Современные сети

ML => DL

- Классика Машинного Обучения = Выделение признаков + модель для обучения
- Classic Machine Learning = feature extraction + model

X=(x1,x2, …; «усы - длинные», «уши - большие», «глаза - желтые»)=КОТ

$$F(X,W)$$

- Глубокое обучение = сырые данные + глубокая модель для обучения
- Deep Learning = raw data + deep model

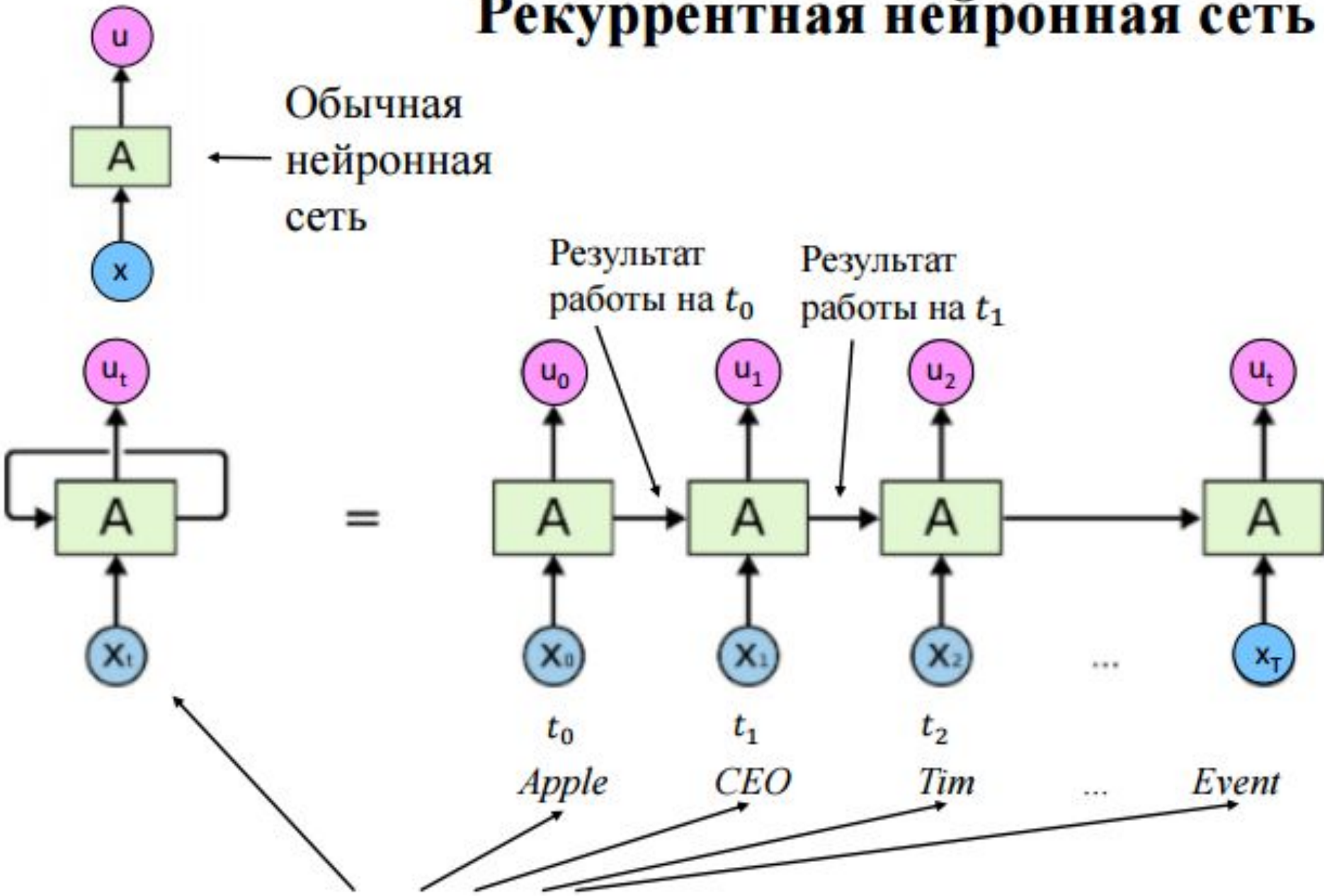
= КОТ

$$F(X,W)$$

LSTM

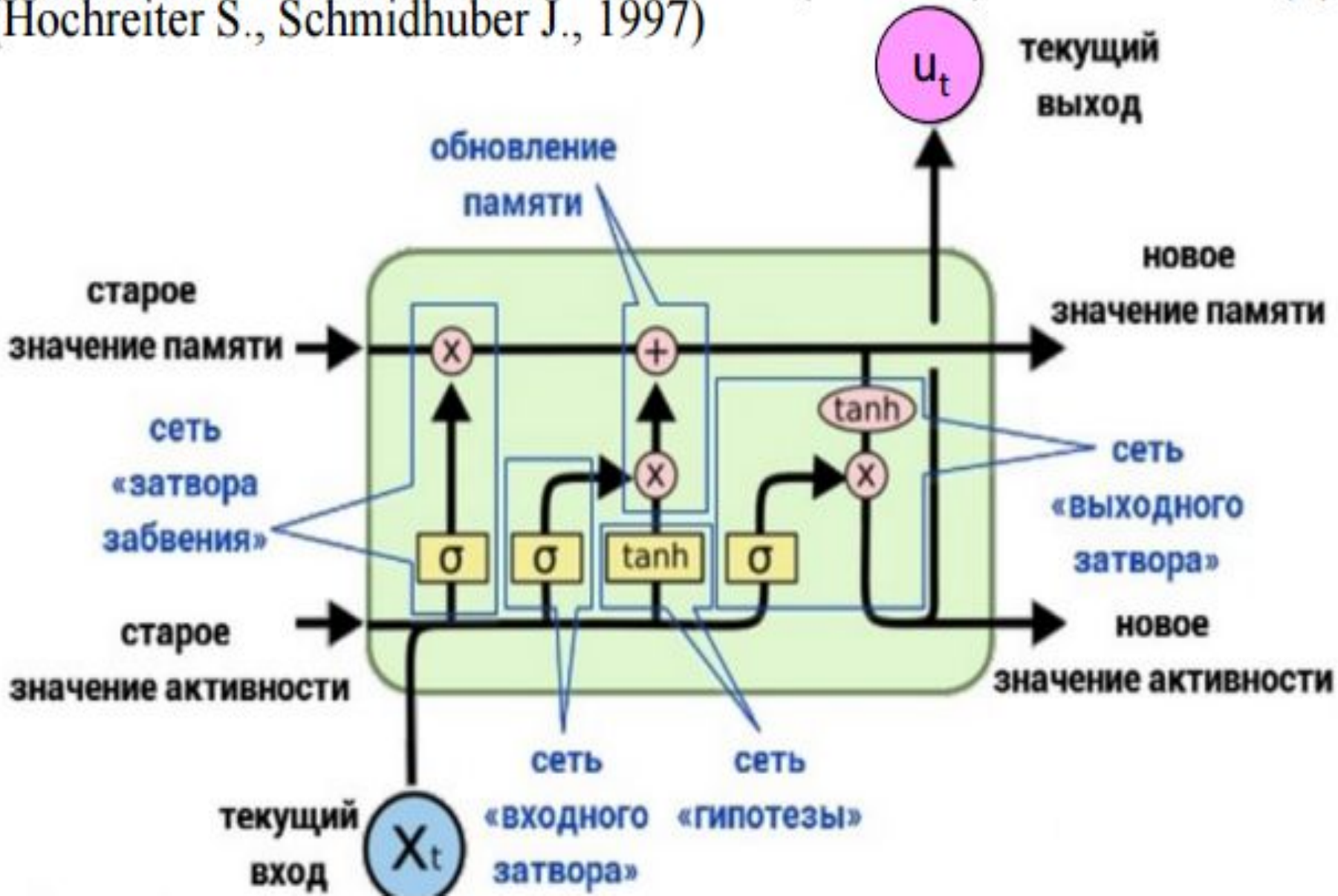
- Юрген Шмидхубер 1997

Рекуррентная нейронная сеть

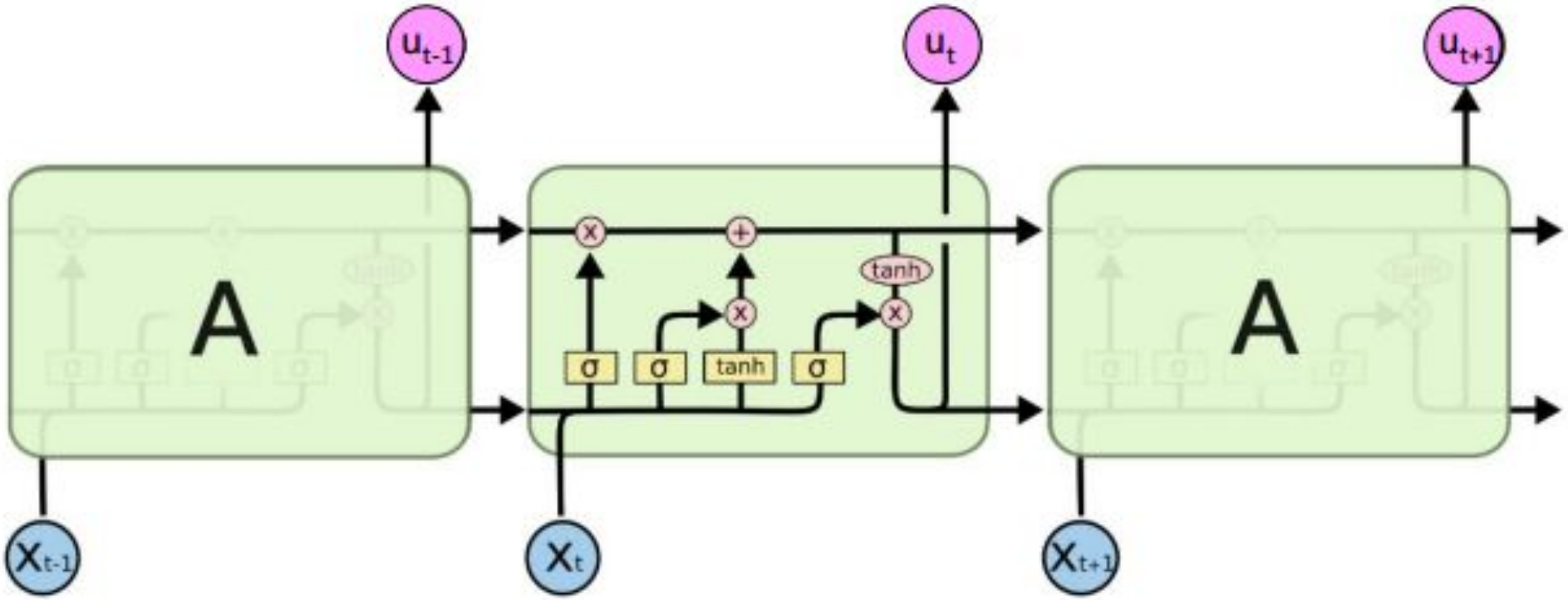


Рекуррентная нейронная сеть с long-short memory (LSTM) ячейками (1)

