СОВРЕМЕННЫЕ СВЁРТОЧНЫЕ АРХИТЕКТУРЫ

Сергей Николенко

СПбГУ — Санкт-Петербург

18 сентября 2026 г.

Random facts:

- 18 сентября 1066 г. Харальд Гардерада вместе с Тостигом Годвинсоном высадились в устье реки Амбер и начали не самое удачное вторжение в Англию
- 18 сентября 1940 г. четыре подростка случайно наткнулись на узкое отверстие, образовавшееся после падения сосны, в которую попала молния; отверстие вело в пещеру Ласко
- 18 сентября 1911 г. в результате покушения в Киевском оперном театре погиб Пётр Столыпин
- 18 сентября — День памяти погибших мотоциклистов и Всемирный день мониторинга качества воды
- 18 сентября — важная дата для морских сражений Петра I: в 1714 г. в этот день произошло Гангутское сражение, а в 1720 — сражение при Гренгаме
- 18 сентября 1934 г. СССР вошёл в состав Лиги наций; хватило лет на пять

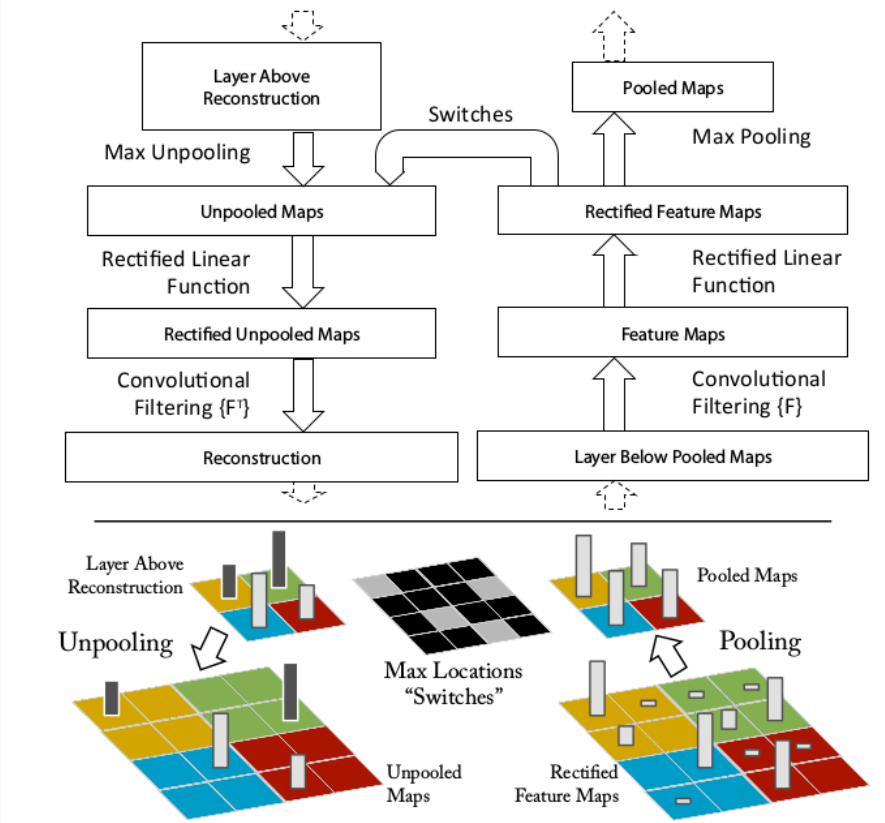
Краткое резюме предыдущей серии

- В прошлый раз мы дошли от зрительной коры до AlexNet: свёртка, пулинг, ReLU, дропаут и аугментация — и первая победа на ImageNet.
- Но AlexNet мы обучили как чёрный ящик. Что там внутри?
- Сегодня сначала заглянем внутрь обученной сети через деконволюцию, а потом разберём зоопарк архитектур, который из этого вырос: VGG, Inception, ResNet и далее.

Что видят свёрточные сети

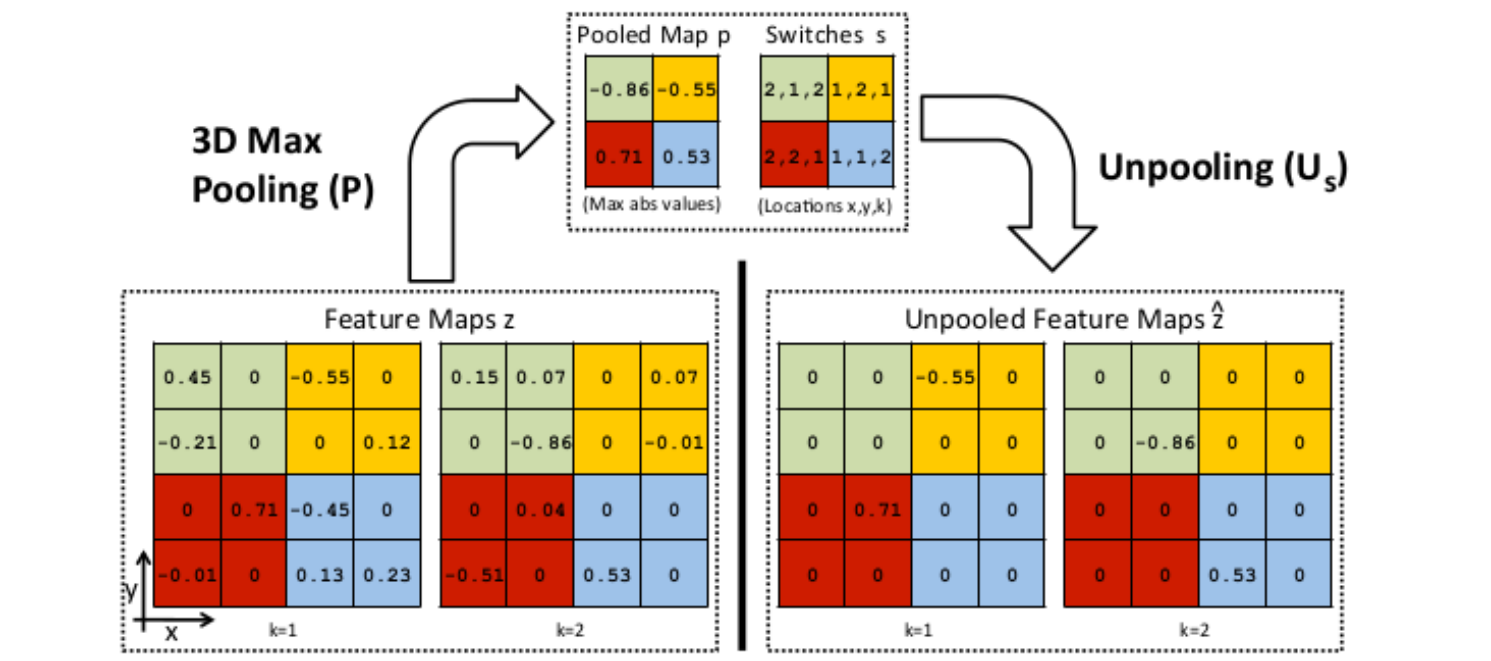
Деконволюция

- (Zeiler et al., 2011; Zeiler, Fergus, 2014): *деконволюция* как обратная свёртке операция.
- Чтобы обратить свёртку, достаточно транспонировать.



Деконволюция

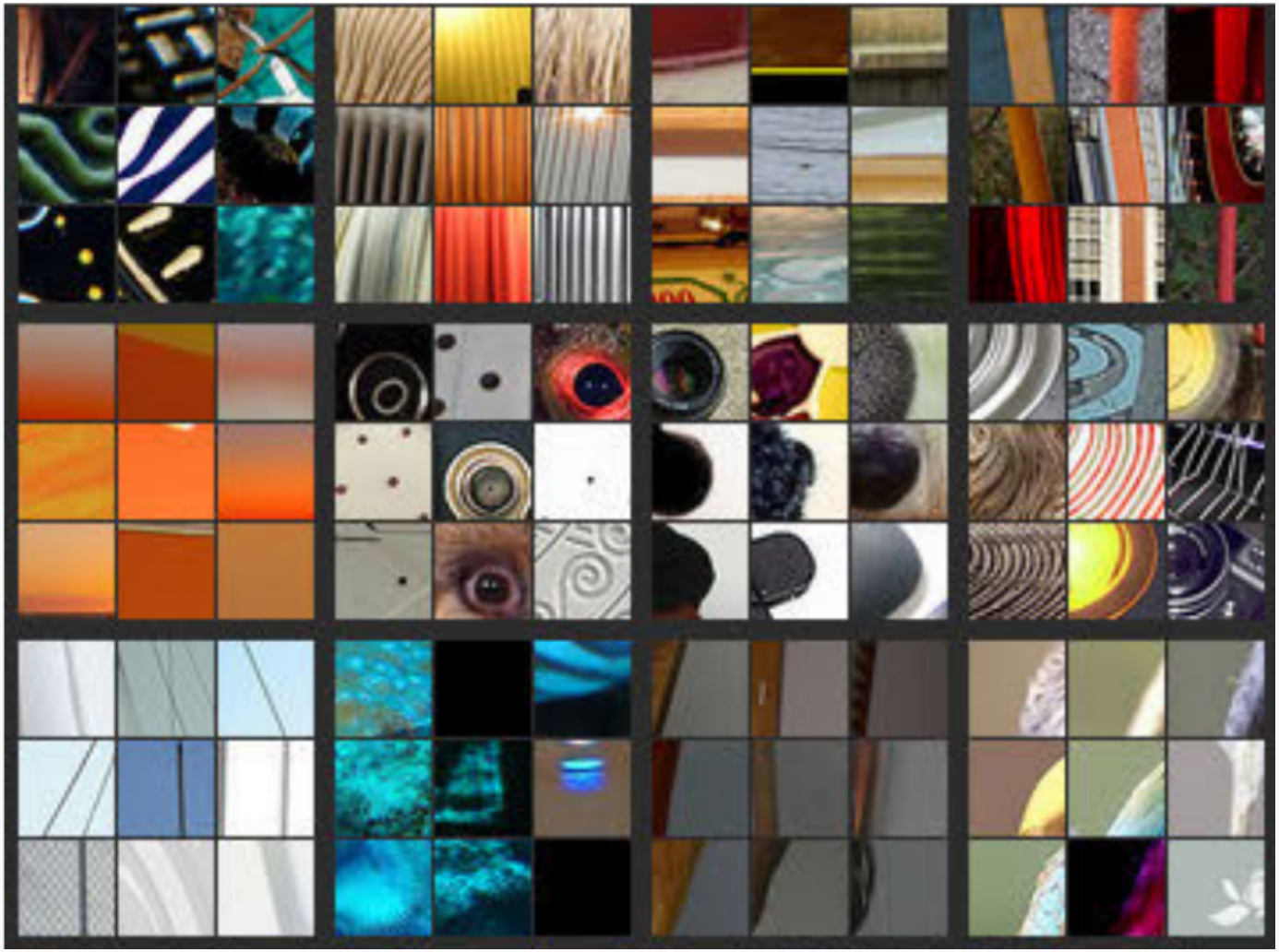
- Интересный вопрос – как обратить субдискретизацию.
- Она, конечно, необратима, но можно запомнить положение максимумов и обратить примерно.



