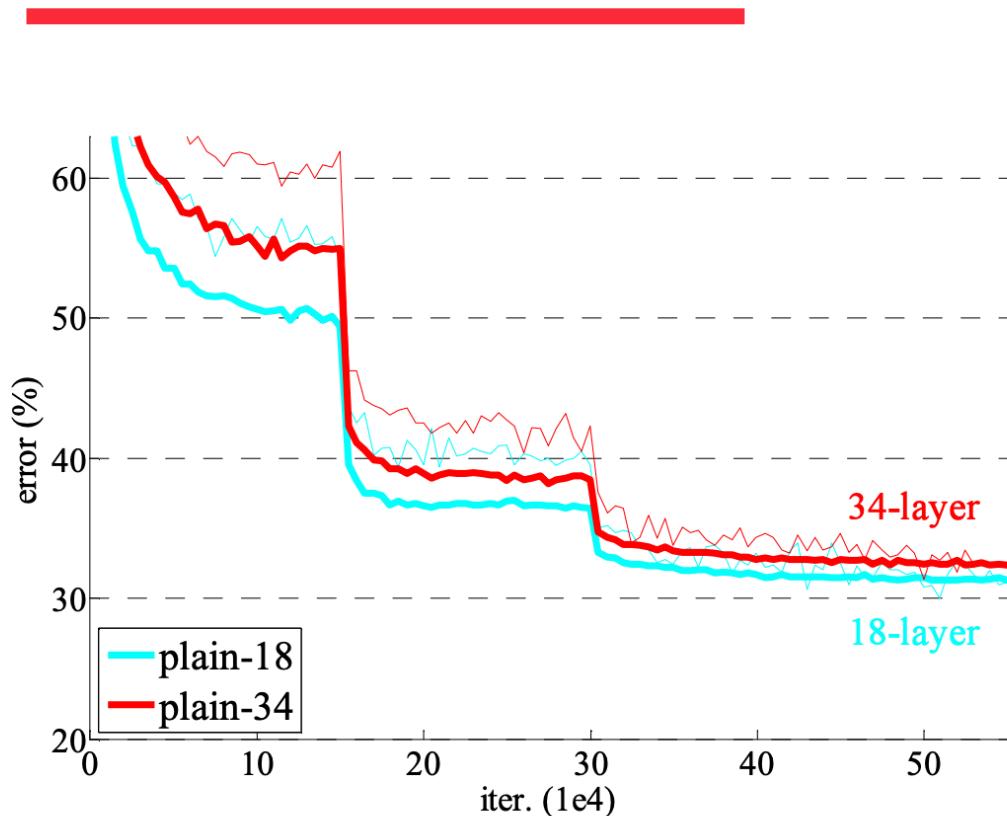
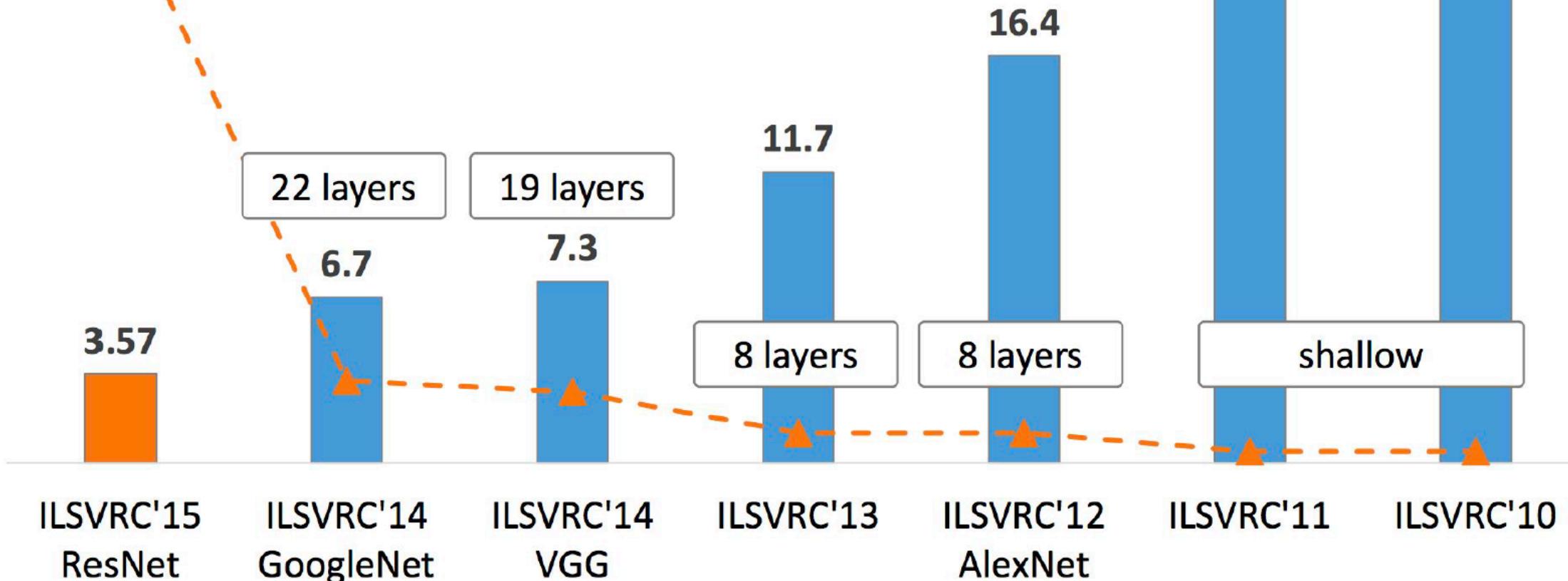