(Hochreiter S., Schmidhuber J., 1997)

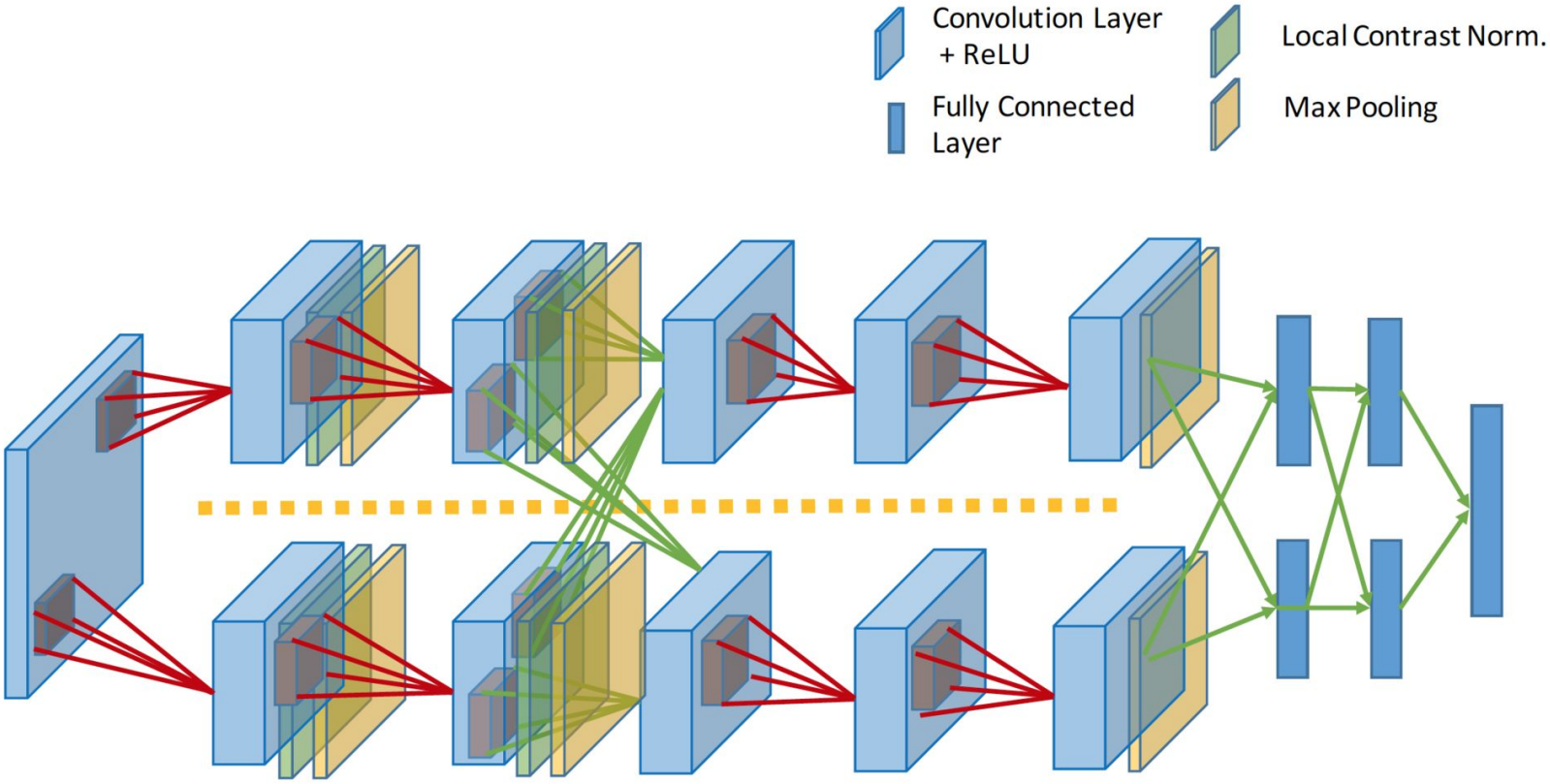


Рекуррентная нейронная сеть с LSTM ячейками (2)



AlexNet

- DropOut (RIP), Data Augmentation и ReLu.



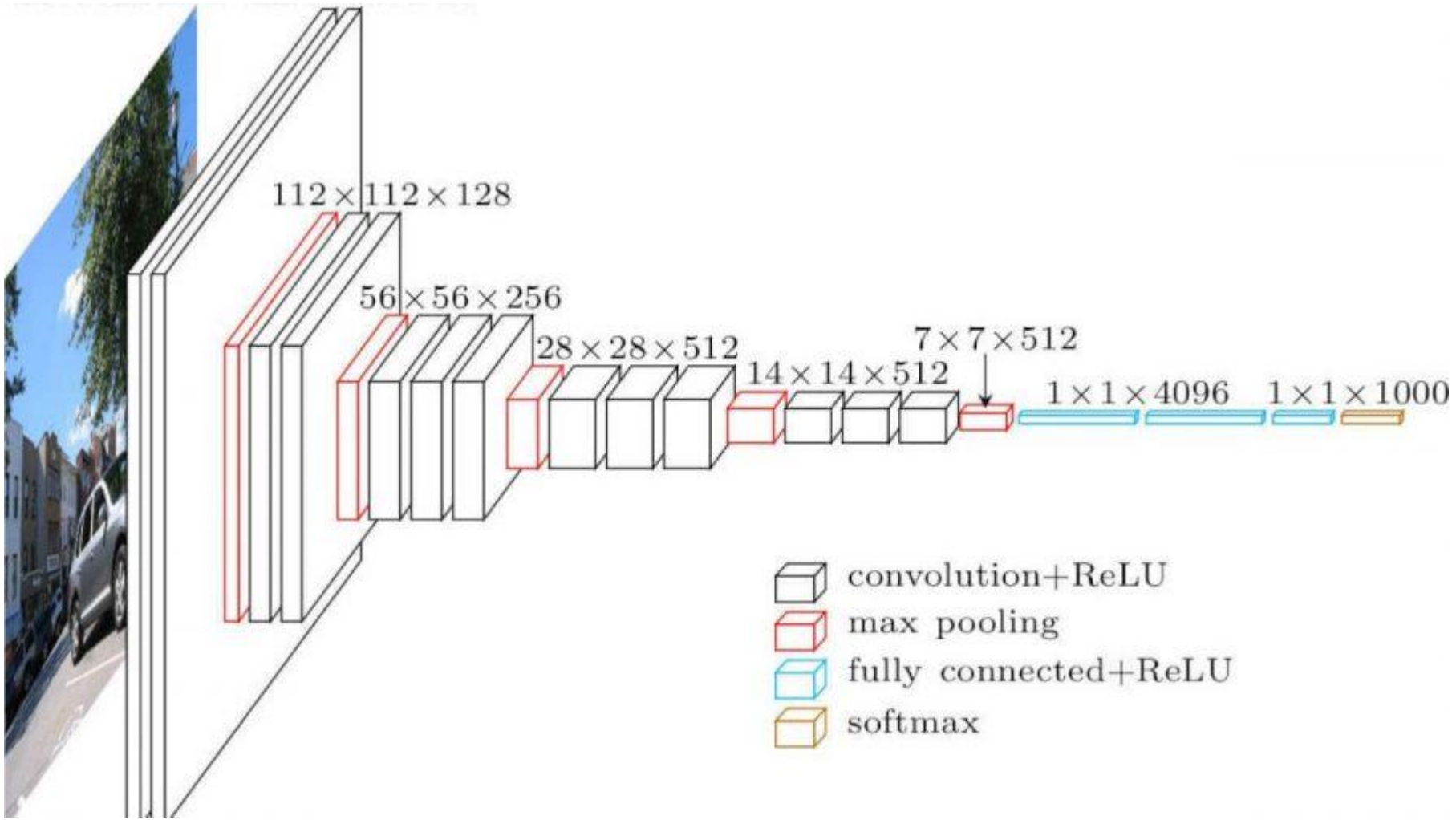
AlexNet

- DropOut (RIP), Data Augmentation и ReLu.
- 1 неделя обучения на 2 GPUs

VGG16

- 16 слоев
- Вход: изображение размером 224x224 пиксела, 3 канала цвета
- 2 раза : свертка-свертка-подвыборка,
- 3 раза: свертка-свертка-свертка-подвыборка
  - Ядро 3x3
  - Пулинг (подвыборка)2x2
- 3 полносвязных слоя:
- 4096 нейронов, 4096 нейронов , 1000 нейронов
- 1000 классов объектов.
- Выход: вероятность (one hot encoding) класса объекта