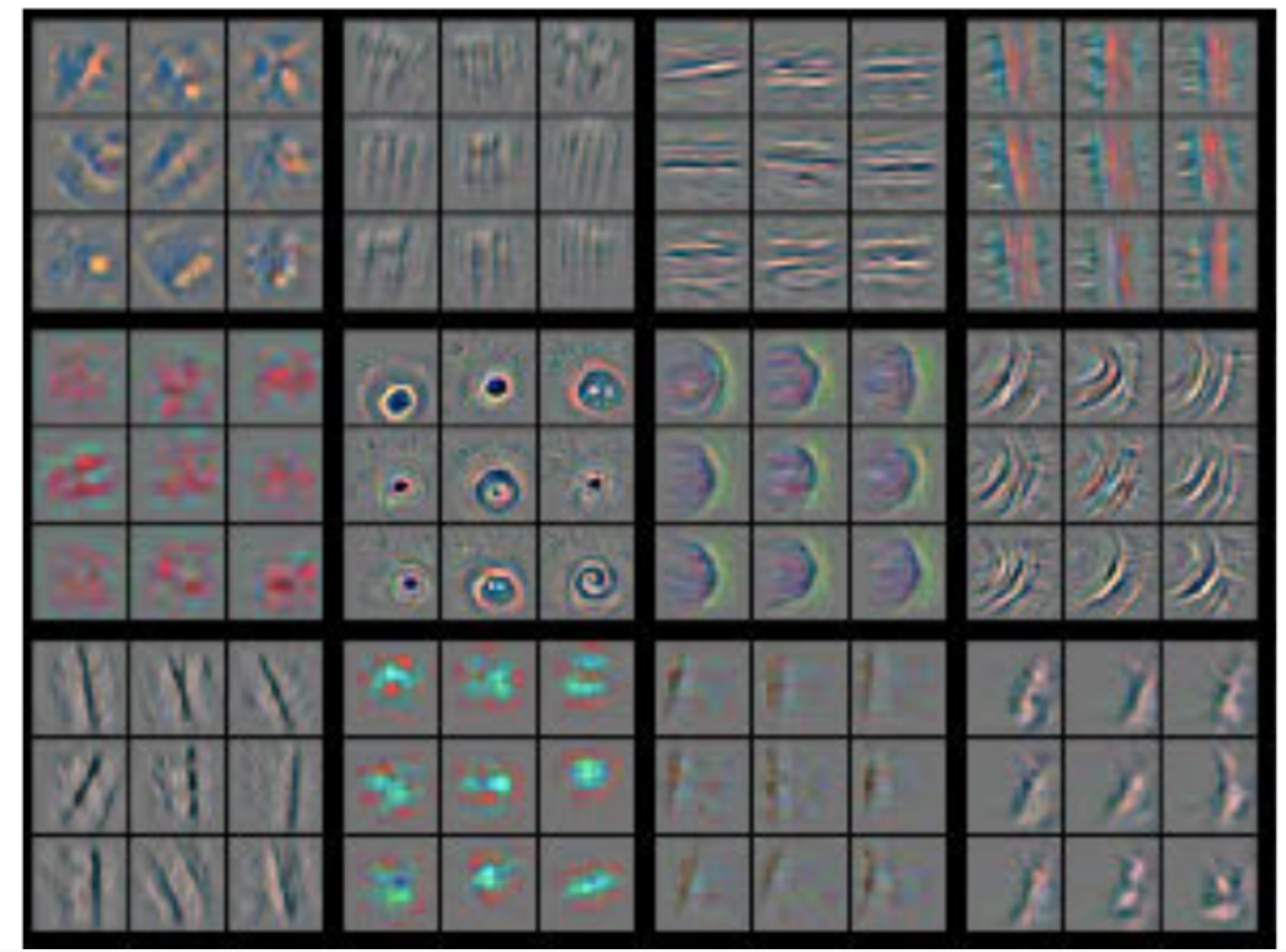
Уровень 1

(Zeiler, Fergus, 2014): визуализируем признаки через деконволюцию. Это сеть, обученная на ImageNet.