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

# Revolution of Depth

**152 layers**



ImageNet Classification top-5 error (%)



# ResNeXt (2016)

---

- <https://arxiv.org/abs/1611.05431>
- Совмещение идей **Inception** (параллельные пути вычисления признаков) и **ResNet** (residual-соединения)
- Введение “новой” размерности - **cardinality**

# ResNeXt (2016)

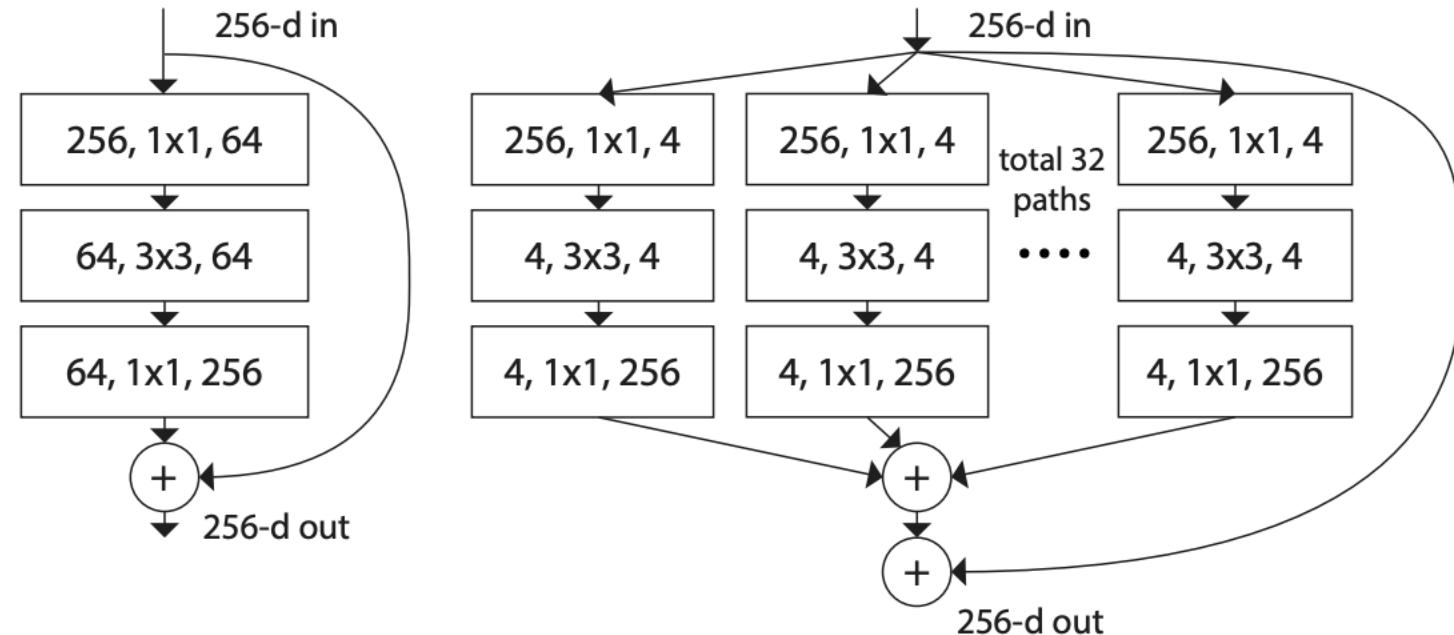
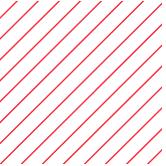


Figure 1. **Left:** A block of ResNet [14]. **Right:** A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).



# Squeeze-n-Excitation (SENet) (2017)

---

- <https://arxiv.org/abs/1709.01507>
- Идея о перевзвешивании карт признаков (дифференцируемым образом!)