VGG16


112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

VGG16

- Очень медленная скорость обучения.
- Сеть слишком большая (появляются проблемы с диском и пропускной способностью)

VGG19

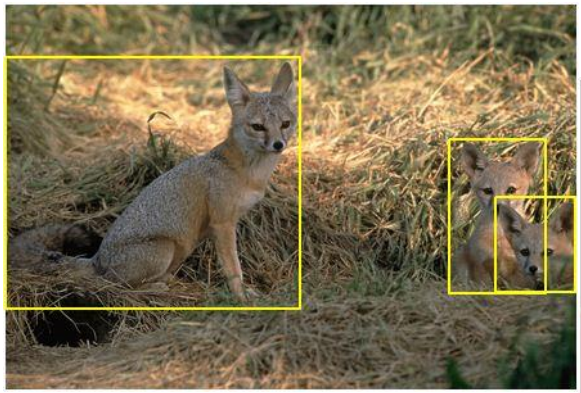
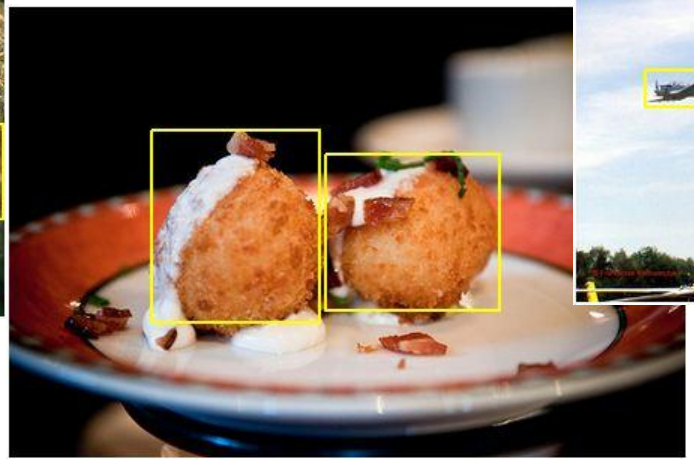
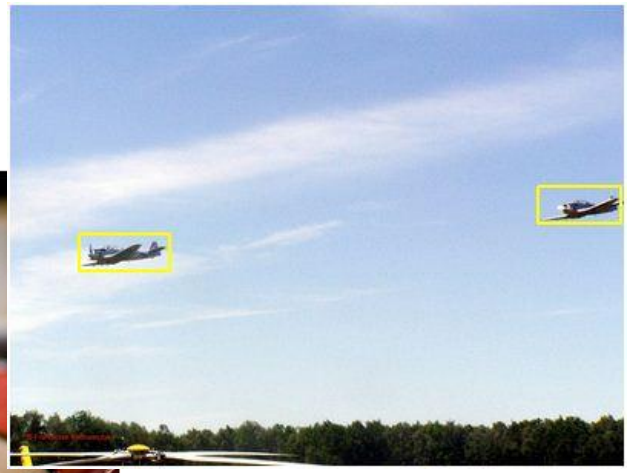
- 19 слоев
- Вход: изображение размером 224x224 пиксела, 3 канала цвета
- 2 раза : свертка-свертка-подвыборка,
- 3 раза: свертка-свертка-свертка-***свертка*** - подвыборка
  - Ядро 3x3
  - Пулинг (подвыборка)2x2
- 3 полносвязных слоя:
- 4096 нейронов, 4096 нейронов , 1000 нейронов
- 1000 классов объектов.
- Выход: вероятность (one hot encoding) класса объекта

VGG19

- Не обучается целиком (затухает градиент)
- Несколько стадий обучения разной глубины
- Обучение 4 Titan Black GPUs 2-3 недели

VGG16- VGG16

- Задача(https://www.kaggle.com/c/imagenet-object-localization-challenge/data)
- ImageNet (1000 классов) ≈ 7 %

Visual Geometry Group

Department of Engineering Science, University of Oxford

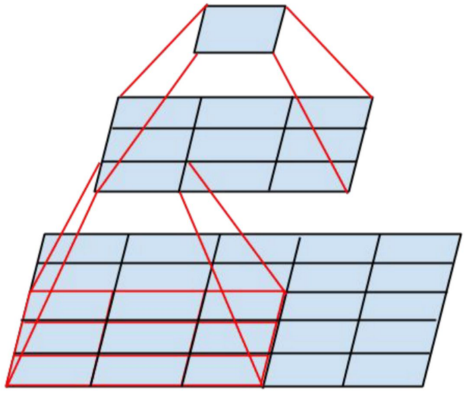
K. Simonyan, A. Zisserman

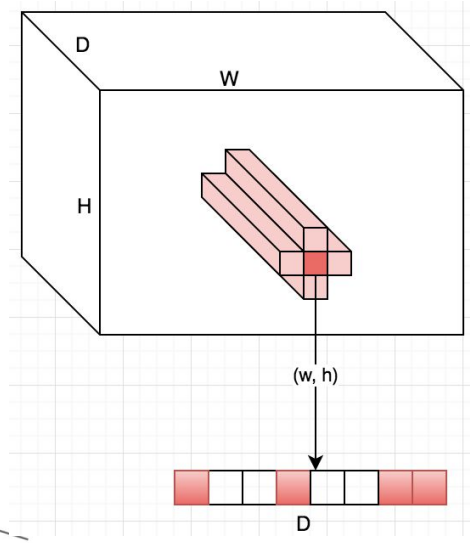
Very Deep Convolutional Networks for Large-Scale Image Recognition

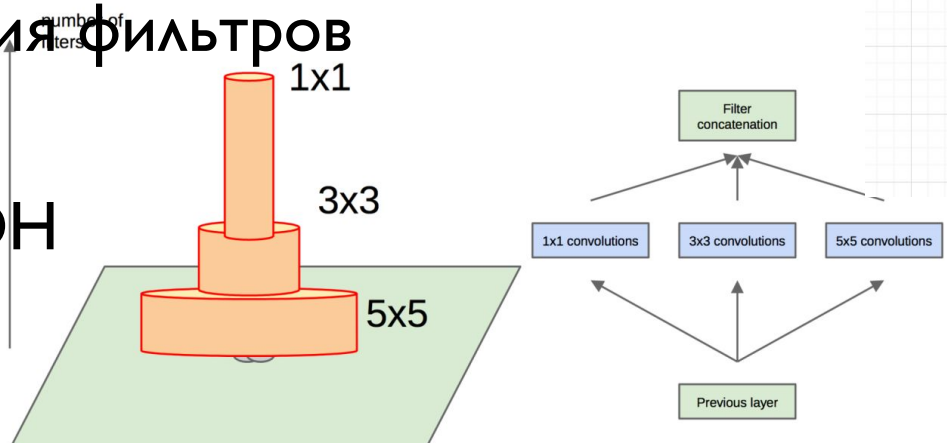
arXiv technical report, 2014

GoogleNet - Inception

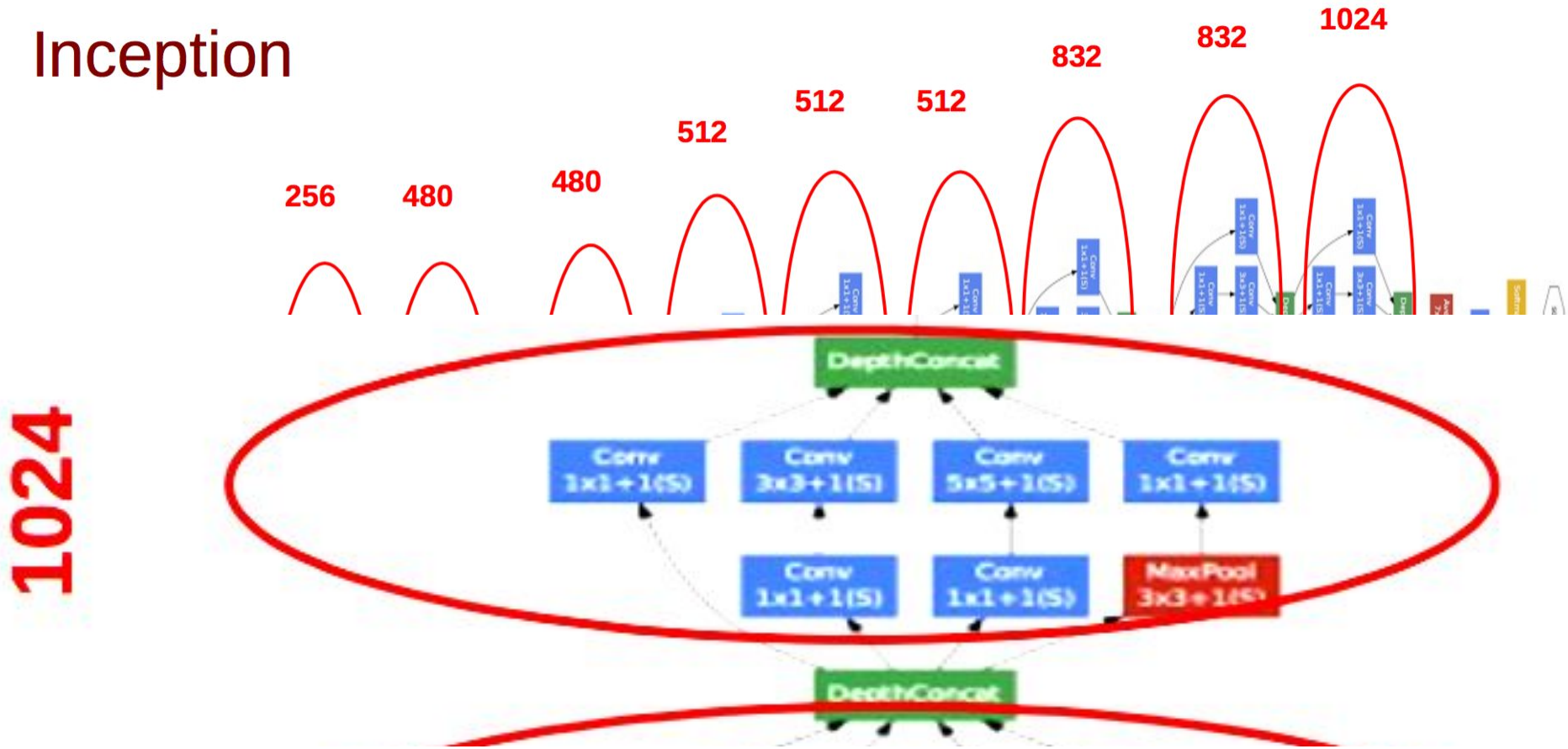
- 144 миллиона параметров
- Стек сверток (nxn по 3x3)
  - Cascade Cross Chanel Parametric Pooling
- Анализ корреляции нейронов предыдущего слоя и объединение коррелированных нейронов в группы, которые будут нейронами следующего слоя.
- Конкатенация фильтров
- ИНСЕПТРОН







GoogleNet - Inception



Width of **inception modules** ranges from 256 filters (in early modules) to 1024 in top inception modules.