(Zeiler, Fergus, 2014): уровень 2

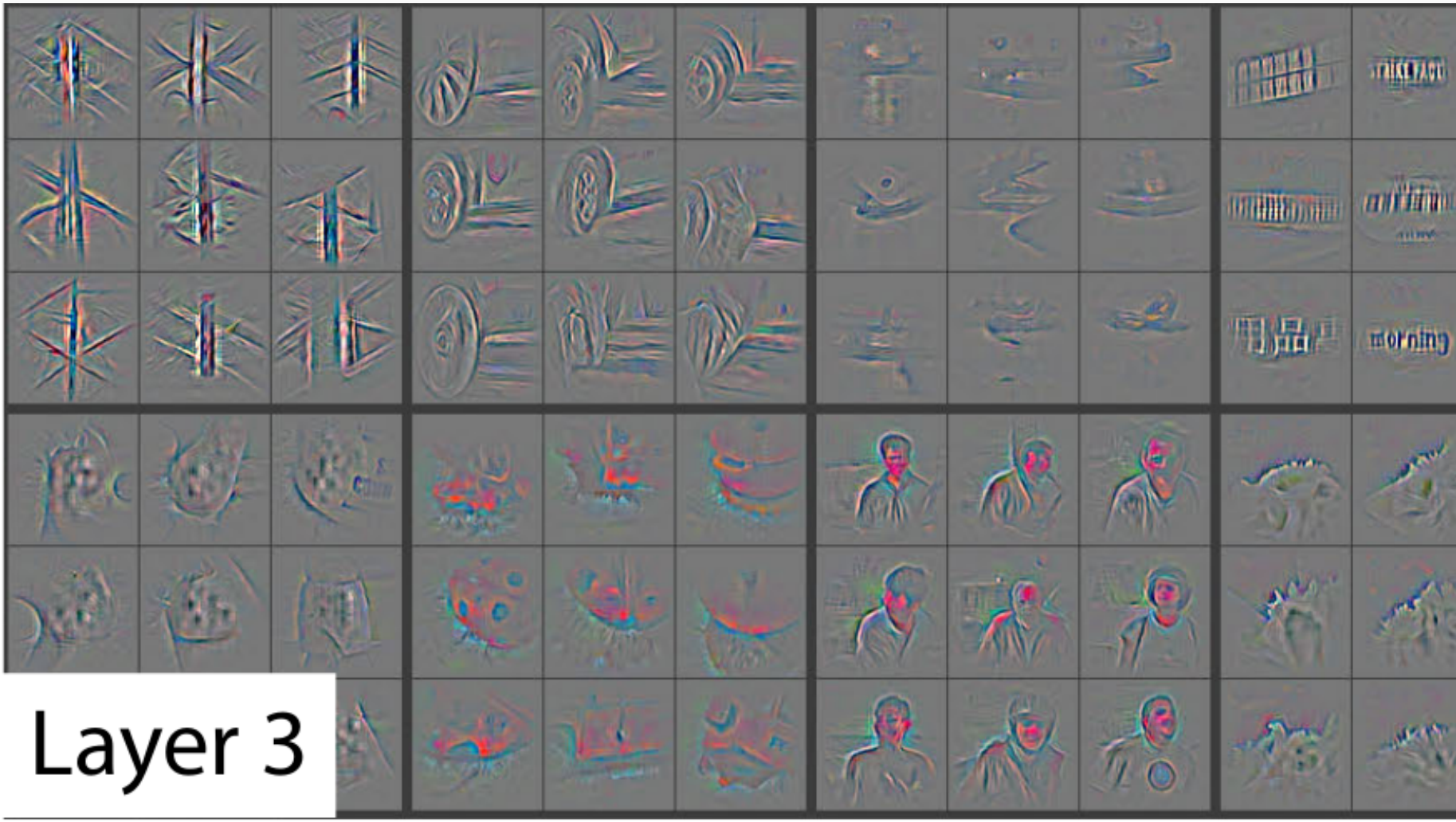


(Zeiler, Fergus, 2014): уровень 2

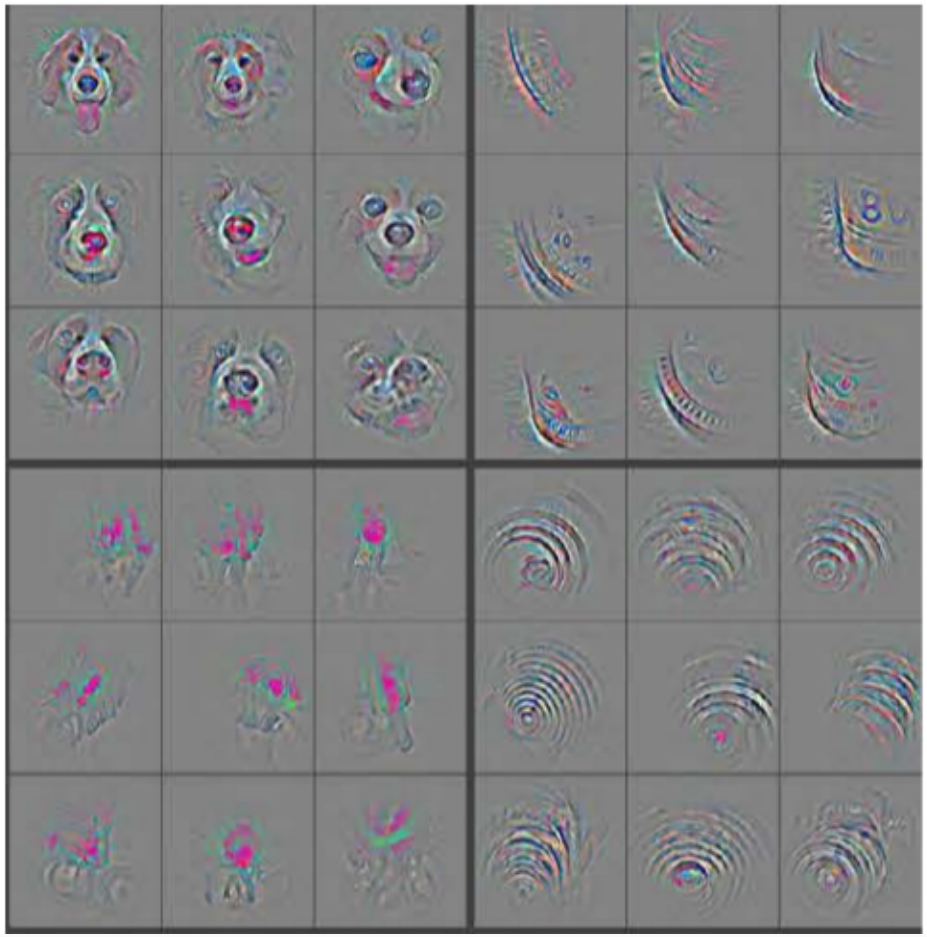


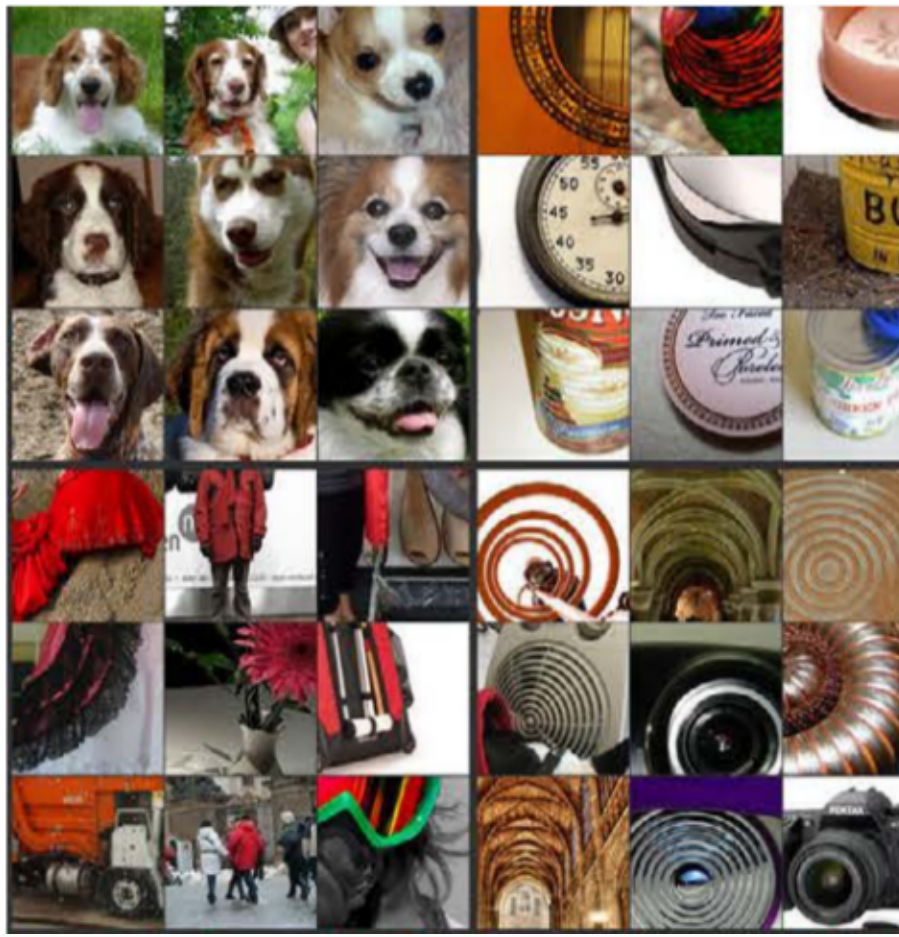
(Zeiler, Fergus, 2014): уровень 3

(Zeiler, Fergus, 2014): уровень 3

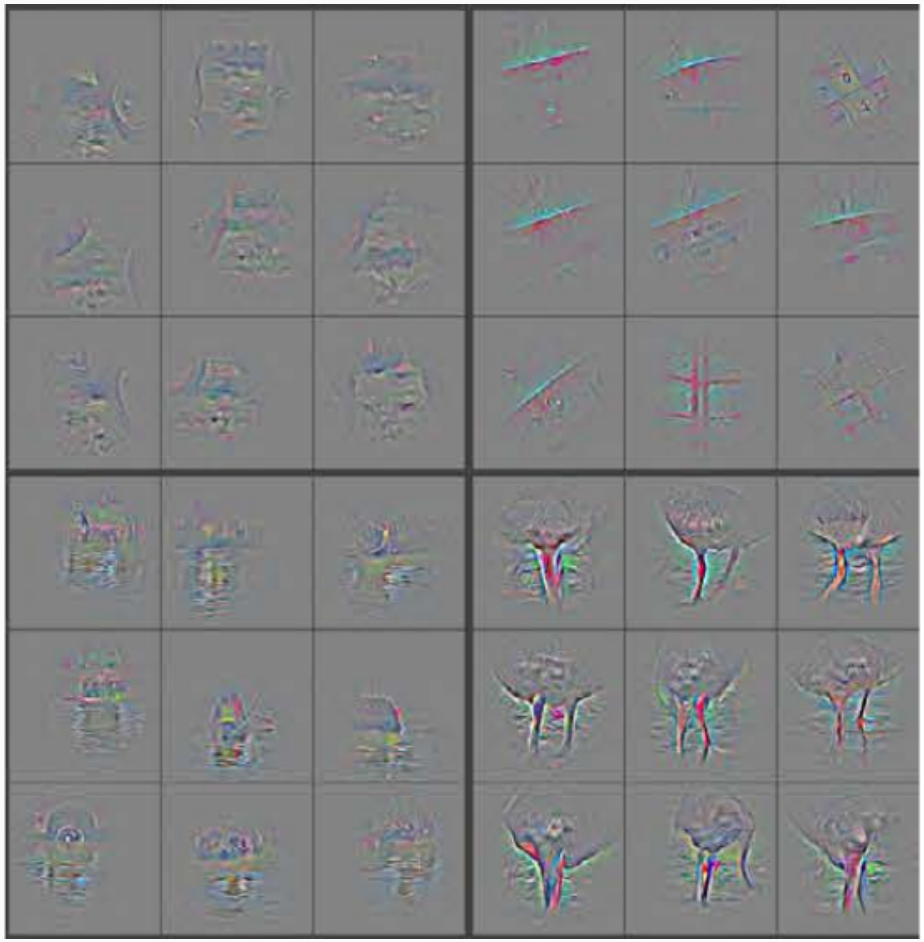


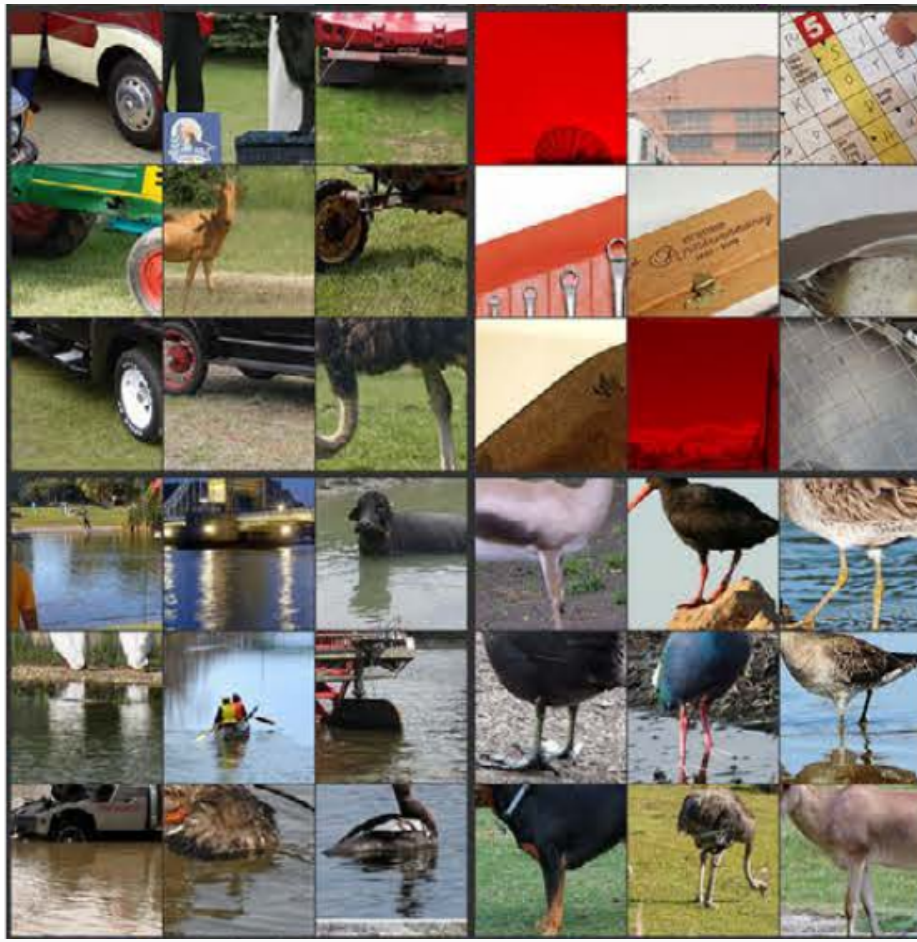
(Zeiler, Fergus, 2014): уровень 4



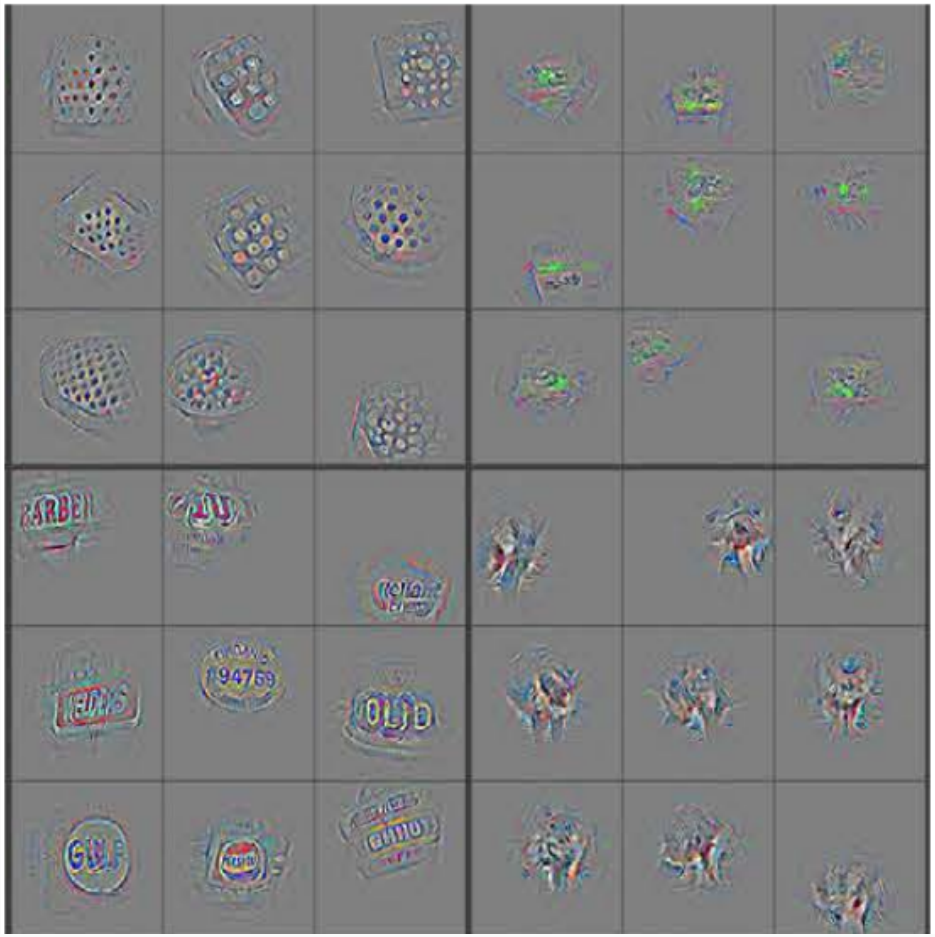


(Zeiler, Fergus, 2014): уровень 4

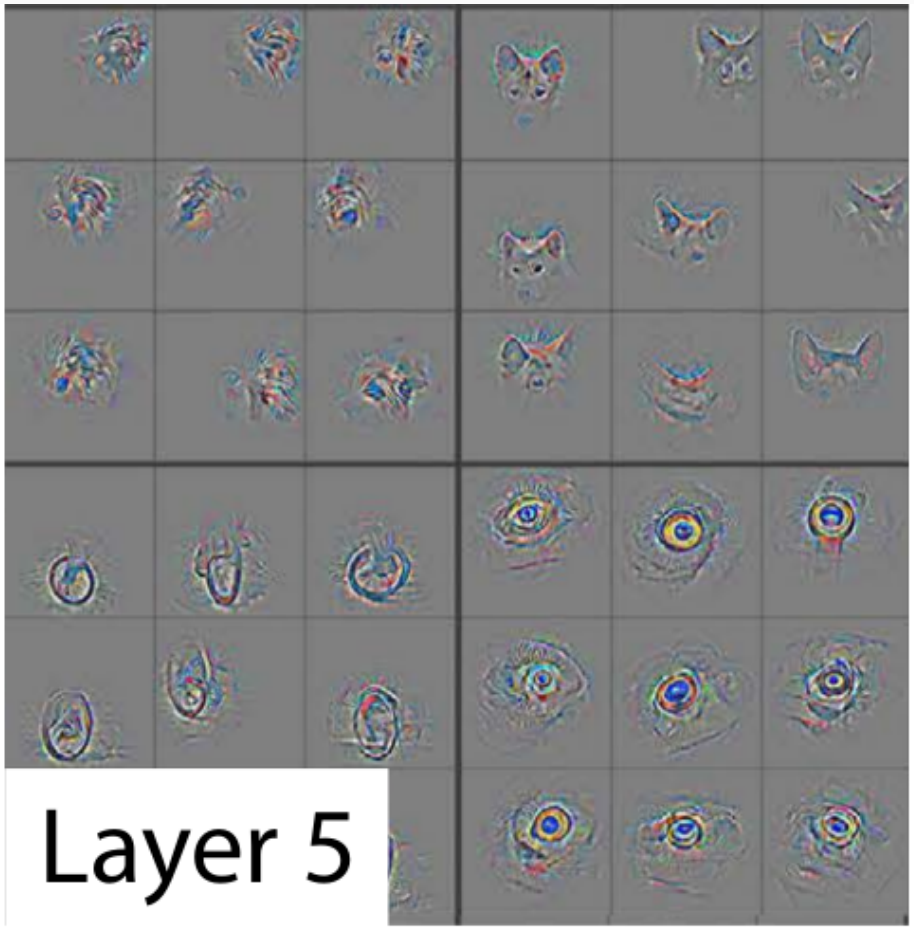




(Zeiler, Fergus, 2014): уровень 5



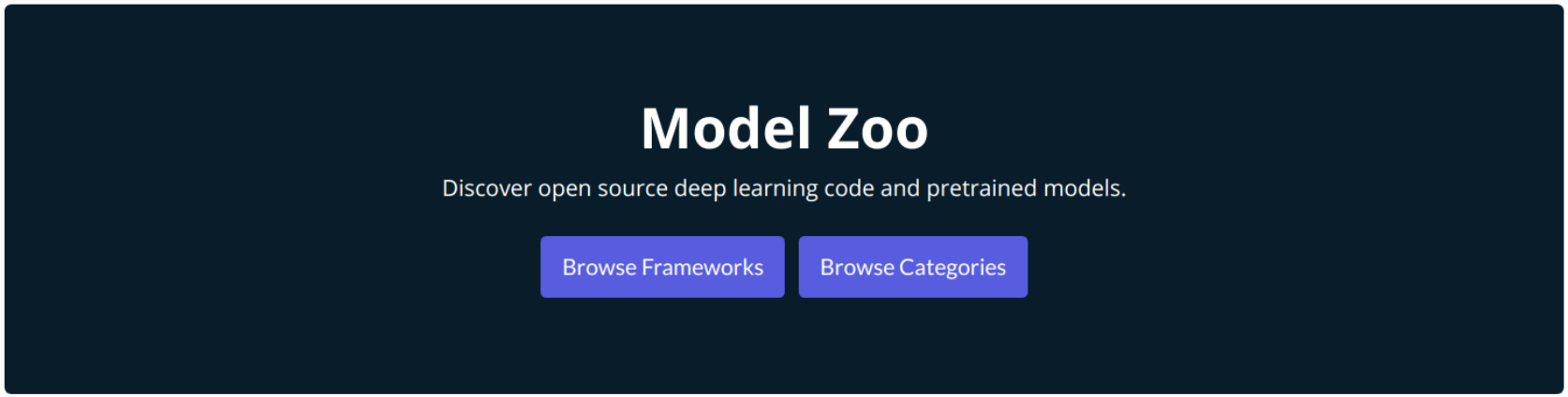
(Zeiler, Fergus, 2014): уровень 5



Современные свёрточные архитектуры

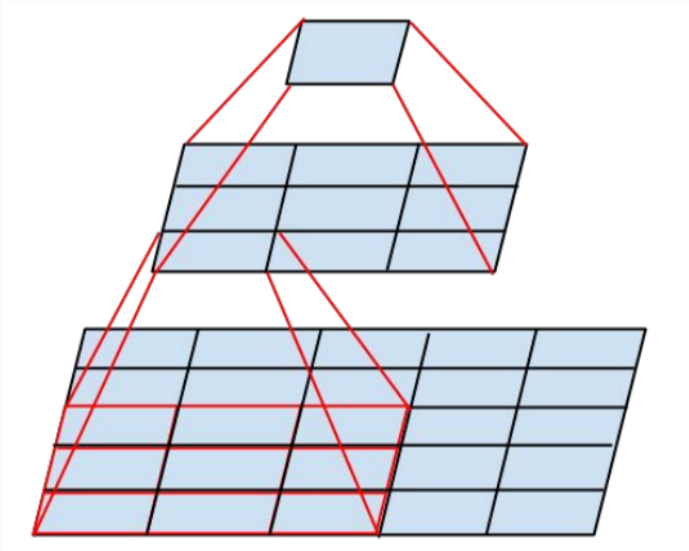
Зоопарк

- Разных моделей, использующих CNN, очень много, и всё время появляются новые.
- Всё можно скачать, часто даже уже веса, предобученные на стандартных датасетах, есть.
- Но обычно они основаны на нескольких относительно стандартных идеях. Их-то мы и рассмотрим.