# Squeeze-n-Excitation (SENet) (2017)

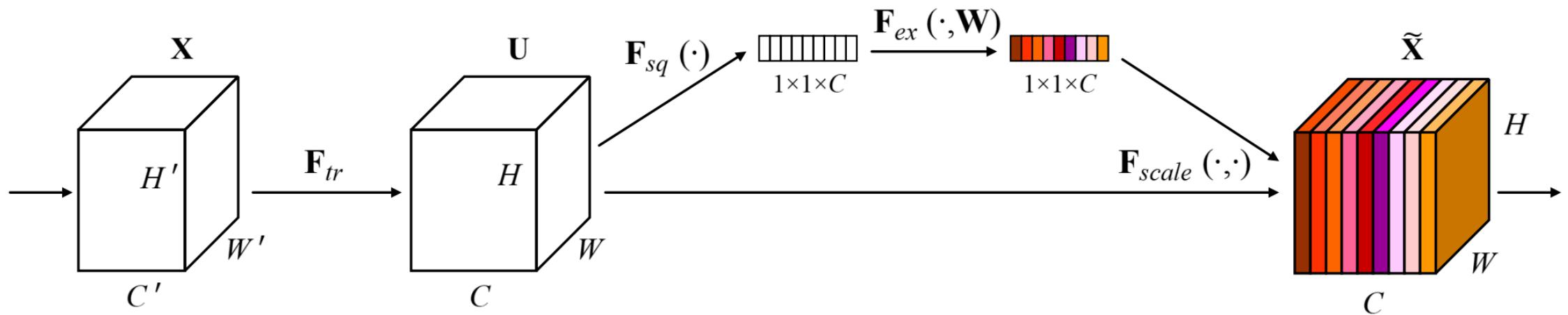
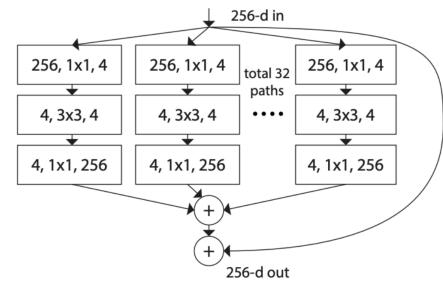
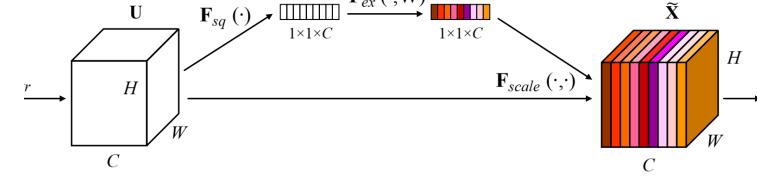


Fig. 1. A Squeeze-and-Excitation block.