GoogleNet - Inception

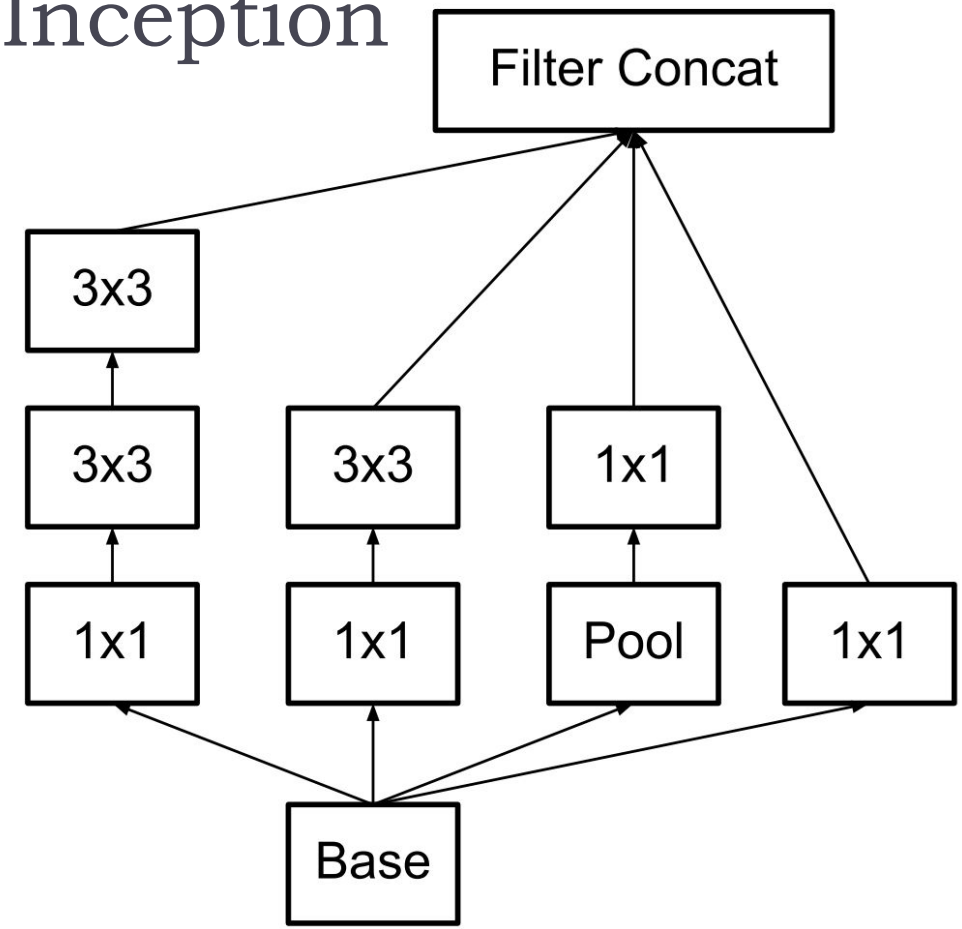
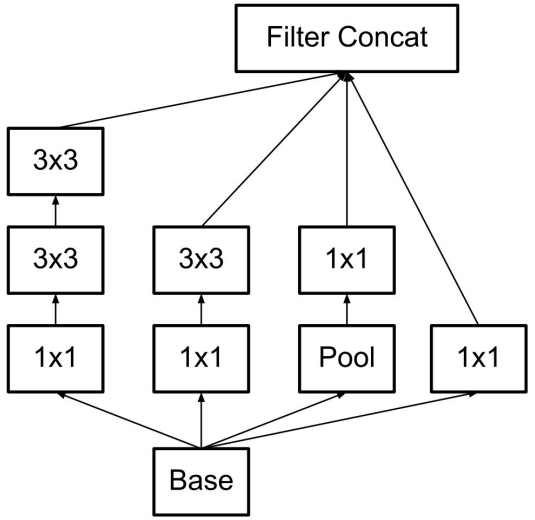


Figure 5. Inception modules where each 5×5 convolution is replaced by two 3×3 convolution, as suggested by principle 3 of Section 2.

Rethinking the Inception Architecture for Computer Vision (11 Dec 2015)
// https://arxiv.org/pdf/1512.00567v3.pdf

GoogleNet - Inception

- Макс. глубина: 22 слоя с параметрами
- Нет полносвязных слоев
- В 12 раз меньше параметров чем в AlexNet
- Основной блок – Inception module (9 штук)
- 1x1 свёртки – уменьшение размерности



Rethinking the Inception Architecture for Computer Vision (11 Dec 2015)
// https://arxiv.org/pdf/1512.00567v3.pdf

GoogleNet

- GoogleNet — первая известная сеть с ациклической архитектурой

ResNet

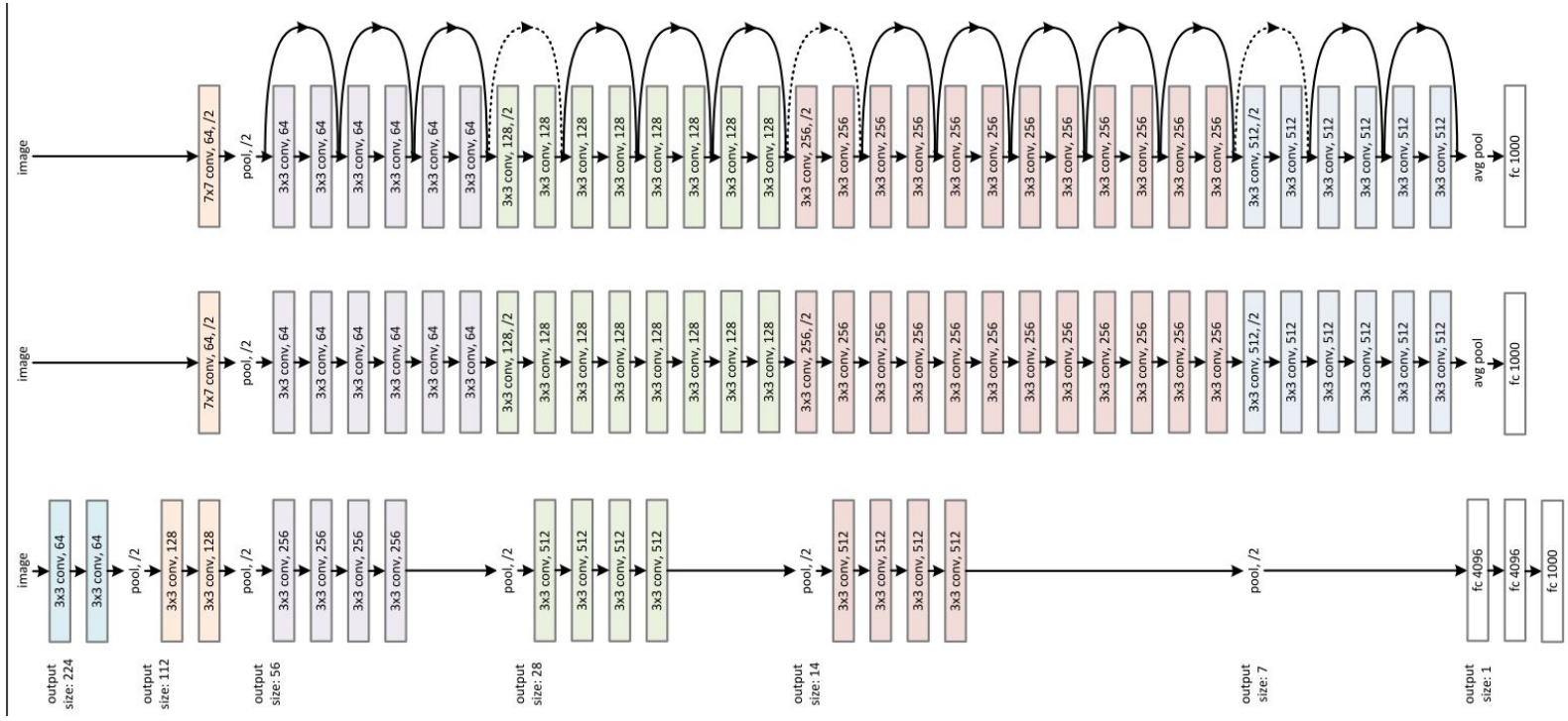


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

residual-функция

- остаточная функция: $\mathcal{F}(x) = \mathcal{H}(x) - x$
- Можно обучить на эту функцию

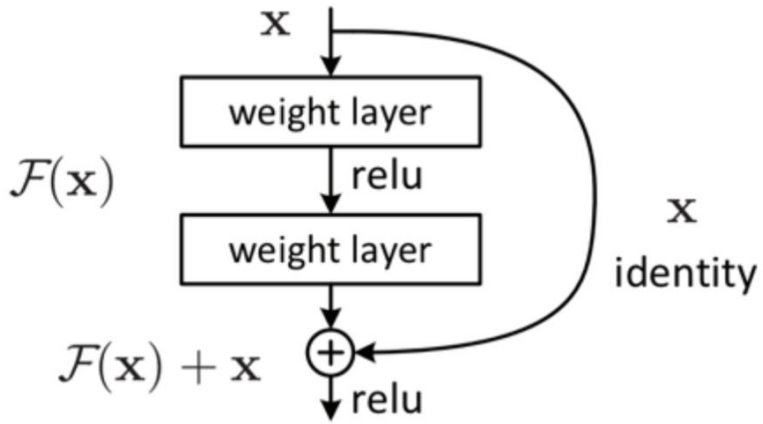


Figure 2. Residual learning: a building block.