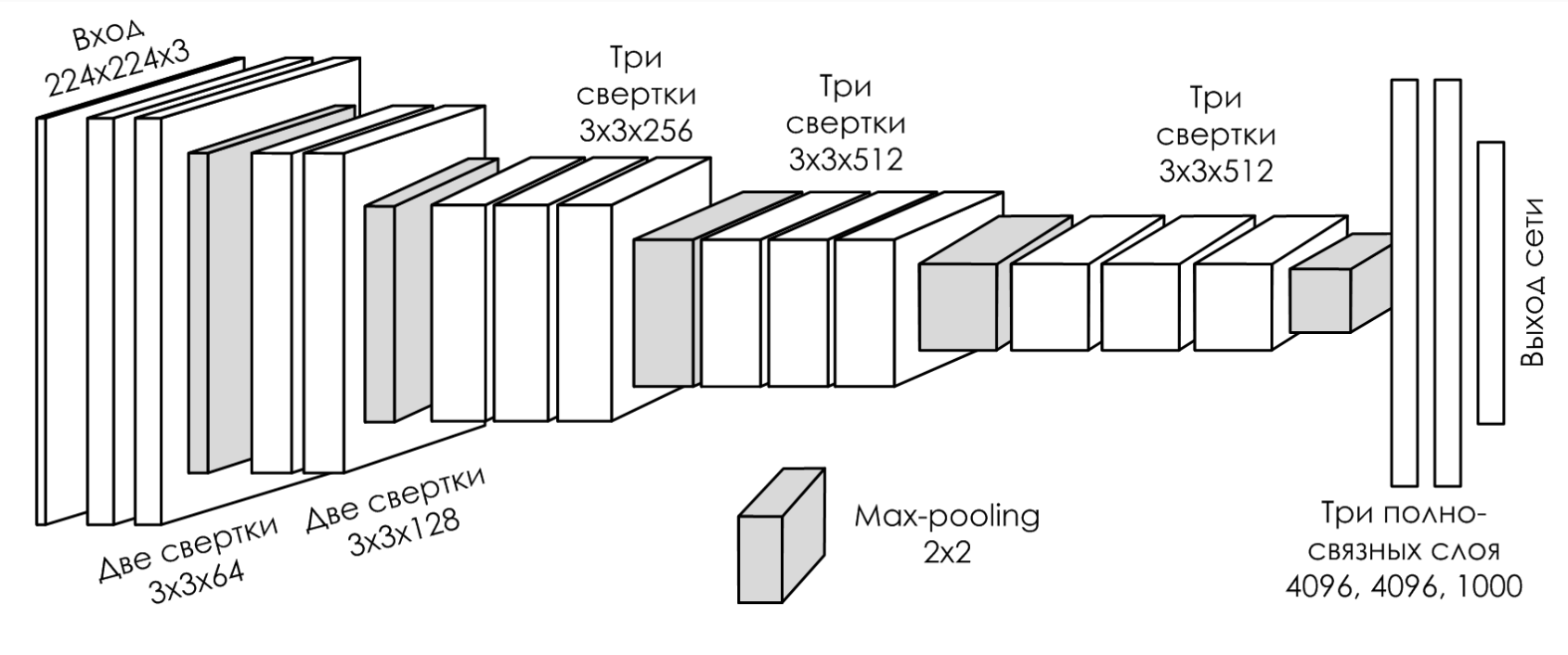
VGG

- VGG (Oxford Visual Geometry Group): давайте представлять большие свёртки как комбинации свёрток 3×3.
- Это уменьшает число весов и делает сеть глубже.



VGG

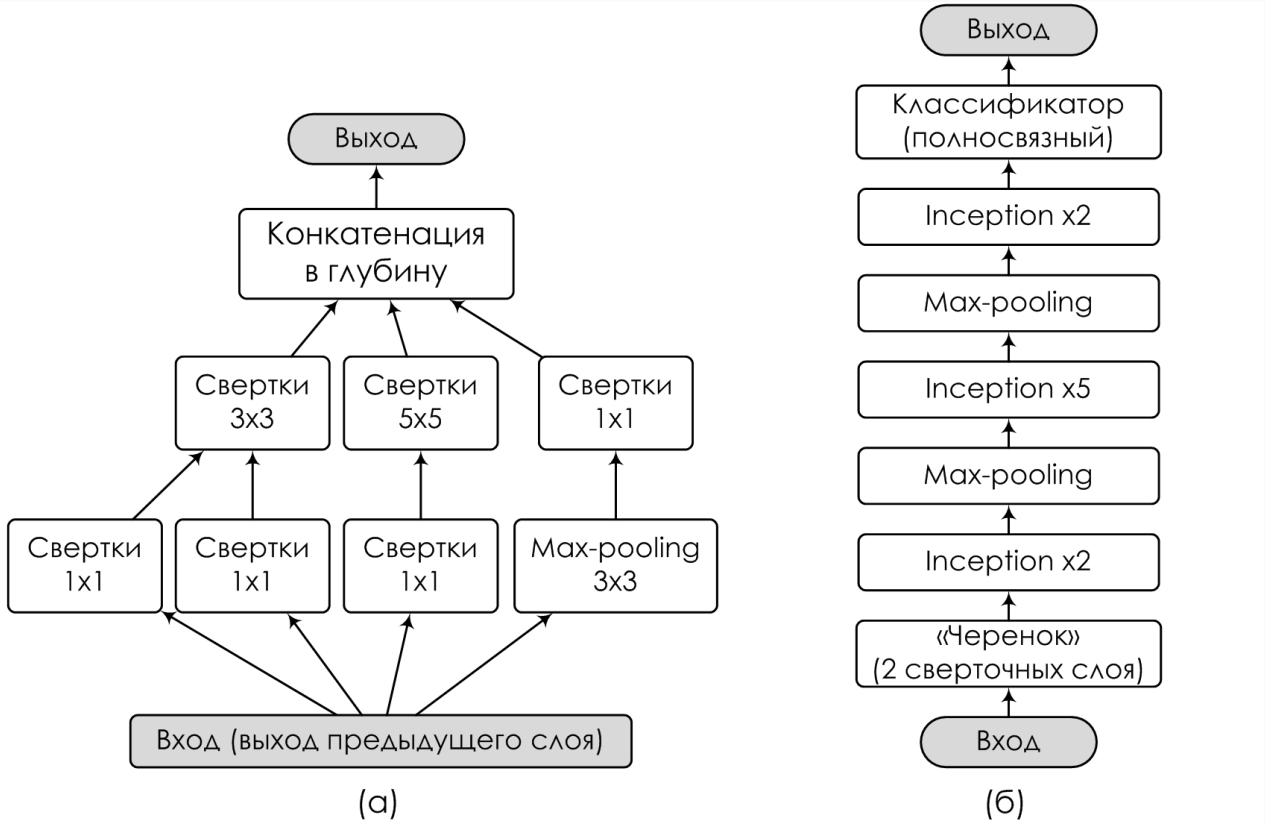
- Сеть VGG, использованная в ImageNet Large Scale Visual Recognition Competition (ILSVRC-2014).



- А сейчас NVIDIA специально оптимизирует GPU для свёрток 3×3.

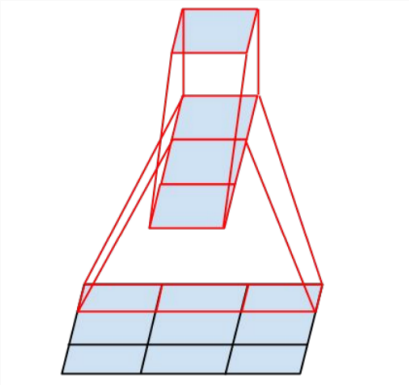
Inception

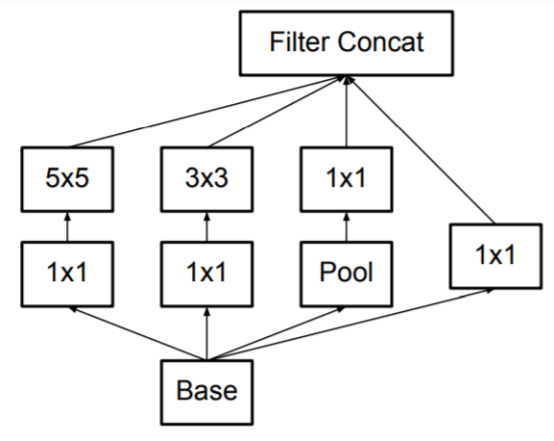
- *Inception*, разработанная командой *GoogLeNet*: давайте используем идею “сеть в сети” (network in network), чтобы сократить веса ещё больше!
- И дополнительные классификаторы (просто лишние слагаемые в целевую функцию).

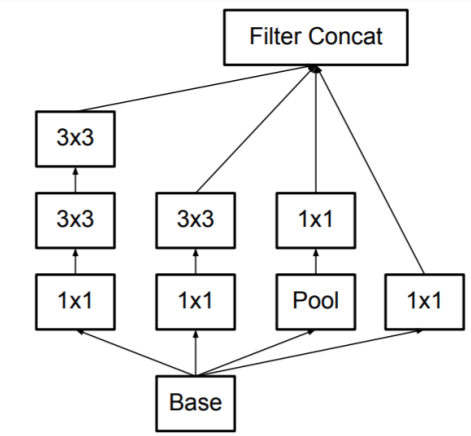


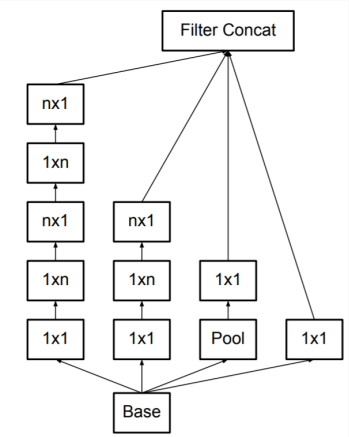
Inception

- Во второй версии (Szegedy et al., 2015) свёртки $n \times n$ заменили на комбинации свёрток $n \times 1$ и $1 \times n$:



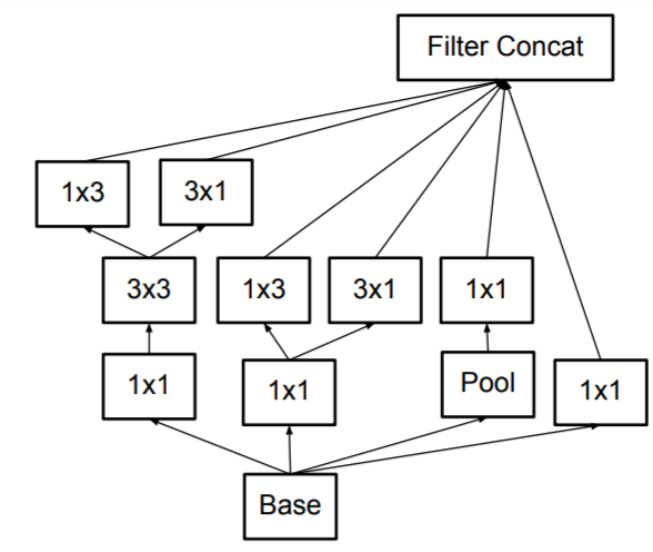






Inception

- А банки фильтров в Inception v2 сделали широкими, а не глубокими:



- В той же статье – Inception v3; то же самое, но ещё RMSProp, факторизованные свёртки 7×7, batchnorm во вспомогательных классификаторах и...

Inception

- Label smoothing:
  - в обычном классификаторе мы обучаем результаты softmax с целевой переменной $q(k) = [k = y]$, т.е. аргументы softmax хотят расти неограниченно;
  - модели становятся слишком уверены в себе, оверфиттинг;
  - решение: давайте вместо этого оптимизировать

$$q'(k) = (1 - \epsilon)[k = y] + \epsilon u(k)$$

для какого-то априорного $u(k)$, например $u(k) = \frac{1}{K}$.