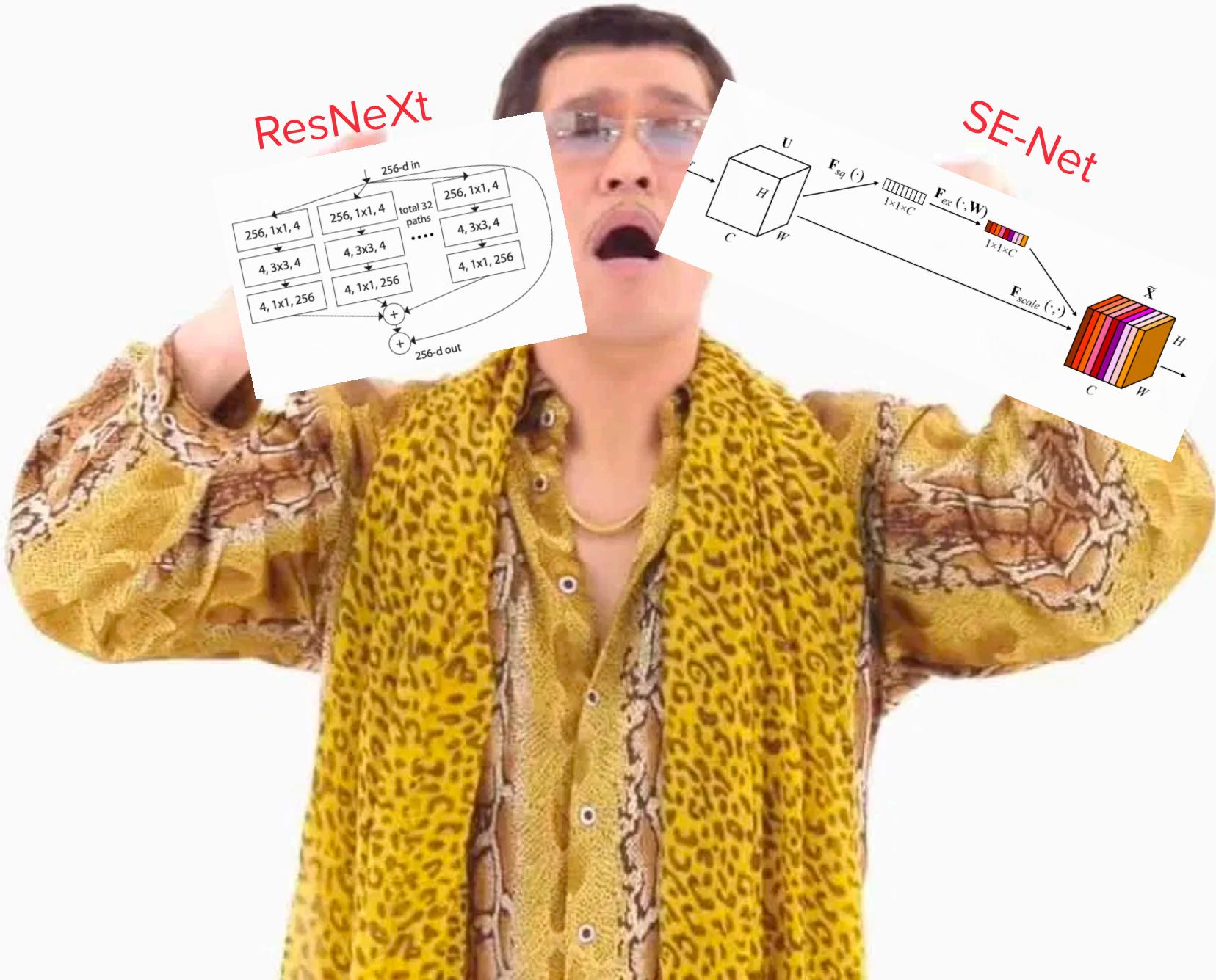
## ResNeXt



## SE-Net



# SE-ResNeXt

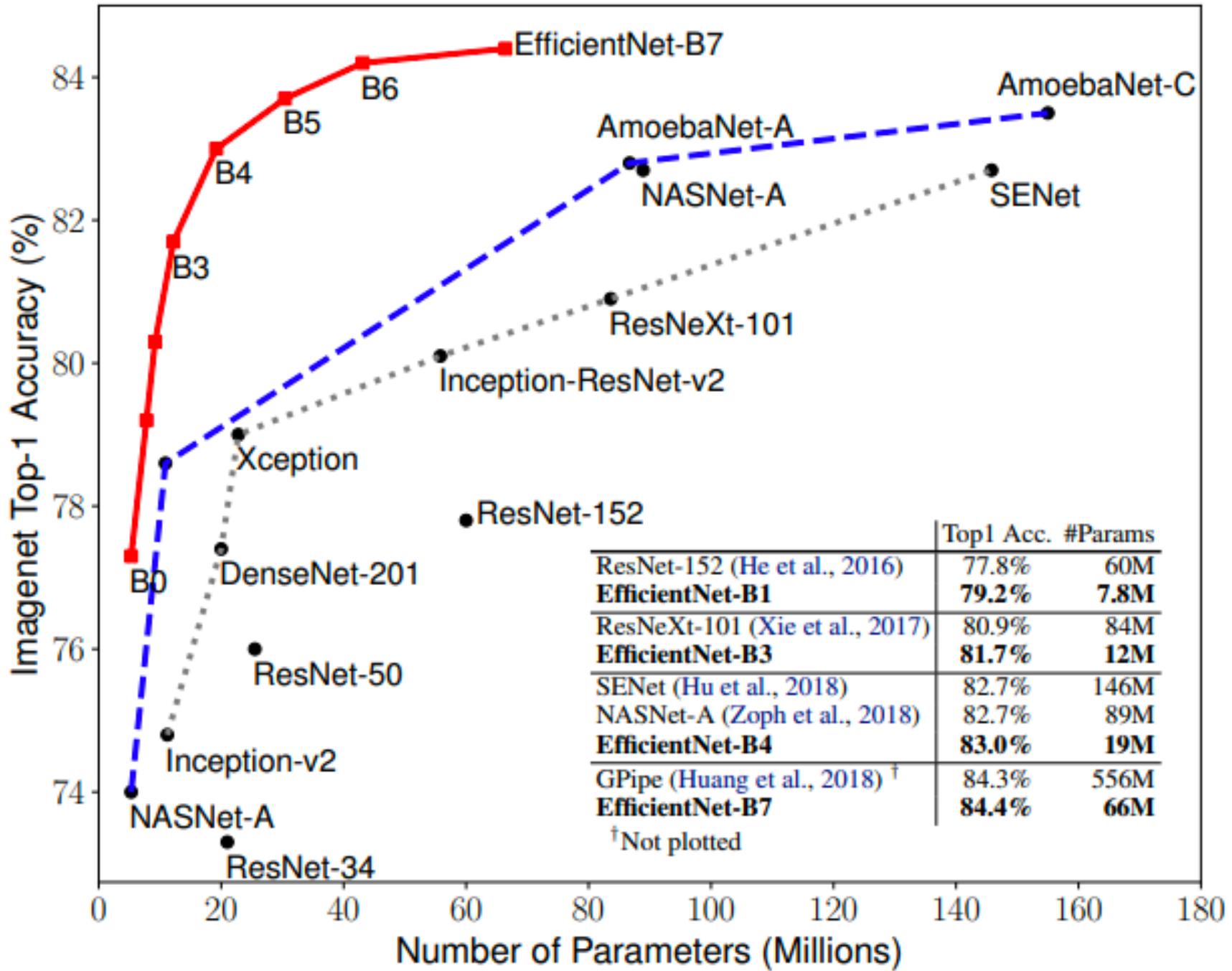




# AutoML

---

- Идея: подбирать гиперпараметры архитектур автоматически на основе метрик
- Используют эволюционные алгоритмы!
- Примеры: NASNet (2017), AmoebaNet (2018), EfficientNet (2019), ...





# Further reading

---

- MobileNet ([V1](#), [V2](#), [V3](#))
- [DenseNet](#)
- [Xception](#)
- ...