$\mathcal{H}(\mathbf{x})$ is the true function we want to learn

Let's pretend we want to learn

$\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$

instead.

The original function is then

$\mathcal{F}(\mathbf{x}) + \mathbf{x}$

ResNet

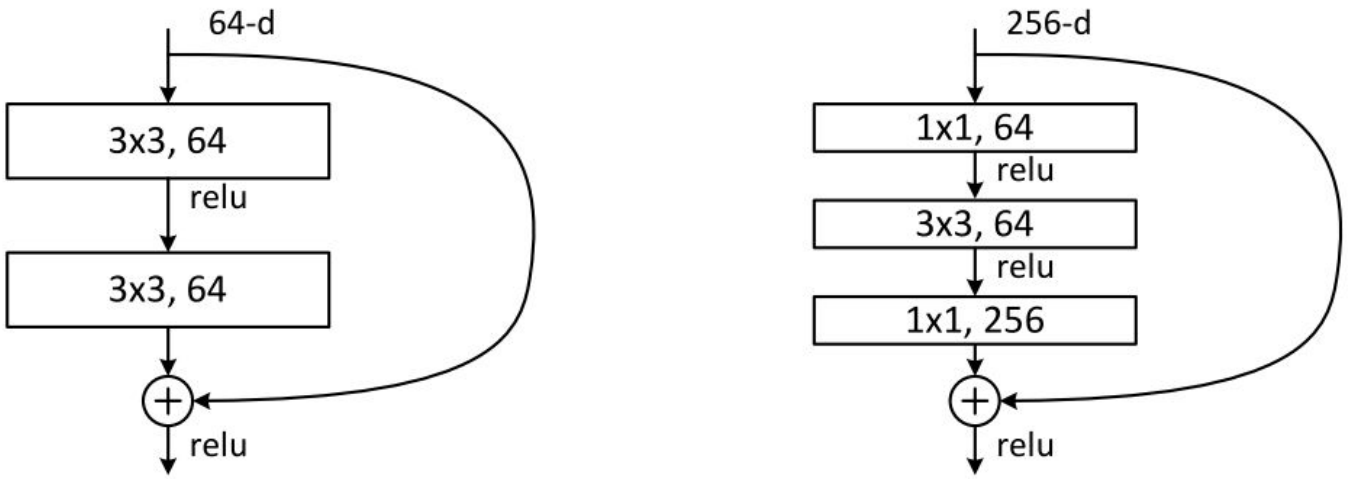
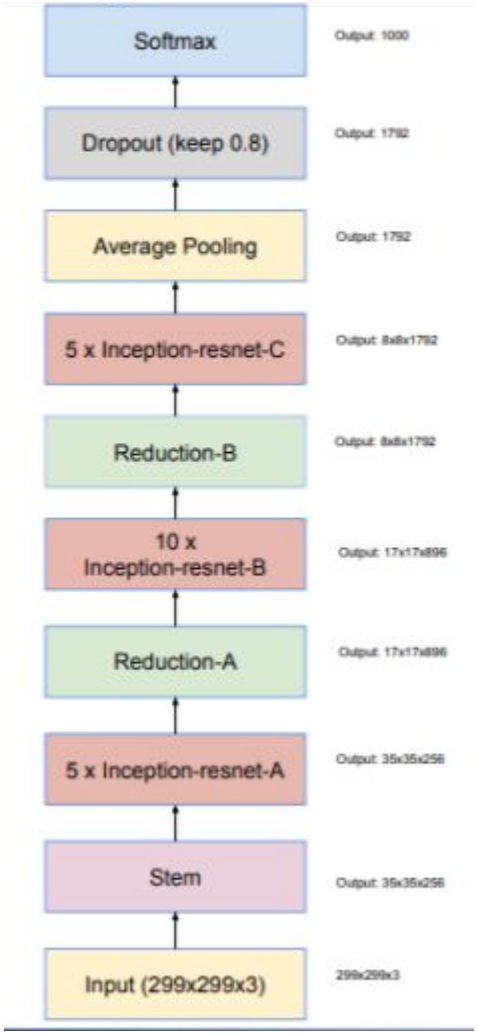


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

ResNet - задача

- ImageNet
- 3.57% - top5 на Imagenet. 2014
- 152 слоя

ResNet+Inception: Inception 4v



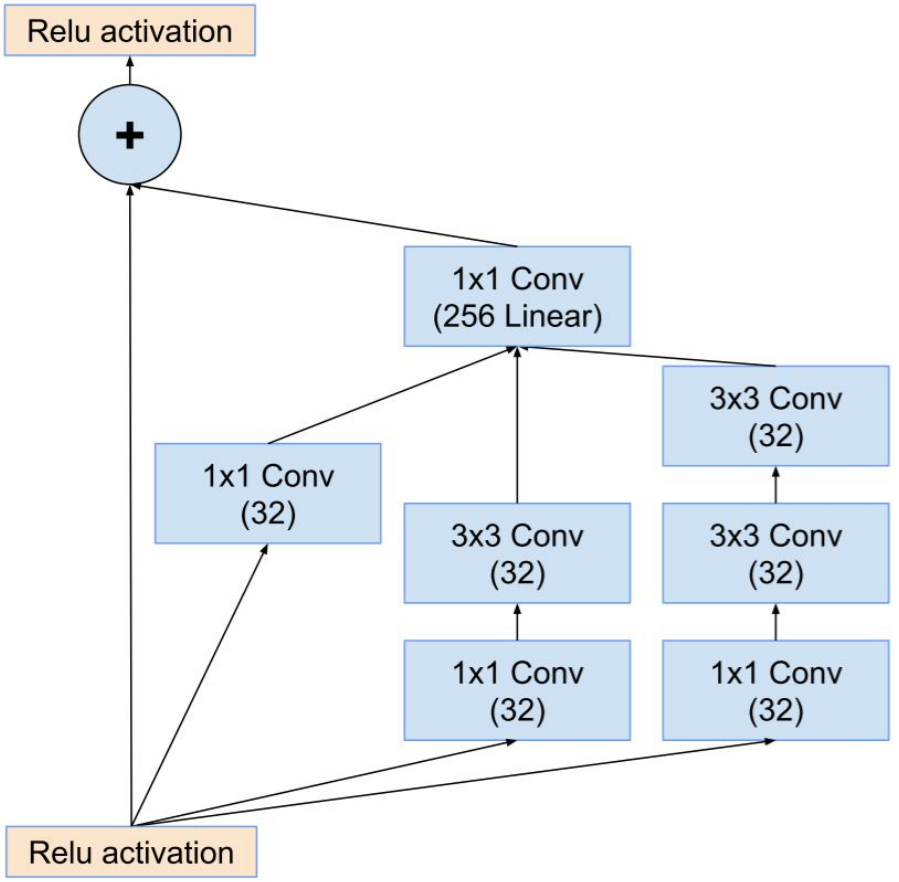


Figure 10. The schema for 35×35 grid (Inception-ResNet-A) module of Inception-ResNet-v1 network.

ResNet+Inception

- Очень глубокая сеть(более 550 слоев) , целиком не обучается (55M параметров)
- Один и тот же блок с функций потерь в нескольких местах
- «Проталкивает» градиент внутрь сети
- Обучалось на облаке CPU
- Добавился BatchNorm, Residual blocks и т.д. (Inception-v4)