Network	Crops Evaluated	Top-5 Error	Top-1 Error
GoogLeNet [20]	10	-	9.15%
GoogLeNet [20]	144	-	7.89%
VGG [18]	-	24.4%	6.8%
BN-Inception [7]	144	22%	5.82%
PReLU [6]	10	24.27%	7.38%
PReLU [6]	-	21.59%	5.71%
Inception-v3	12	19.47%	4.48%
Inception-v3	144	**18.77%**	**4.2%**

ResNet

- Остаточное обучение (residual learning): давайте обучим разности между очередным уровнем и предыдущим.
- Тогда градиенты смогут беспрепятственно проходить куда надо.
- Функция, реализуемая остаточным блоком, выглядит как

$$\mathbf{y}^{(k)} = F(\mathbf{x}^{(k)}) + \mathbf{x}^{(k)},$$

  где $\mathbf{x}^{(k)}$ — входной вектор слоя k, $F(x)$ — функция, которую вычисляет слой нейронов, а $\mathbf{y}^{(k)}$ — выход остаточного блока, который потом станет входом следующего слоя $\mathbf{x}^{(k+1)}$.
- И градиент будет проходить через этот блок беспрепятственно и не будет затухать:

$$\frac{\partial \mathbf{y}^{(k)}}{\partial \mathbf{x}^{(k)}} = 1 + \frac{\partial F(\mathbf{x}^{(k)})}{\partial \mathbf{x}^{(k)}}.$$

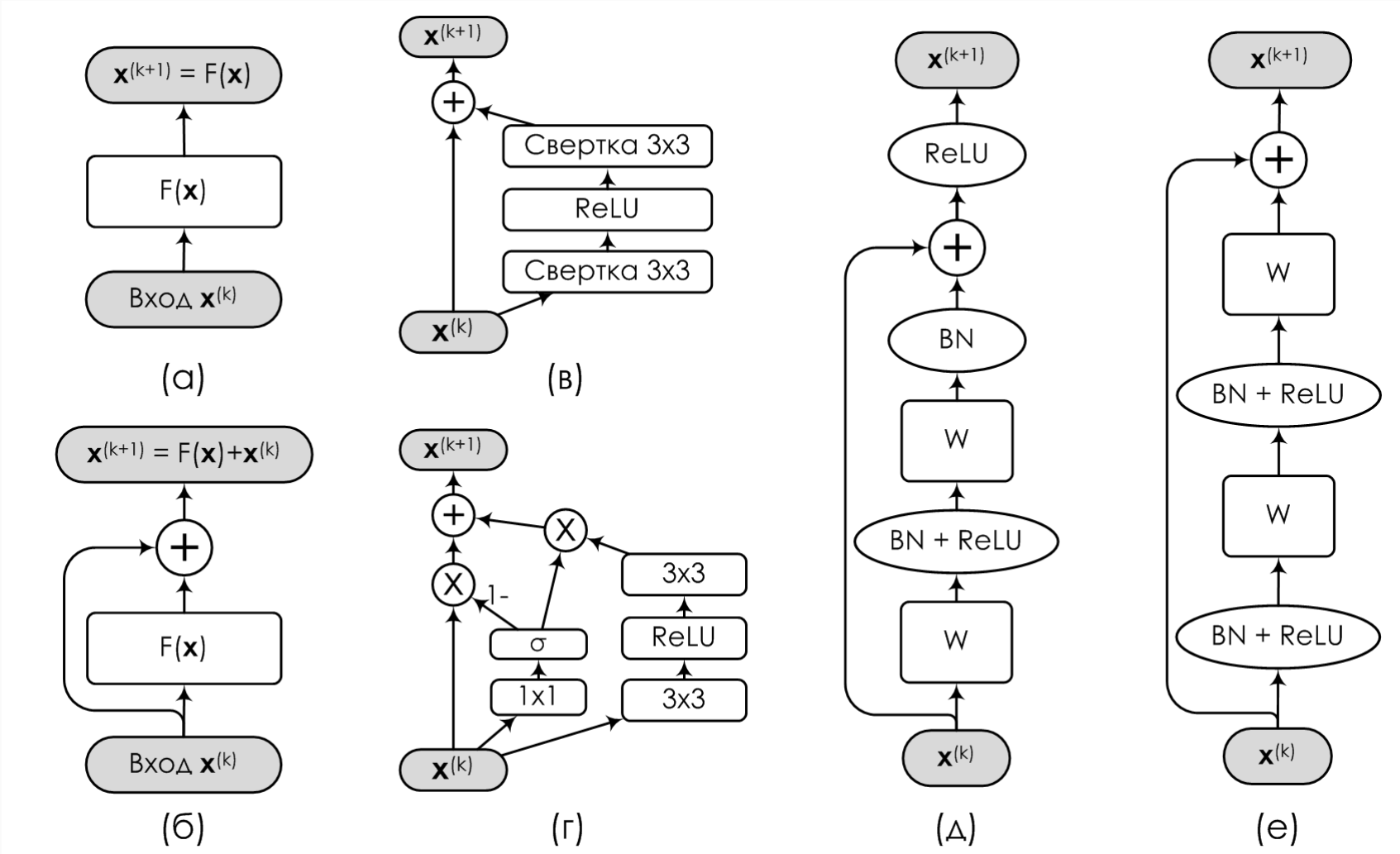
ResNet

- Это привело к *очень* глубоким сетям.
- Другой аналогичный подход – *highway networks* (магистральные сети) Шмидхубера.
- Тоже представляем $\mathbf{y}^{(k)}$, выход слоя k, как линейную комбинацию входа этого слоя $\mathbf{x}^{(k)}$ и результата $F(\mathbf{x}^{(k)})$, веса которой управляются другими преобразованиями:

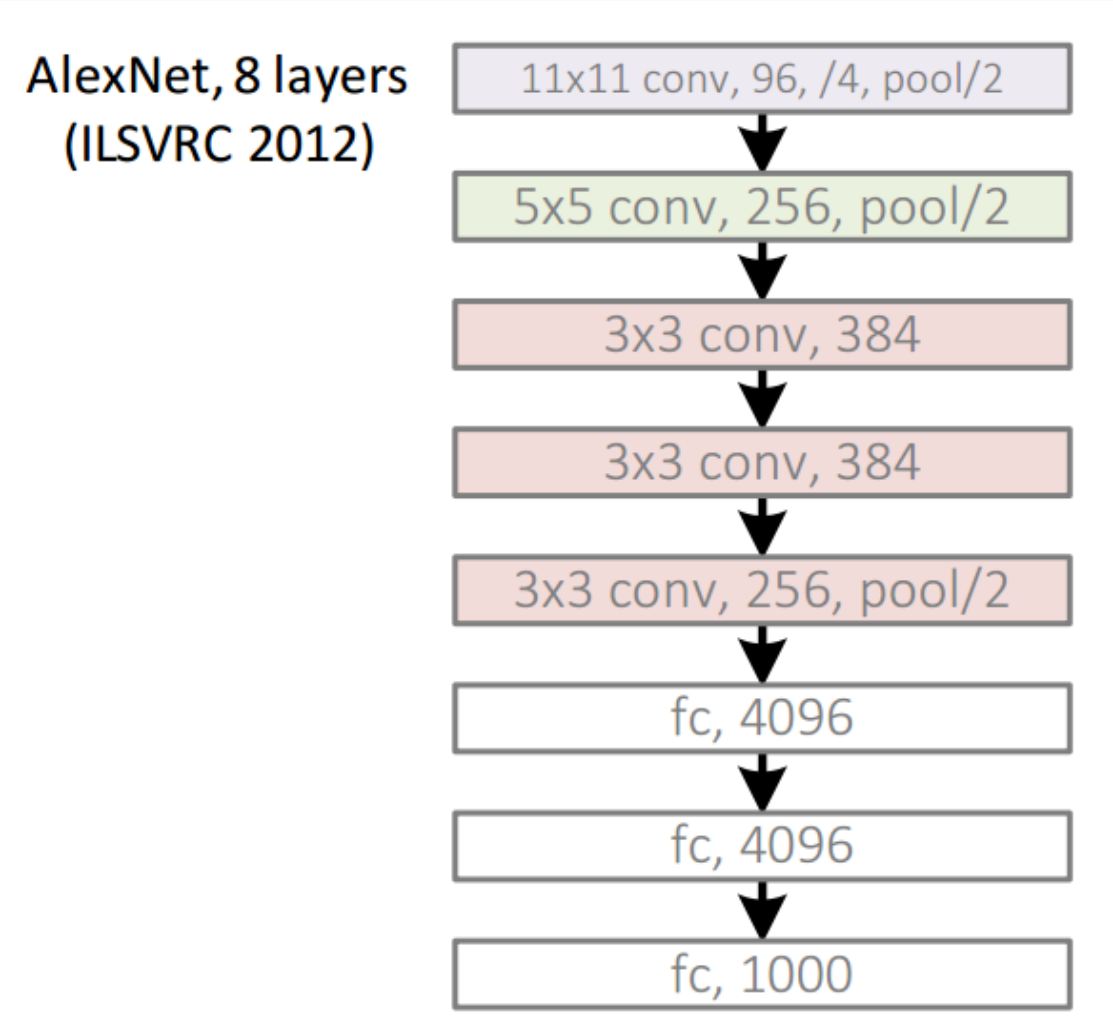
$$\mathbf{y}^{(k)} = C(\mathbf{x}^{(k)})\mathbf{x}^{(k)} + T(\mathbf{x}^{(k)})F(\mathbf{x}^{(k)}),$$

  где C — это гейт переноса (carry gate), а T — гейт преобразования (transform gate); обычно комбинацию делают выпуклой, $C = 1 - T$.
- Практика показывает, что чем «прямее», тем лучше.

ResNet: варианты

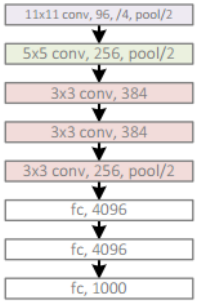


Revolution of Depth (Kaiming He)

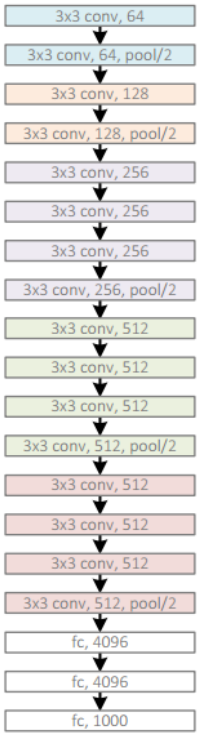


Revolution of Depth (Kaiming He)

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)

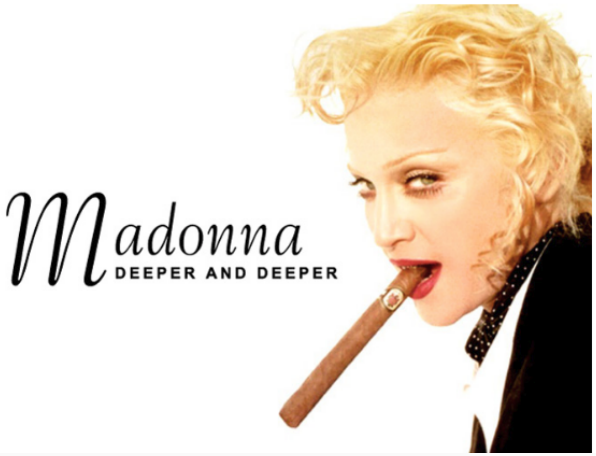
Revolution of Depth (Kaiming He)

ResNet led to the revolution of depth

AlexNet, 8 layers
(ILSVRC 2012)

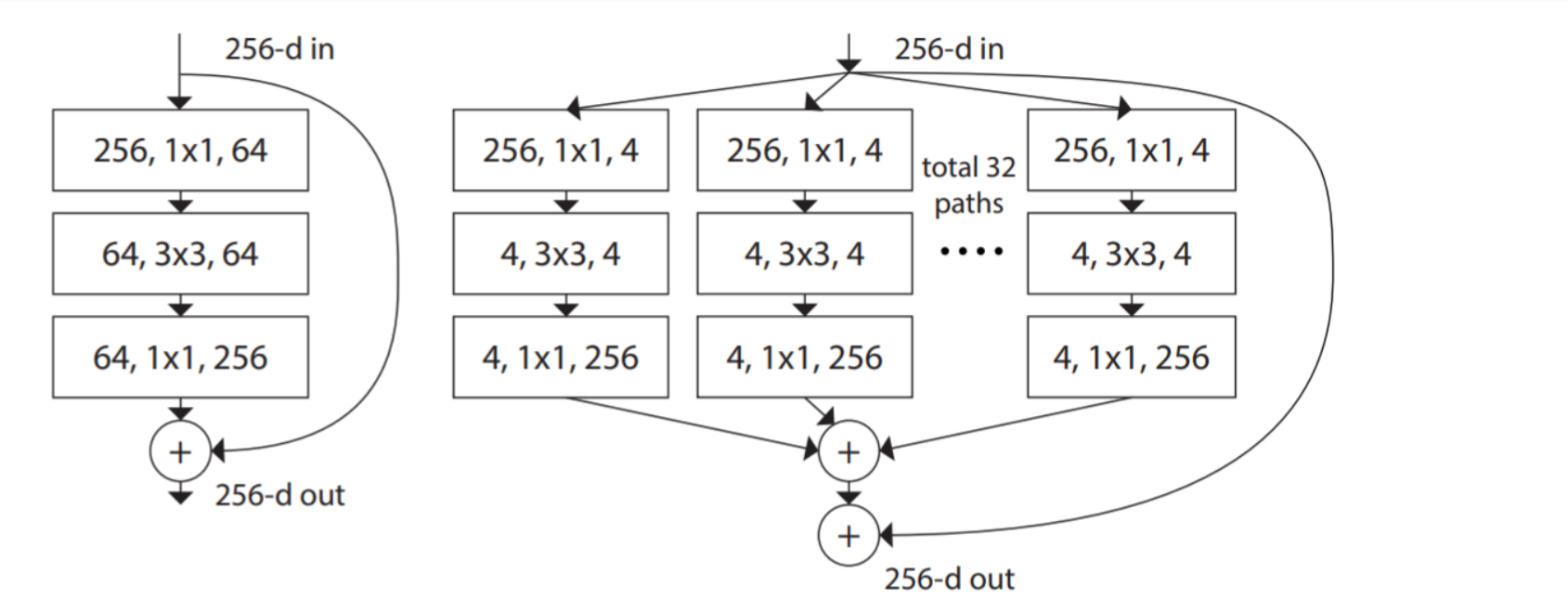
VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)