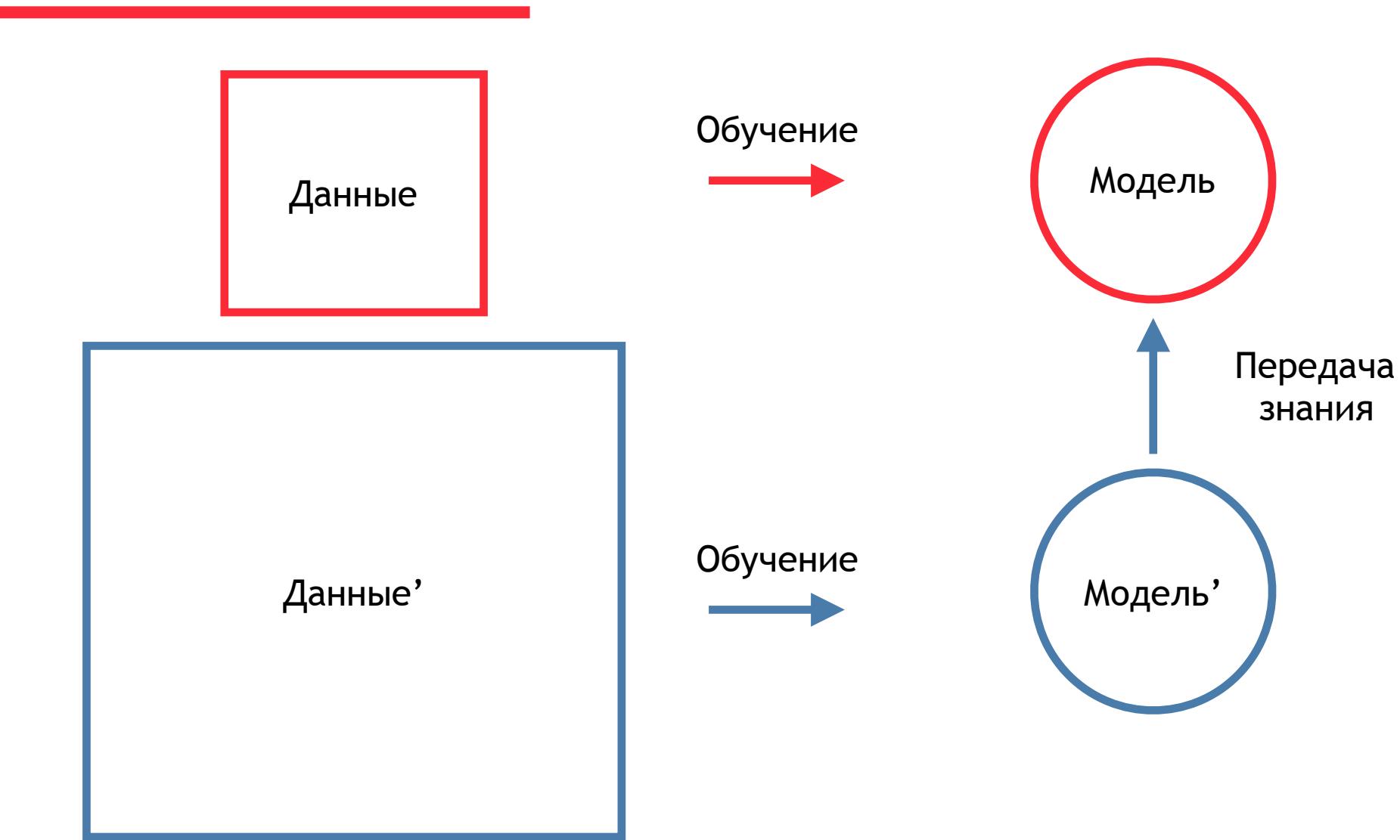
# Transfer Learning



# Traditional Learning



# Transfer Learning





# Transfer Learning

---

- Признаки из “универсального” датасета могут быть хорошей инициализацией для обучения модели на целевом домене (**fine-tuning**)
- Признаки на ранних слоях CNN - примитивы (границы, углы, комбинации цветов, ...)
- Чем “глубже” слой, тем более абстрактны признаки