- ImageNet : 3.08% 2016

- https://arxiv.org/pdf/1602.07261.pdf

DenseNet

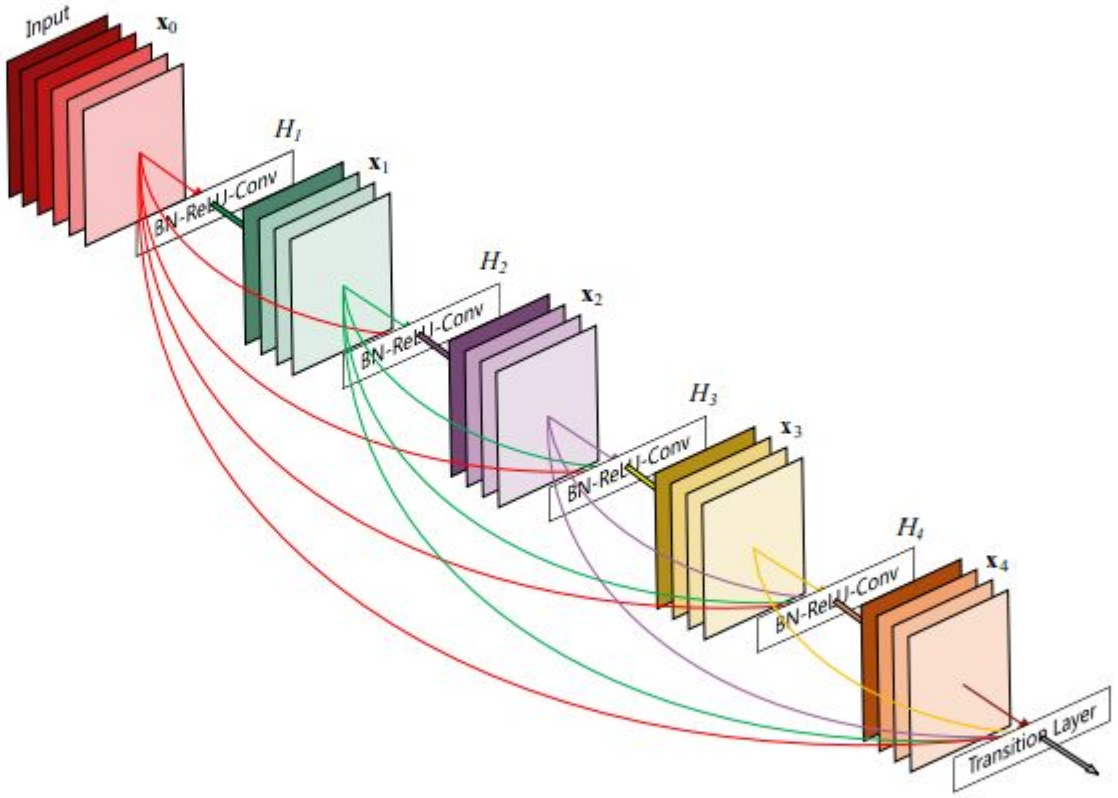


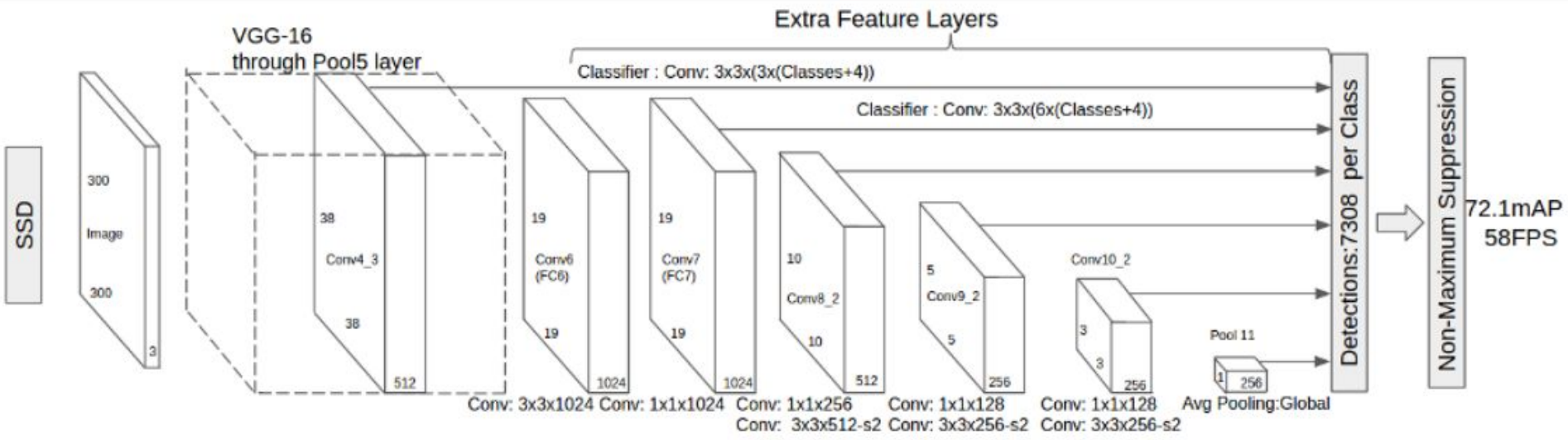
Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

https://arxiv.org/pdf/1608.06993.pdf

SqueezeNet

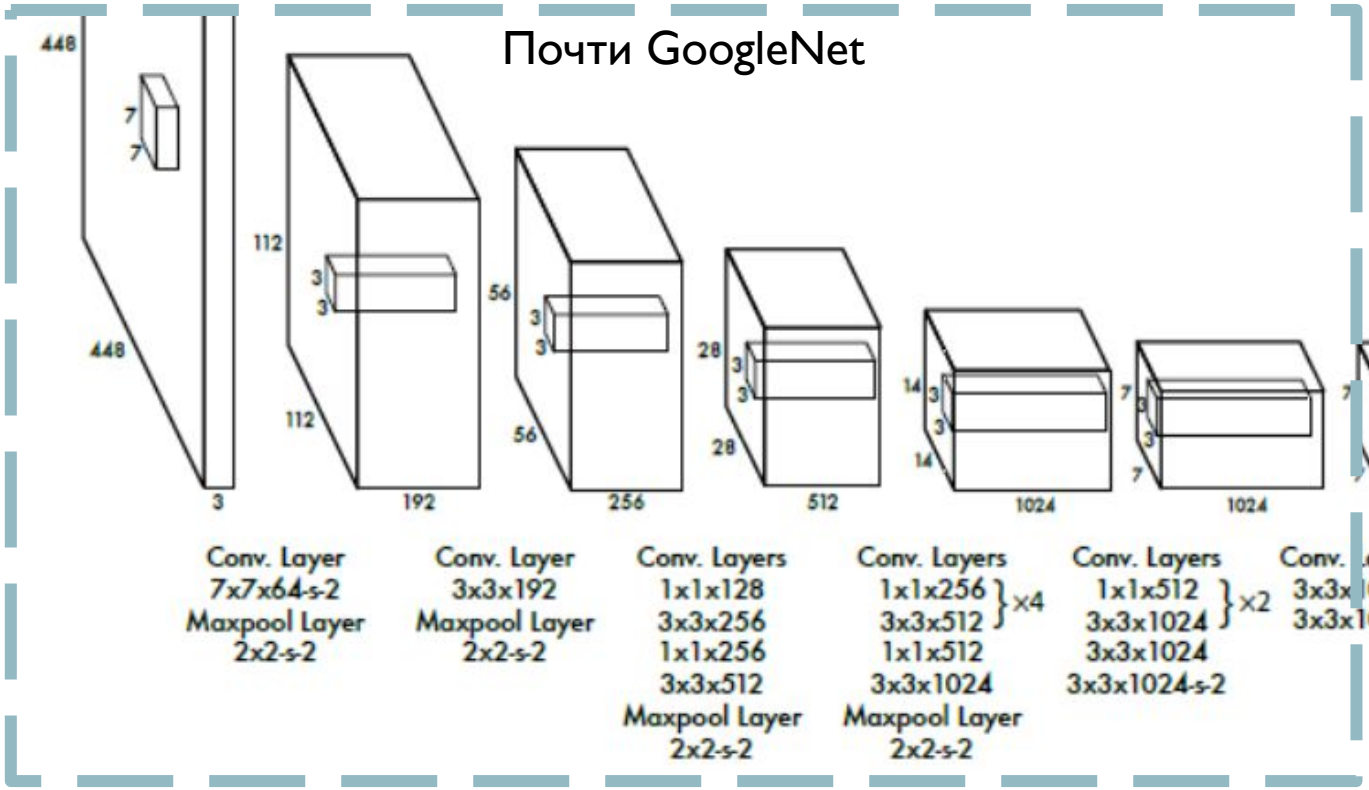
- convolution layer (conv1),
- 8 Fire modules (fire2-9),
- conv layer (conv10)