# Transfer Learning

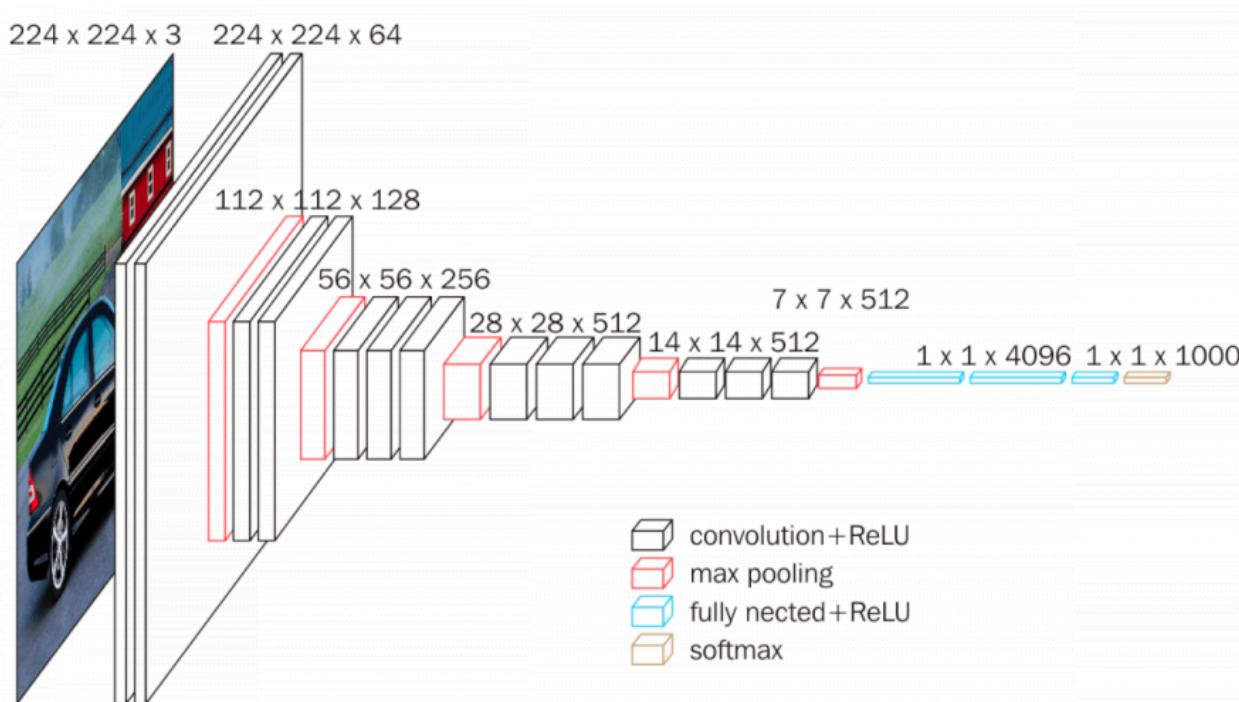
---

- ImageNet, COCO, Open Images - крупные датасеты с неплохим классовым (и внутриклассовым) разнообразием
- SOTA-архитектуры на этих датасетах = хорошие кандидаты для решения **вашей** задачи

# Transfer Learning

---

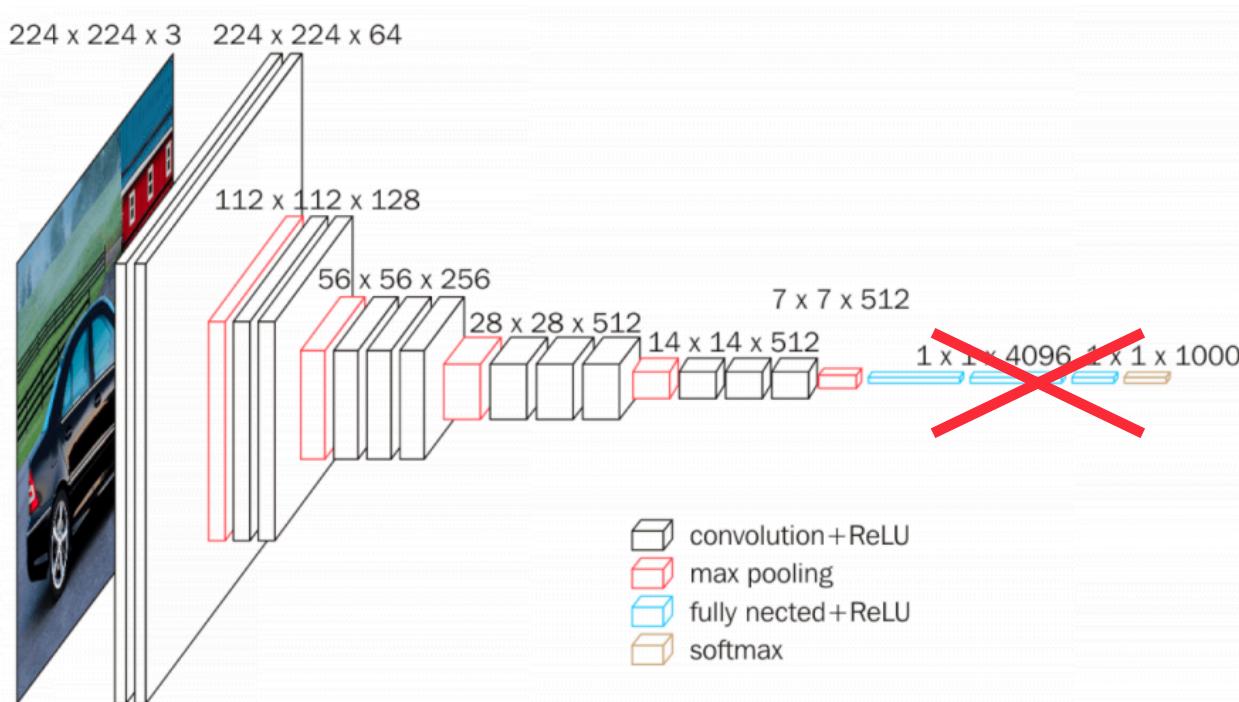
- Типичная архитектура CNN: feature extractor (conv) + classifier (fc)