SSD

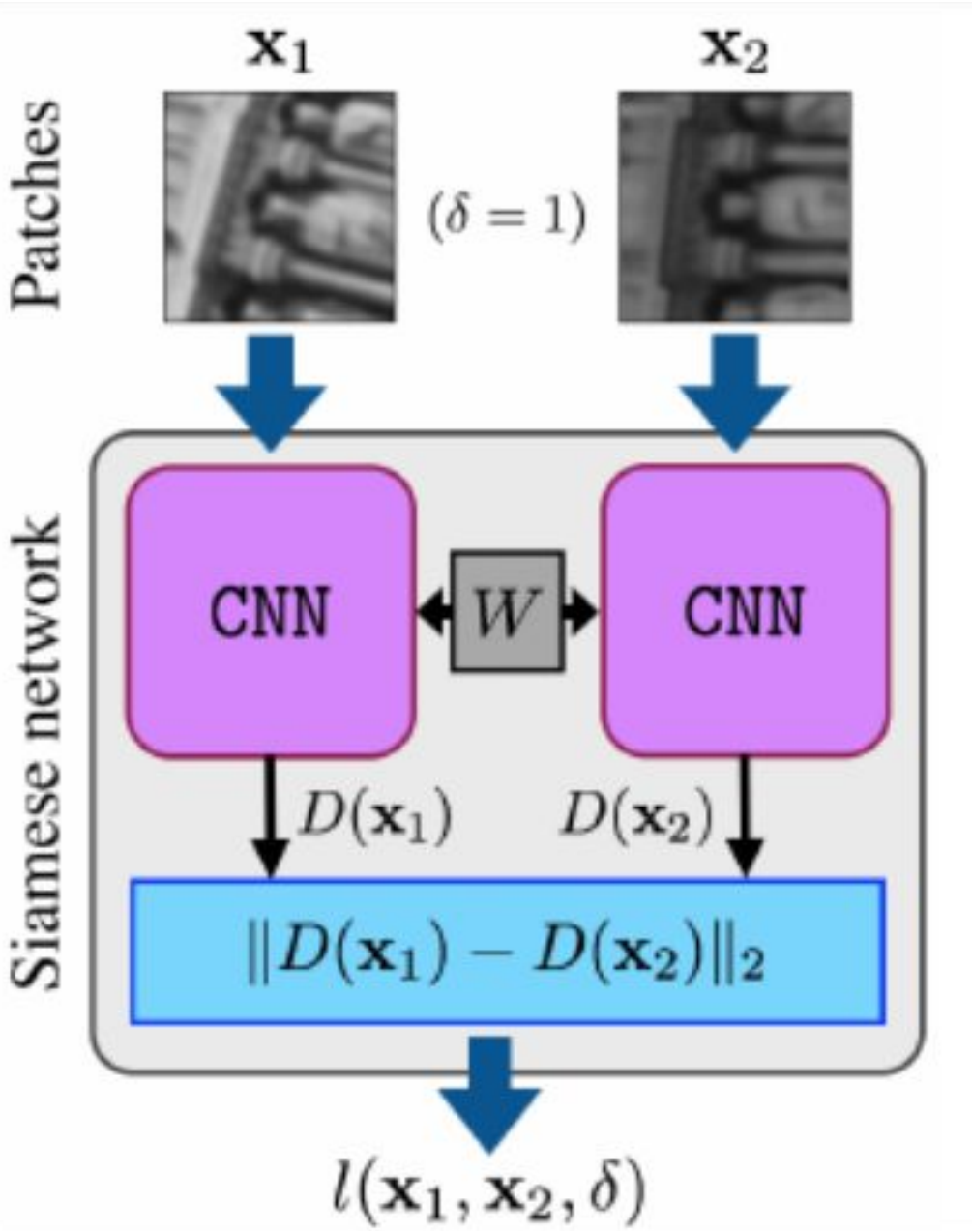

SSD
300
Image
300
3
VGG-16
through Pool5 layer
38
Conv4_3
38
512
Extra Feature Layers
Classifier : Conv: 3x3x(3x(Classes+4))
Classifier : Conv: 3x3x(6x(Classes+4))
19
Conv6
(FC6)
19
1024
19
Conv7
(FC7)
19
1024
10
Conv8_2
10
512
5
Conv9_2
5
256
Conv10_2
3
3
256
Pool 11
1
256
Detections:7308 per Class
Non-Maximum Suppression
72.1mAP
58FPS
Conv: 3x3x1024
Conv: 1x1x1024
Conv: 1x1x256
Conv: 3x3x512-s2
Conv: 1x1x128
Conv: 3x3x256-s2
Conv: 1x1x128
Conv: 3x3x256-s2
Avg Pooling:Global