# Transfer Learning

---

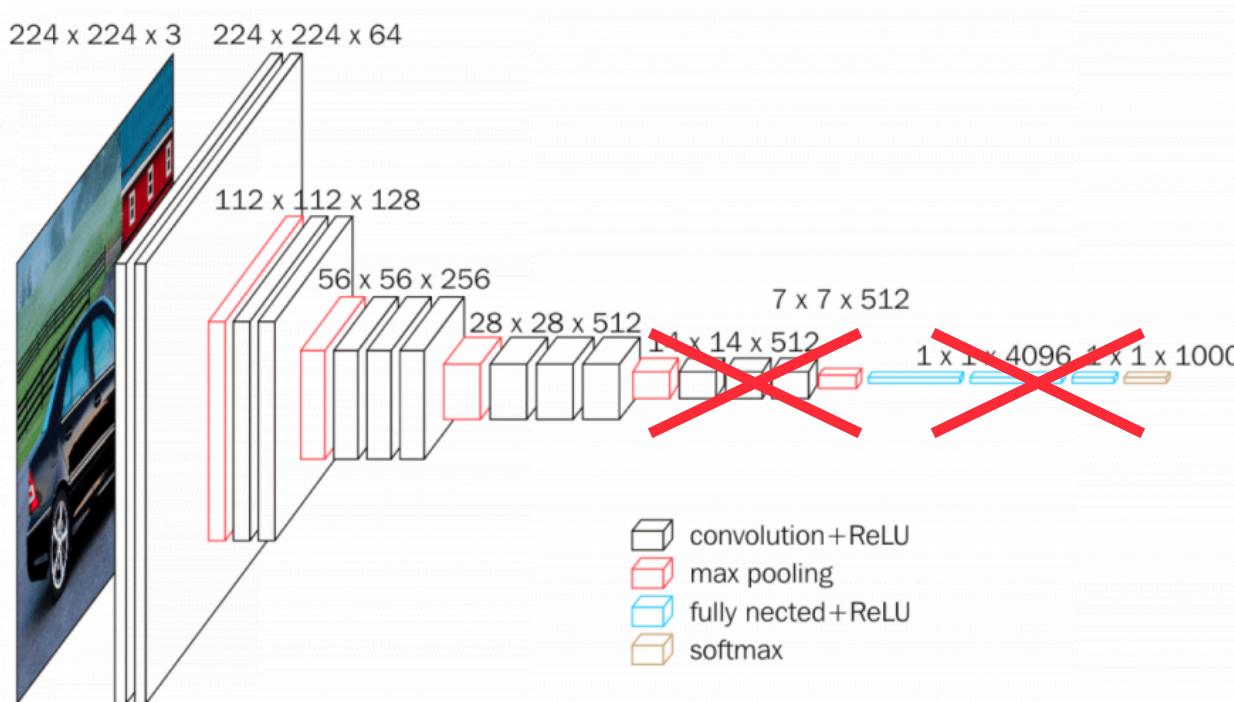
- Типичная архитектура CNN: feature extractor (conv) + classifier (fc)
- “Отрежем” classifier, получим модель для извлечения признаков



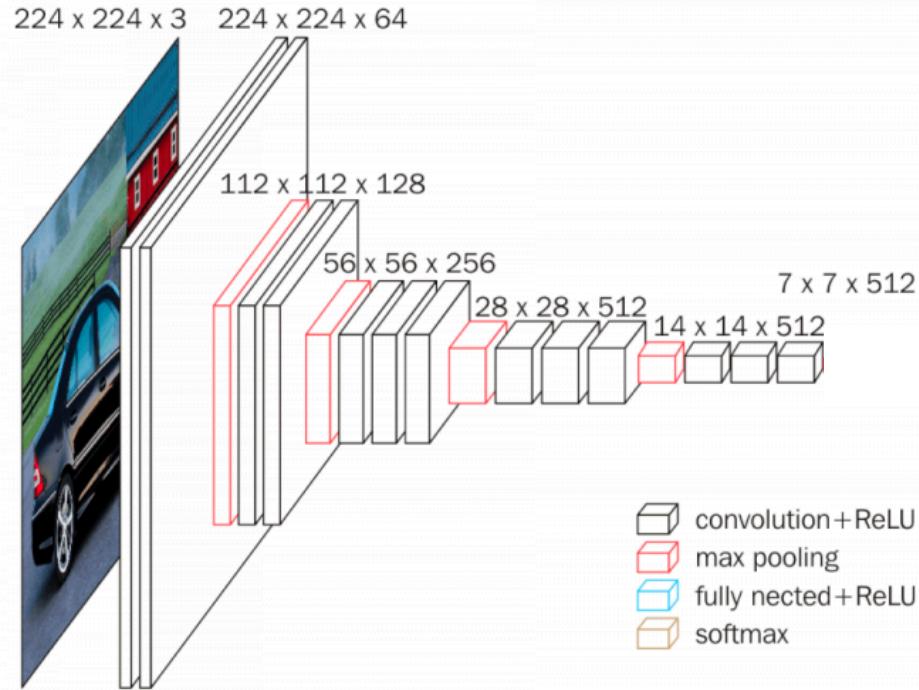
# Transfer Learning

---

- Типичная архитектура CNN: feature extractor (conv) + classifier (fc)
- “Отрежем” classifier, получим модель для извлечения признаков
- Можем отрезать слои и дальше, уменьшая их “абстрактность”