YOLO

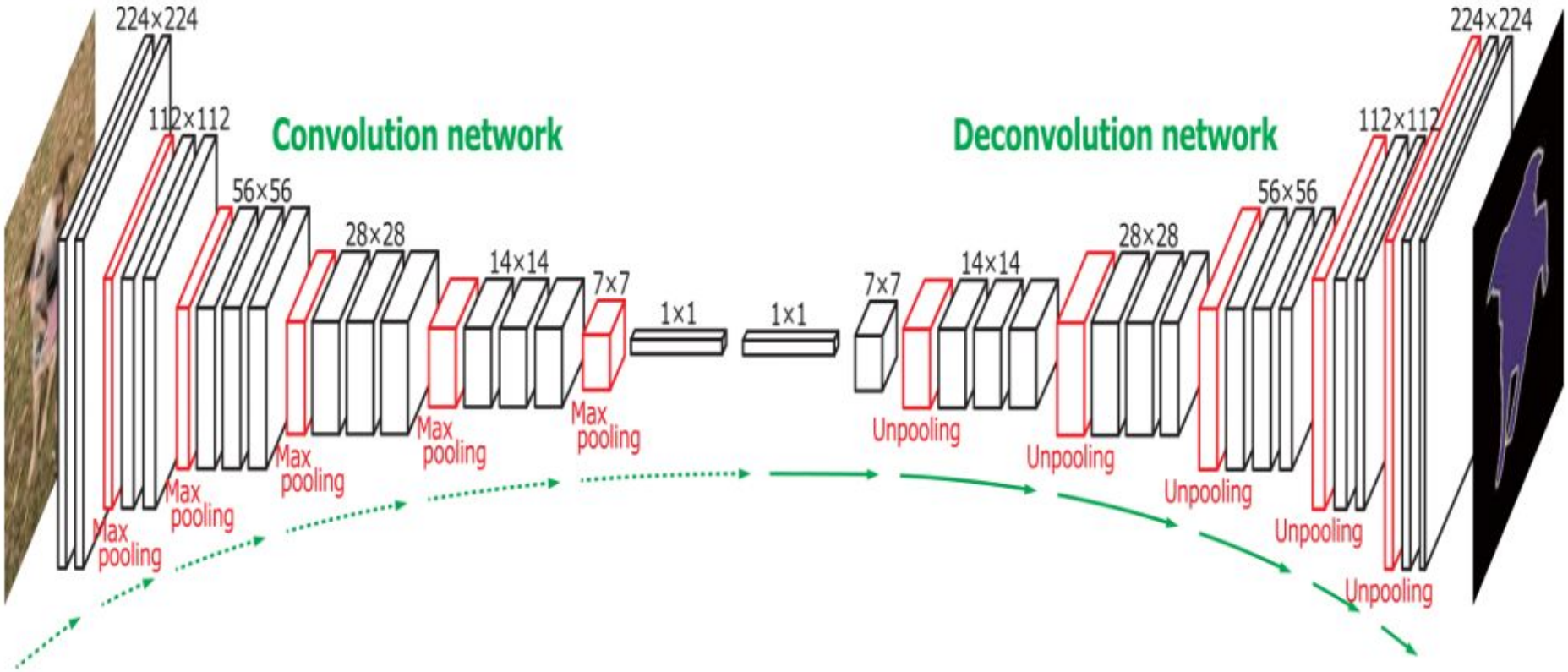
- Вход: 448x448x3



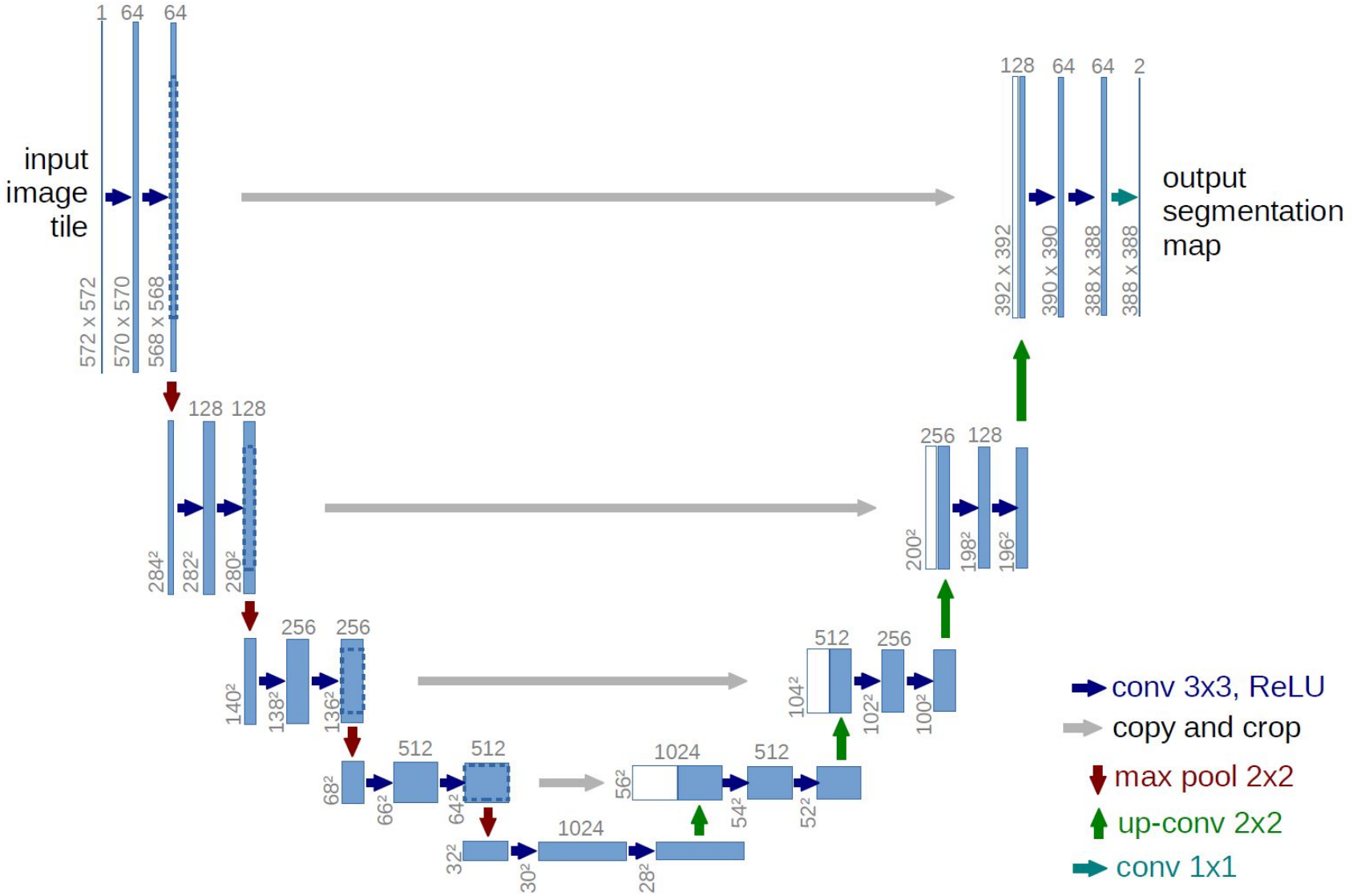
Сиамская сеть



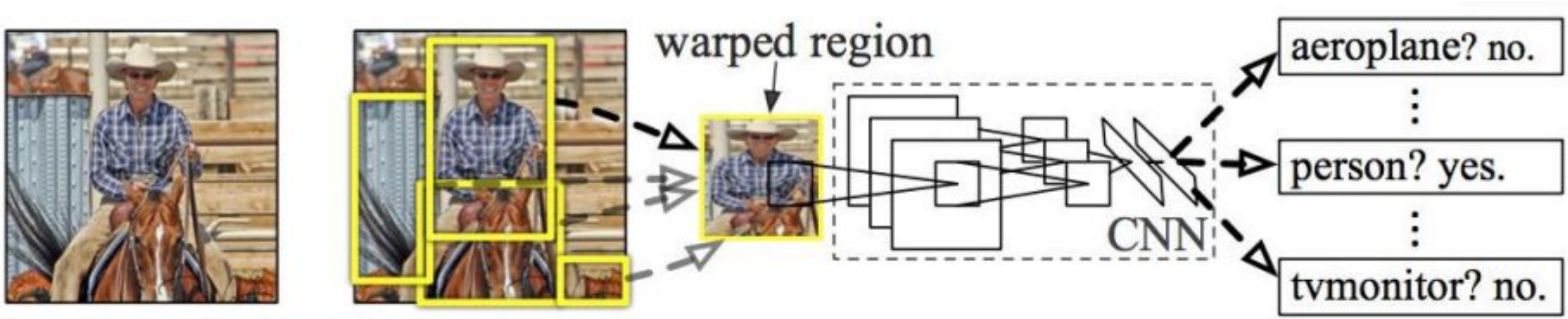
Сегментация и рисование



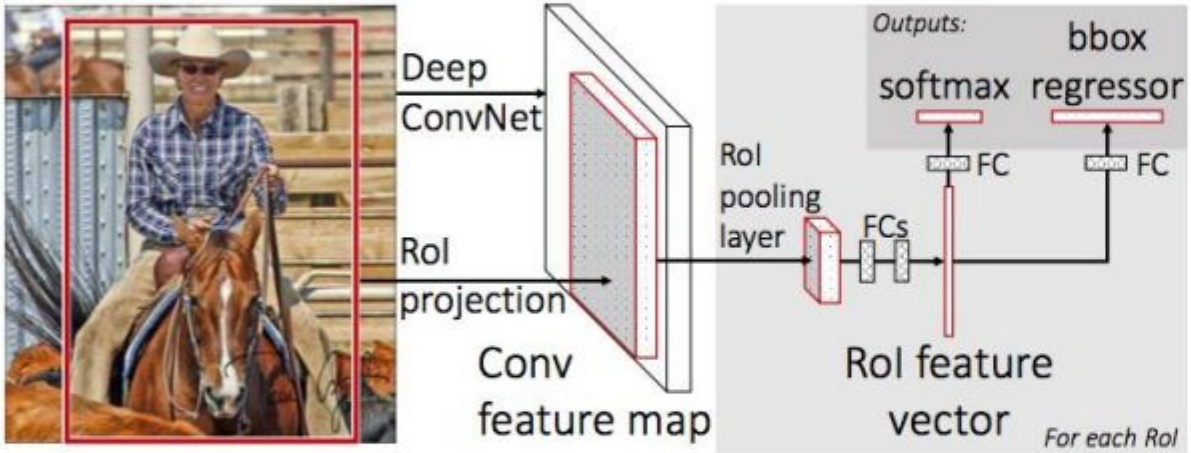
U-net - сегментация



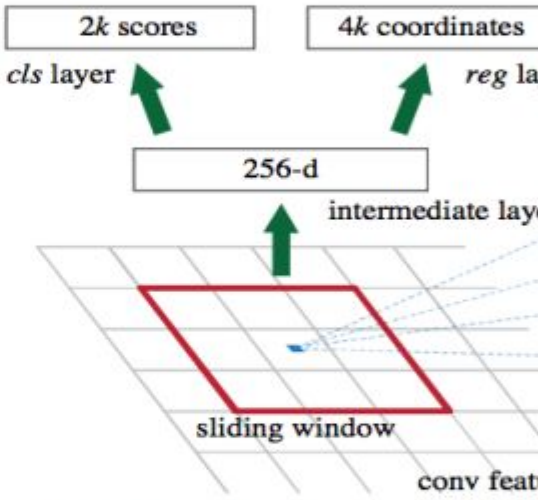
R-CNN (region CNN)



Fast R-CNN



Faster R-CNN with region

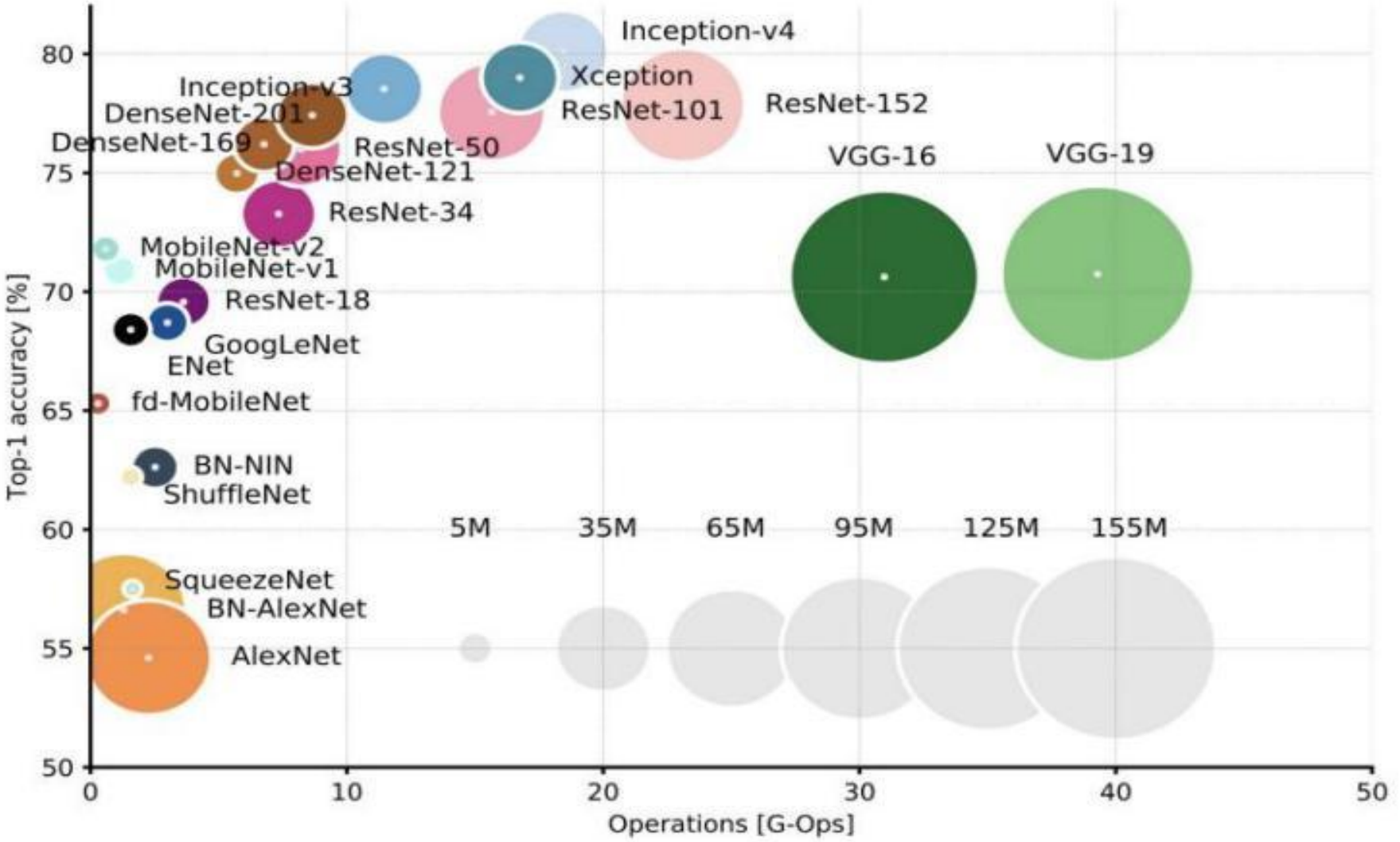


принципы построения DCNN для CV

- Избегайте representational bottlenecks: не стоит резко снижать размерность представления данных, это нужно делать плавно от начала сети и до классификатора на выходе.
- Высокоразмерные представления следует обрабатывать локально, увеличивая размерность: недостаточно плавно снижать размерность, стоит использовать принципы, описанные в предыдущей статье, для анализа и группировки коррелированных участков.
- Пространственные сверки можно и нужно факторизовывать на еще более мелкие: это позволит сэкономить ресурсы и пустить их на увеличение размера сети.
- Необходимо соблюдать баланс между глубиной и шириной сети: не стоит резко увеличивать глубину сети отдельно от ширины, и наоборот; следует равномерно увеличивать или уменьшать обе размерности.

YOLOv2 для генерации изображений

- Сснижающих качество распознавания


Inception-v4
Inception-v3
Xception
DenseNet-201
ResNet-101
ResNet-152
DenseNet-169
ResNet-50
VGG-16
VGG-19
DenseNet-121
ResNet-34
MobileNet-v2
MobileNet-v1
ResNet-18
GoogLeNet
ENet
fd-MobileNet
BN-NIN
ShuffleNet
5M
35M
65M
95M
125M
155M
SqueezeNet
BN-AlexNet
AlexNet
Top-1 accuracy [%]
80
75
70
65
60
55
50
0
10
20
30
40
50
Operations [G-Ops]

Резюме

- https://keras.io/applications/

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
ResNeXt50	96 MB	0.777	0.938	25,097,128	-
ResNeXt101	170 MB	0.787	0.943	44,315,560	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

Как работать

- Основной инструмент – обучение с учителем
- Признаки переносятся на другие задачи!
- На следующей неделе – применения сетей для других задач зрения
- Для обучения свёрток нужны GPU (вывод можно и на CPU
- Есть стандартные библиотеки и модели
- Обучать сети с нуля долго и сложно
- Дообучать (fine-tuning) гораздо проще!
- Полезный трюк – заморозить почти все слои