# Transfer Learning



- **Вариант #1:** добавить **только классификатор** на выходы полученной модели и обучить его на своих данных
- **Вариант #2:** добавить **сверточные слои и классификатор** и обучить



# Transfer Learning

---

Что учить?	Своих данных мало	Своих данных много
Свои данные похожи на исходные		
Свои данные отличаются от исходных		



# Transfer Learning

---

Что учить?	Своих данных мало	Своих данных много
Свои данные похожи на исходные	Только классификатор	
Свои данные отличаются от исходных		



# Transfer Learning

---

Что учить?	Своих данных мало	Своих данных много
Свои данные похожи на исходные	Только классификатор	Классификатор + высокоровневые слои
Свои данные отличаются от исходных		

# Transfer Learning

---

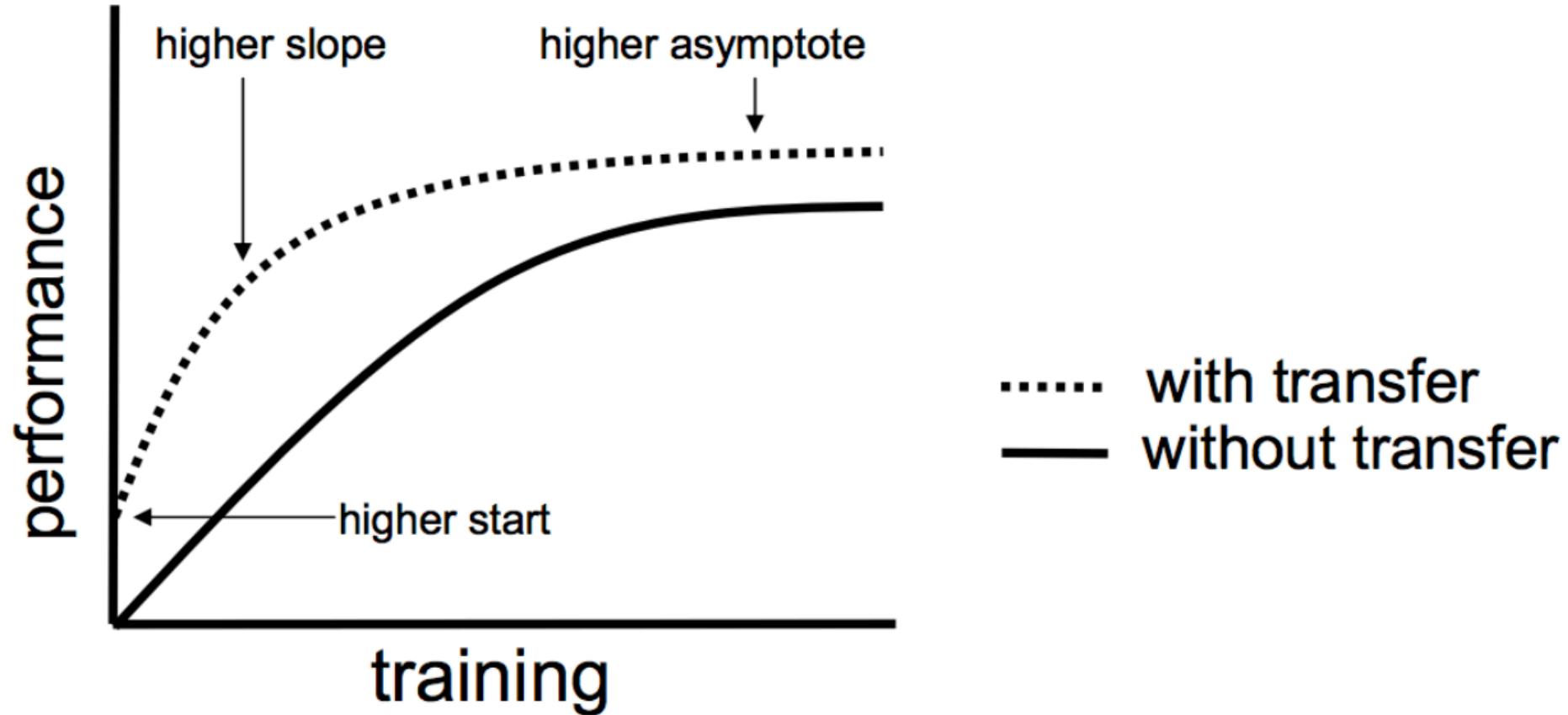
Что учить?	Своих данных мало	Своих данных много
Свои данные похожи на исходные	Только классификатор	Классификатор + высокоуровневые слои
Свои данные отличаются от исходных	Классификатор + низкоуровневые слои	

# Transfer Learning

---

Что учить?	Своих данных мало	Своих данных много
Свои данные похожи на исходные	Только классификатор	Классификатор + высокоуровневые слои
Свои данные отличаются от исходных	Классификатор + низкоуровневые слои	Классификатор + высокоуровневые слои + низкоуровневые

# Transfer Learning



# Резюме



# Резюме

---

- Самые эффективные идеи могут быть очень простыми
- Автоматизация подбора архитектуры не за горами
- Используйте Transfer Learning для своих задач



# В следующий раз

---

- Обучение глубоких сетей: нормализация, регуляризация, аугментация, ...
- OpenCV