ResNeXt

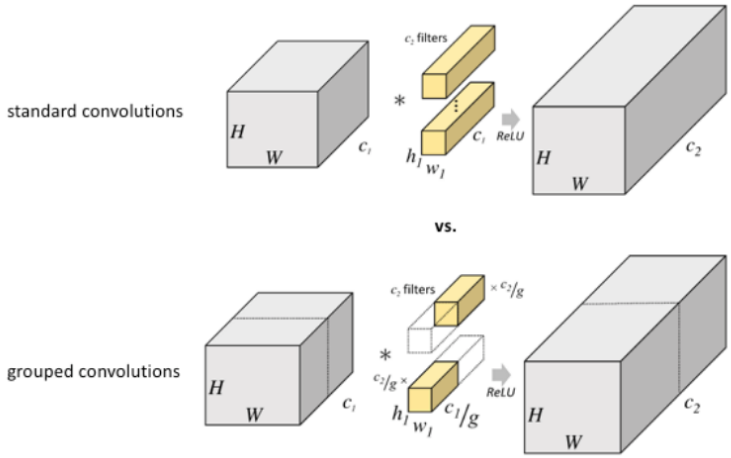
- ResNeXt (Xie et al., 2016): давайте заменим ResNet-блоки на ”split-transform-merge” блоки, похожие на Inception.



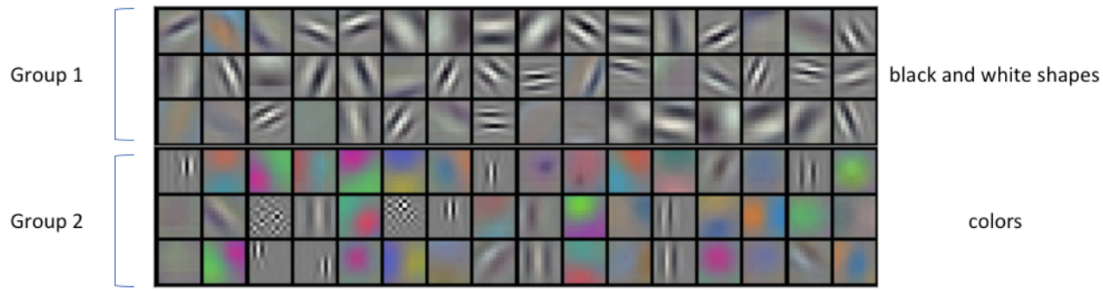
- Вход разбивается на блоки по каналам, к каждому блоку свои свёртки.

ResNeXt

- Идея похожа на group convolutions, которые были ещё в AlexNet для параллелизации:

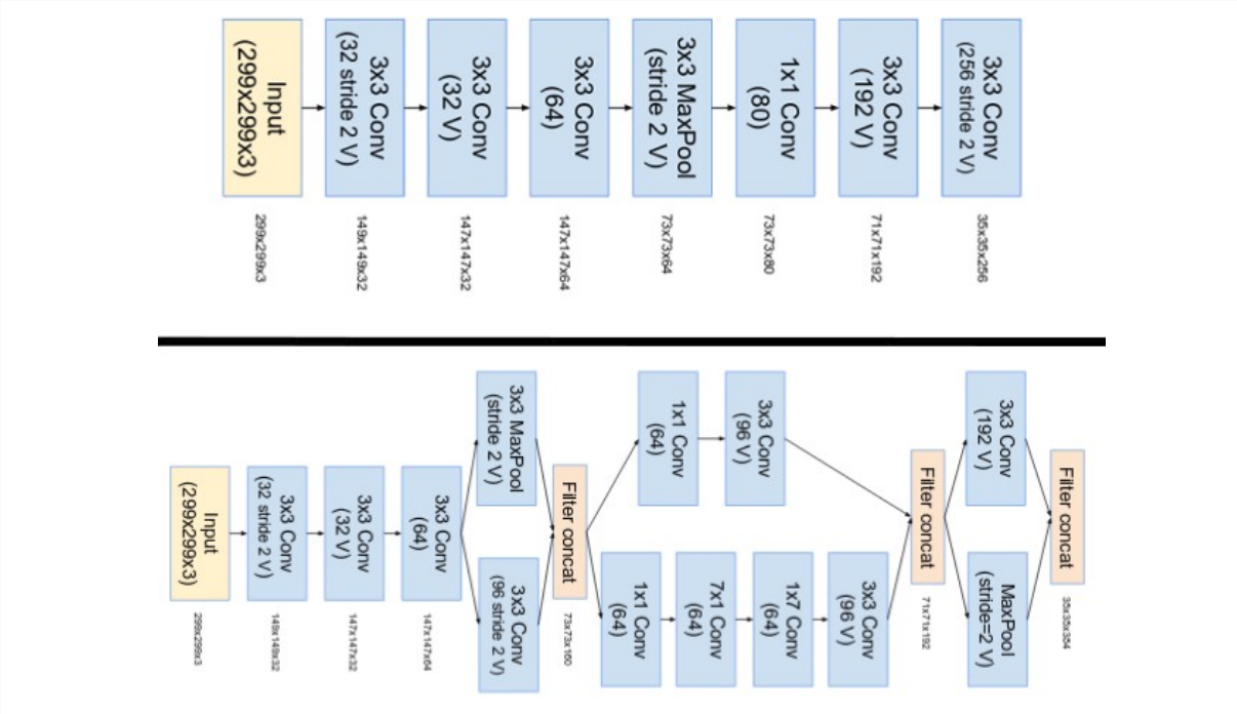


- И они дают специализацию в результатах:



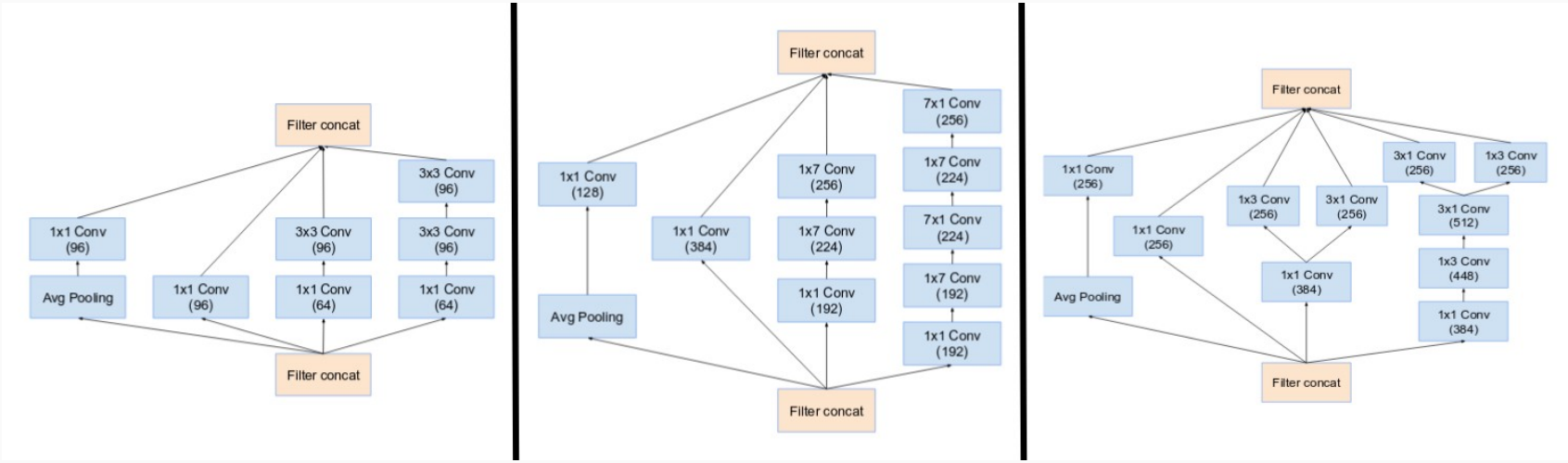
Inception v4 и Inception ResNet

- Ещё одна классическая статья (Szegedy et al., 2016): вводятся Inception v4 и Inception ResNet.
- Inception v4 – идея в том, чтобы всё стандартизировать, упростить модули. Во-первых, «черенок»:



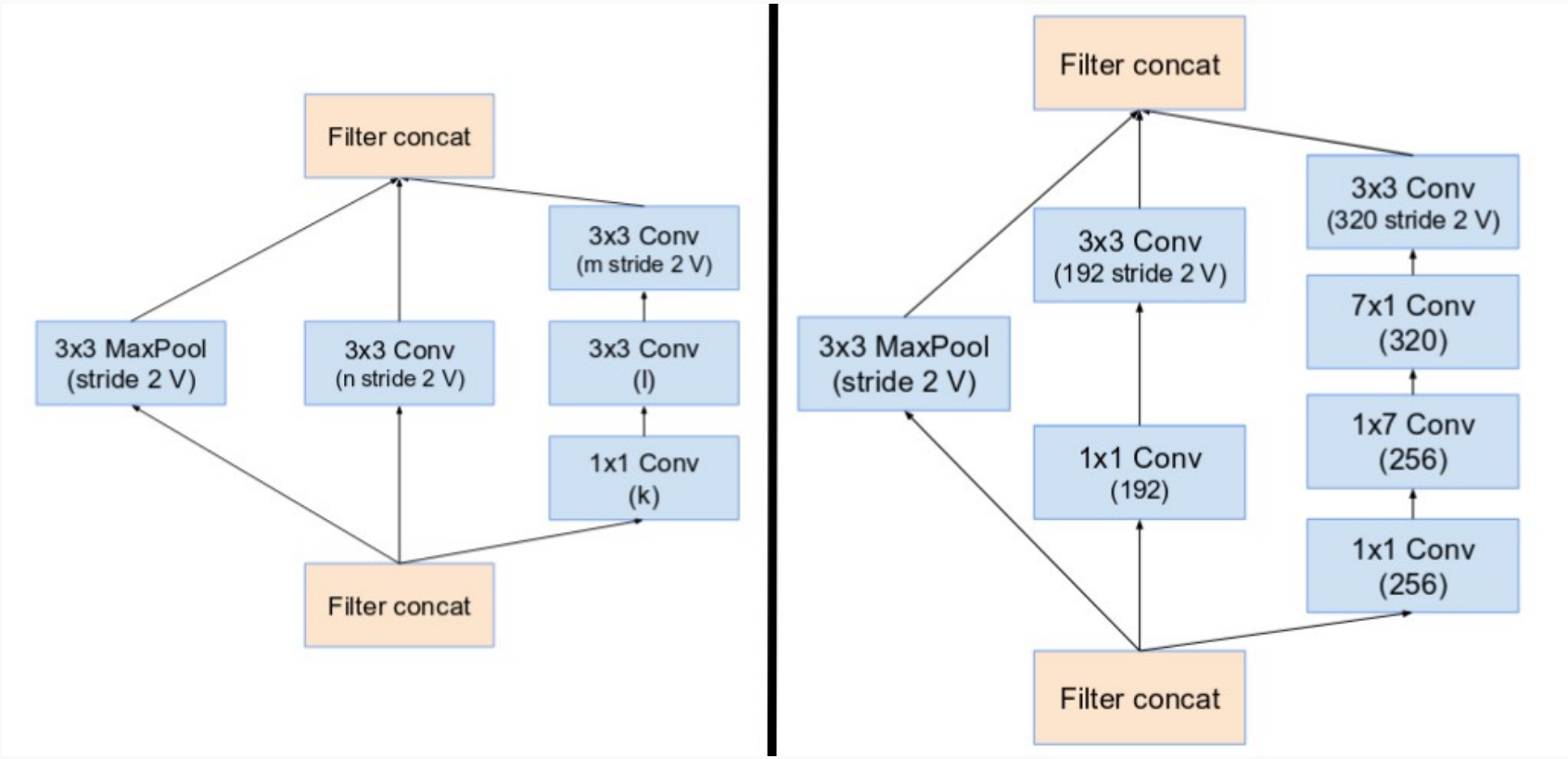
Inception v4 и Inception ResNet

- Во-вторых, теперь три базовых блока A, B, C:



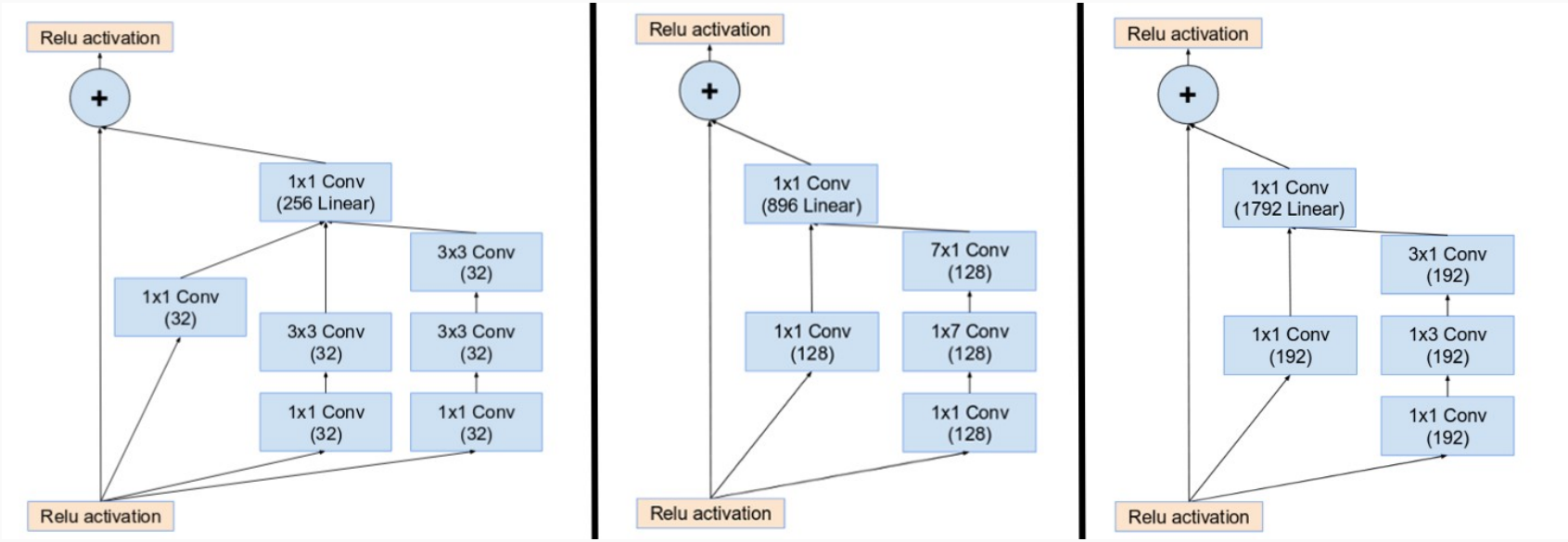
Inception v4 и Inception ResNet

- И специальные reduction blocks для сокращения размерности сетки:



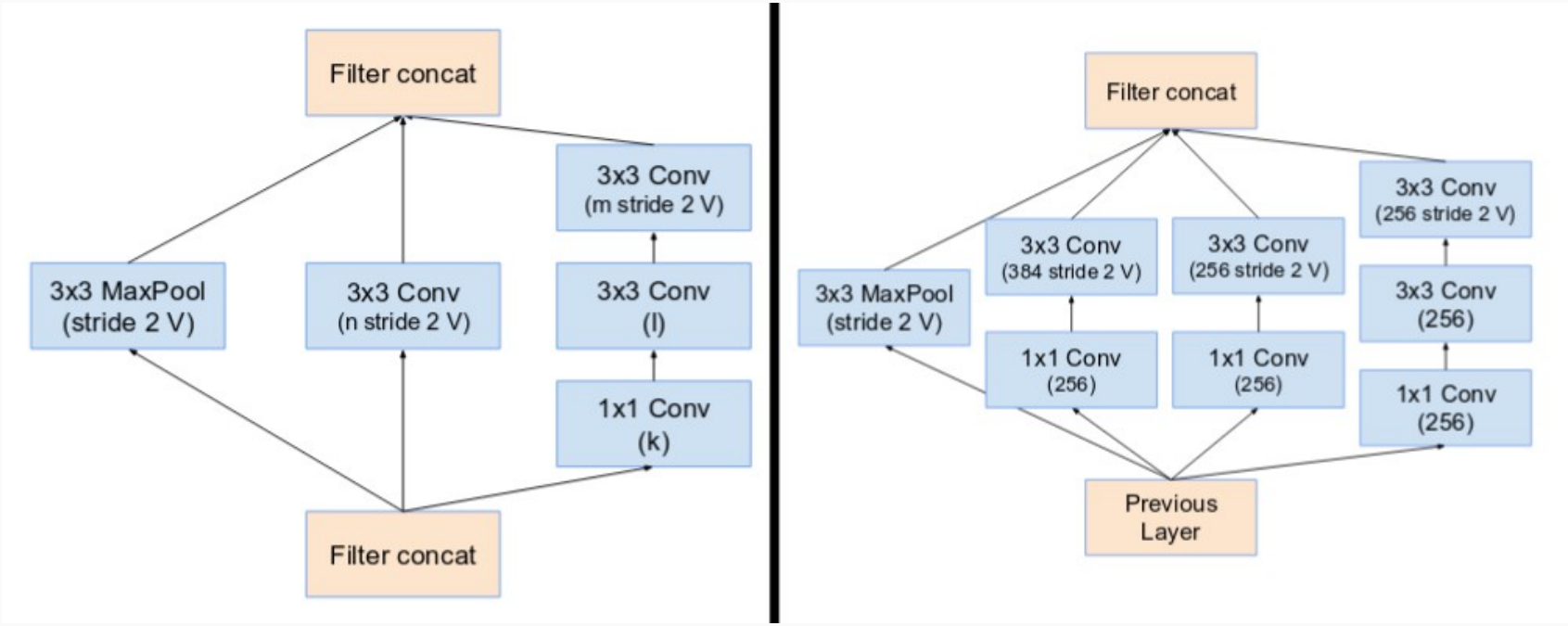
Inception v4 и Inception ResNet

- В Inception ResNet к тем же блокам добавляются остаточные связи:



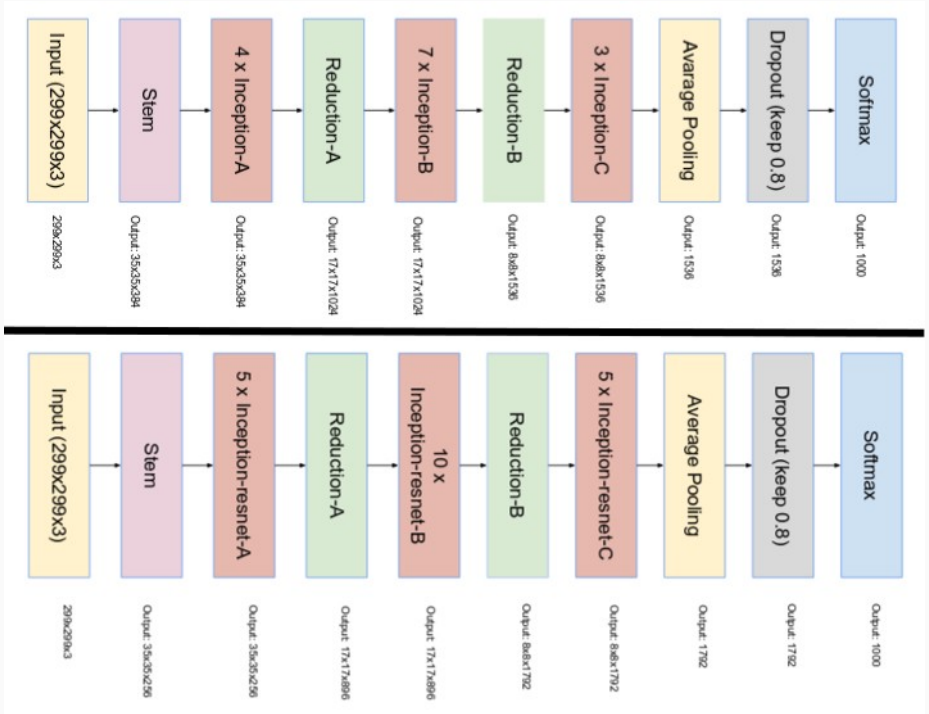
Inception v4 и Inception ResNet

- Субдискретизации теперь нет, но по-прежнему есть reduction blocks:



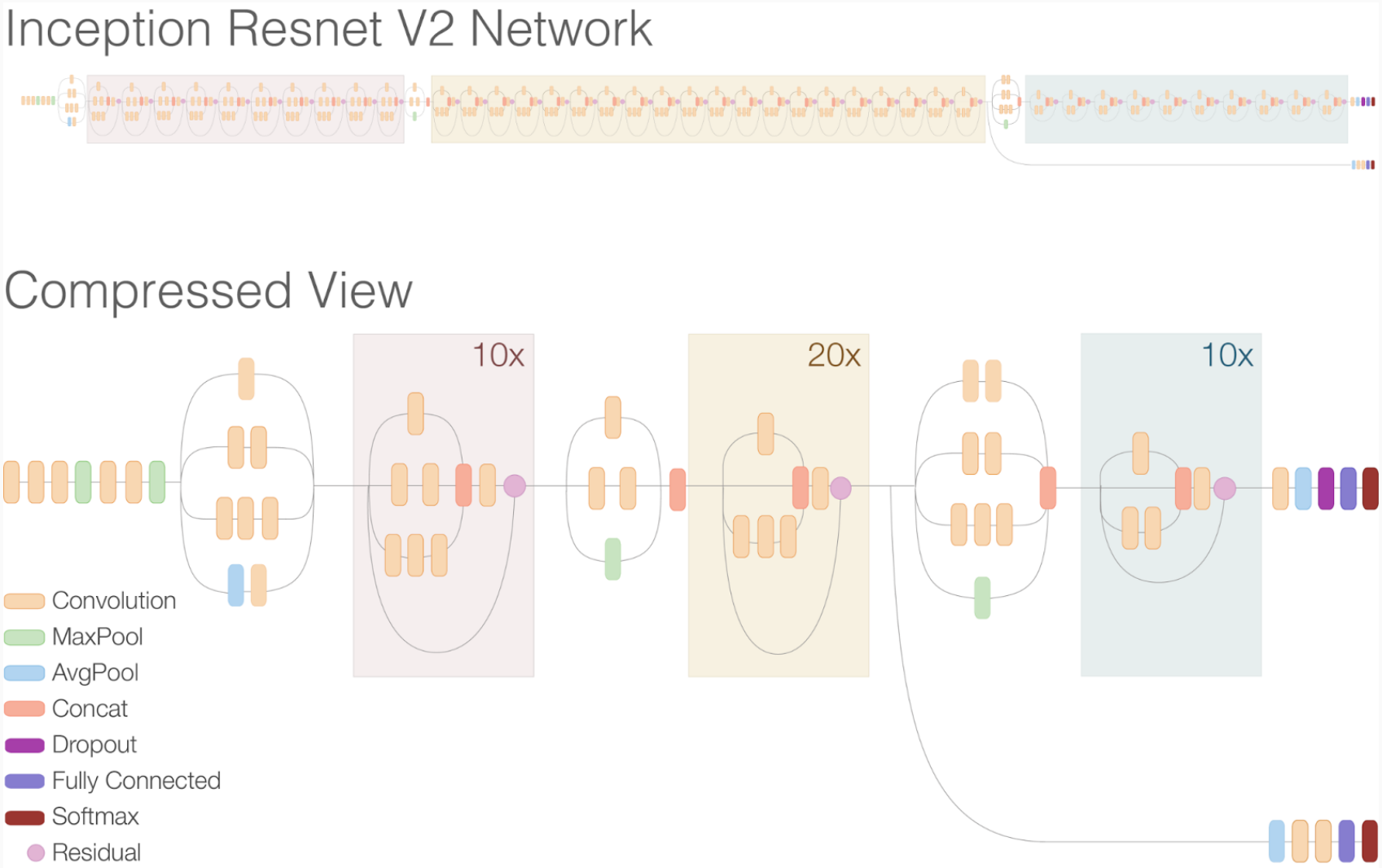
Inception v4 и Inception ResNet

- В итоге архитектура даже упростилась; сверху Inception v4, снизу Inception ResNet:



Inception v4 и Inception ResNet

- Inception ResNet v2:



Inception v4 и Inception ResNet

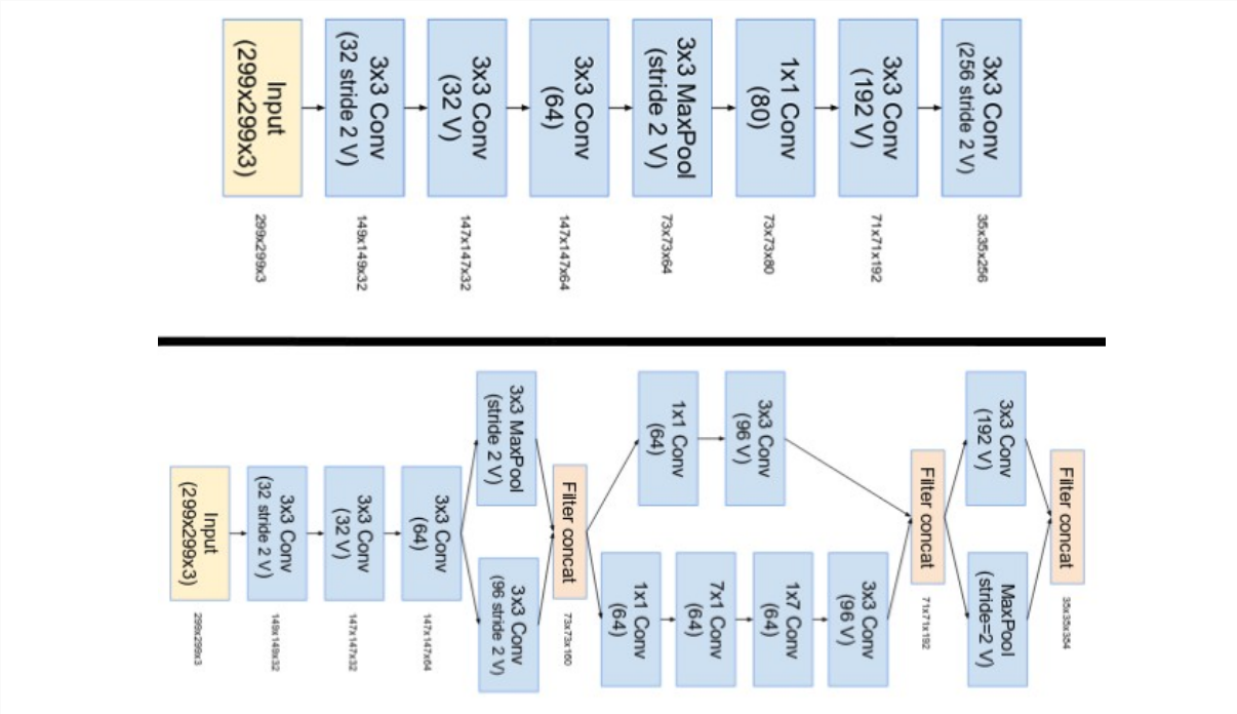
- И работает неплохо:

Network	Crops	**Top-1 Error**	**Top-5 Error**
ResNet-151 [5]	10	21.4%	5.7%
Inception-v3 [15]	12	19.8%	4.6%
Inception-ResNet-v1	12	19.8%	4.6%
Inception-v4	12	18.7%	4.2%
Inception-ResNet-v2	12	18.7%	4.1%

Network	Crops	**Top-1 Error**	**Top-5 Error**
ResNet-151 [5]	dense	19.4%	4.5%
Inception-v3 [15]	144	18.9%	4.3%
Inception-ResNet-v1	144	18.8%	4.3%
Inception-v4	144	17.7%	3.8%
Inception-ResNet-v2	144	17.8%	3.7%

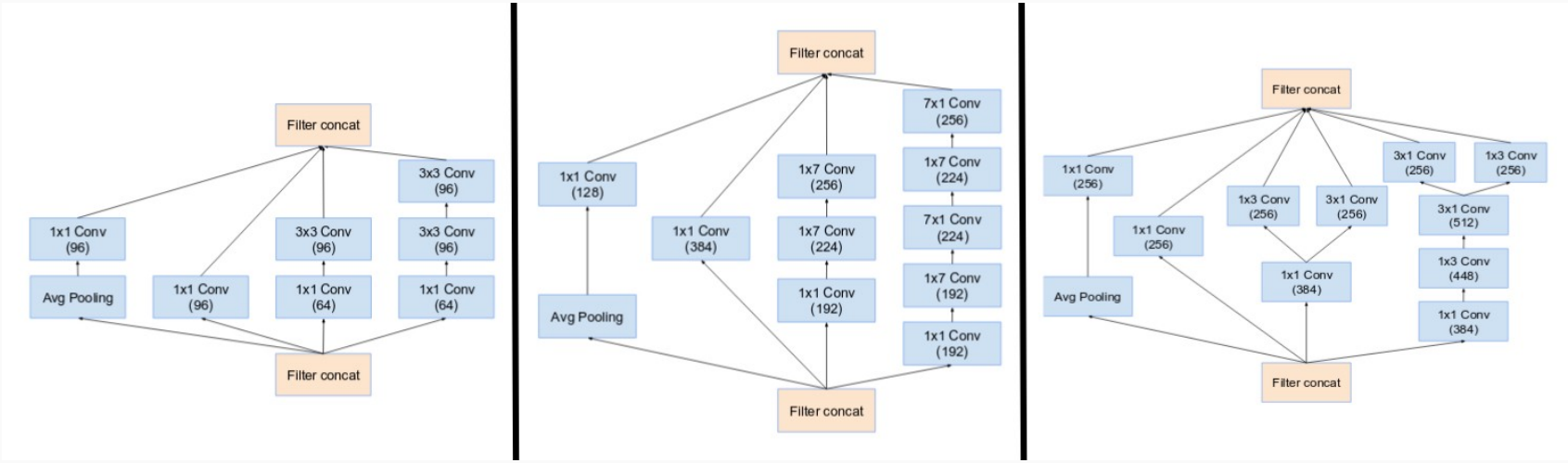
Inception v4 и Inception ResNet

- Ещё одна классическая статья (Szegedy et al., 2016): вводятся Inception v4 и Inception ResNet.
- Inception v4 – идея в том, чтобы всё стандартизировать, упростить модули. Во-первых, «черенок»:



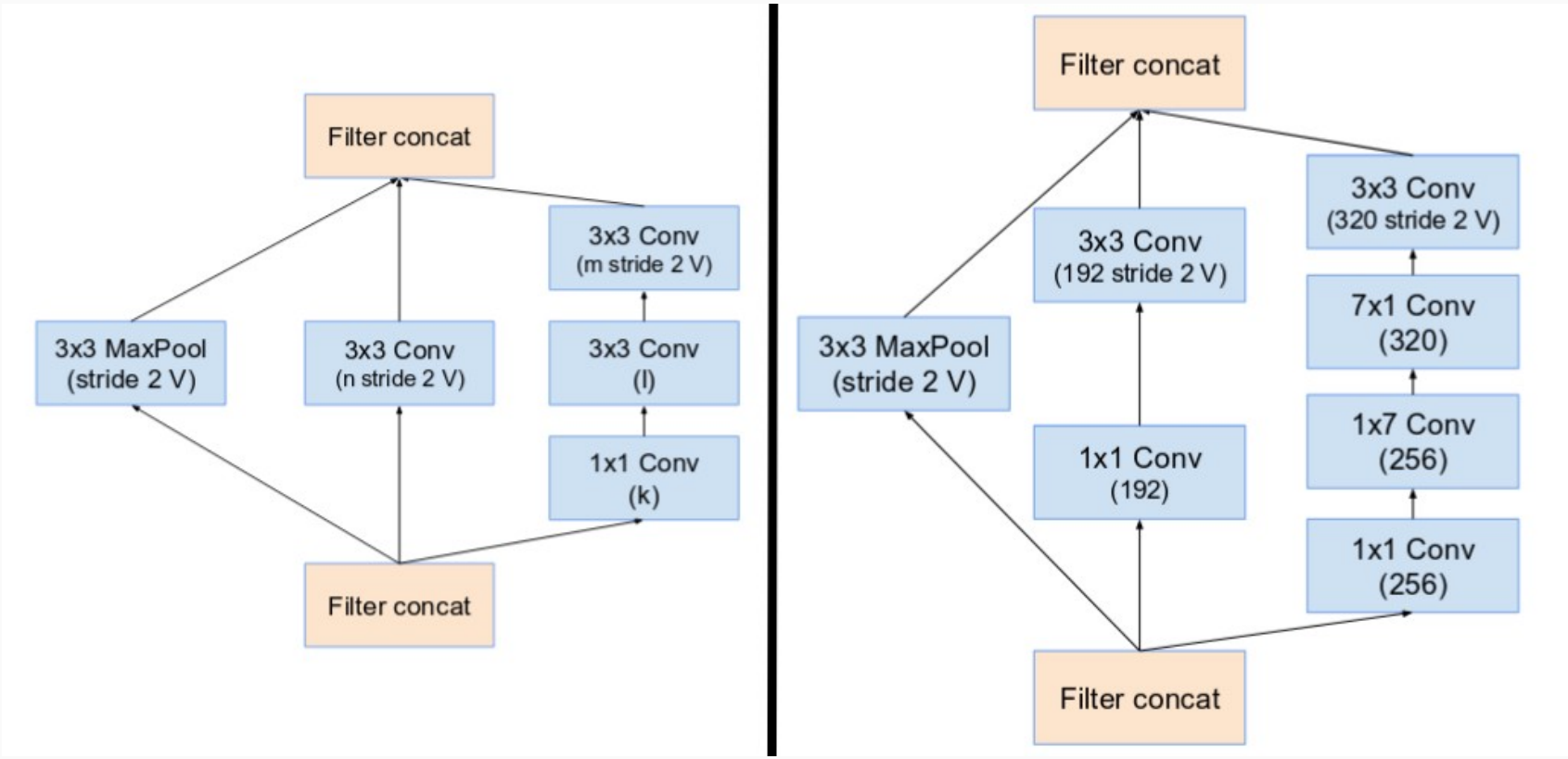
Inception v4 и Inception ResNet

- Во-вторых, теперь три базовых блока A, B, C:



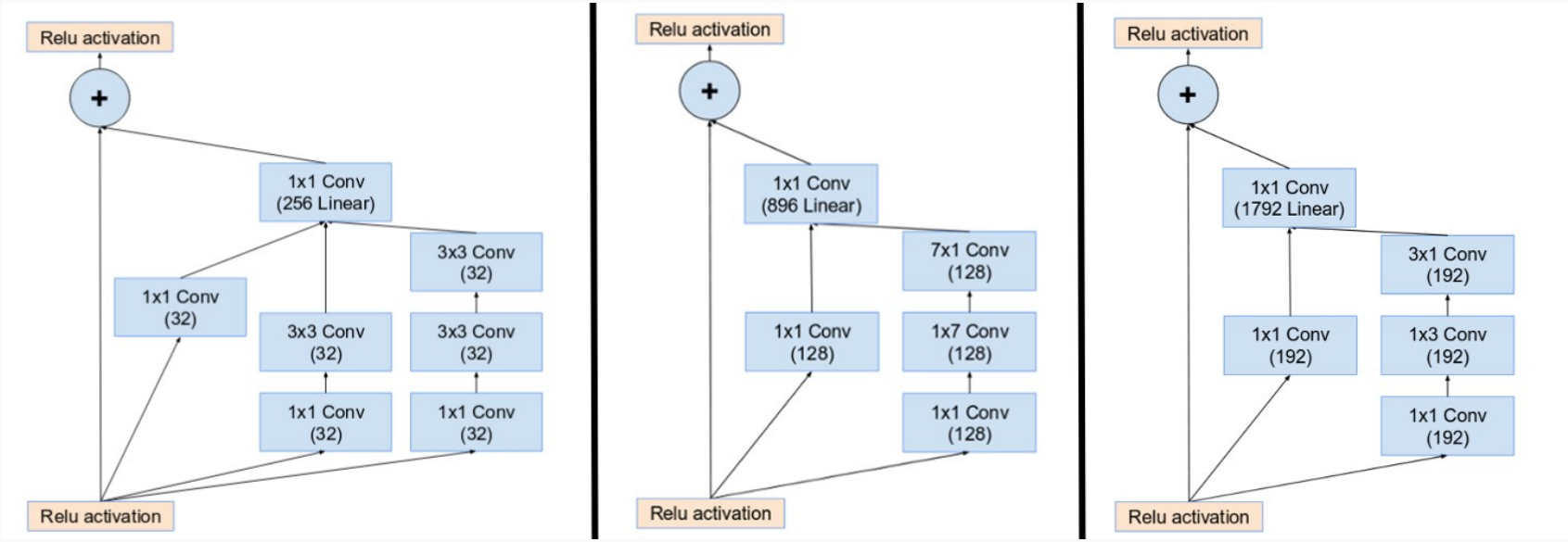
Inception v4 и Inception ResNet

- И специальные reduction blocks для сокращения размерности сетки:



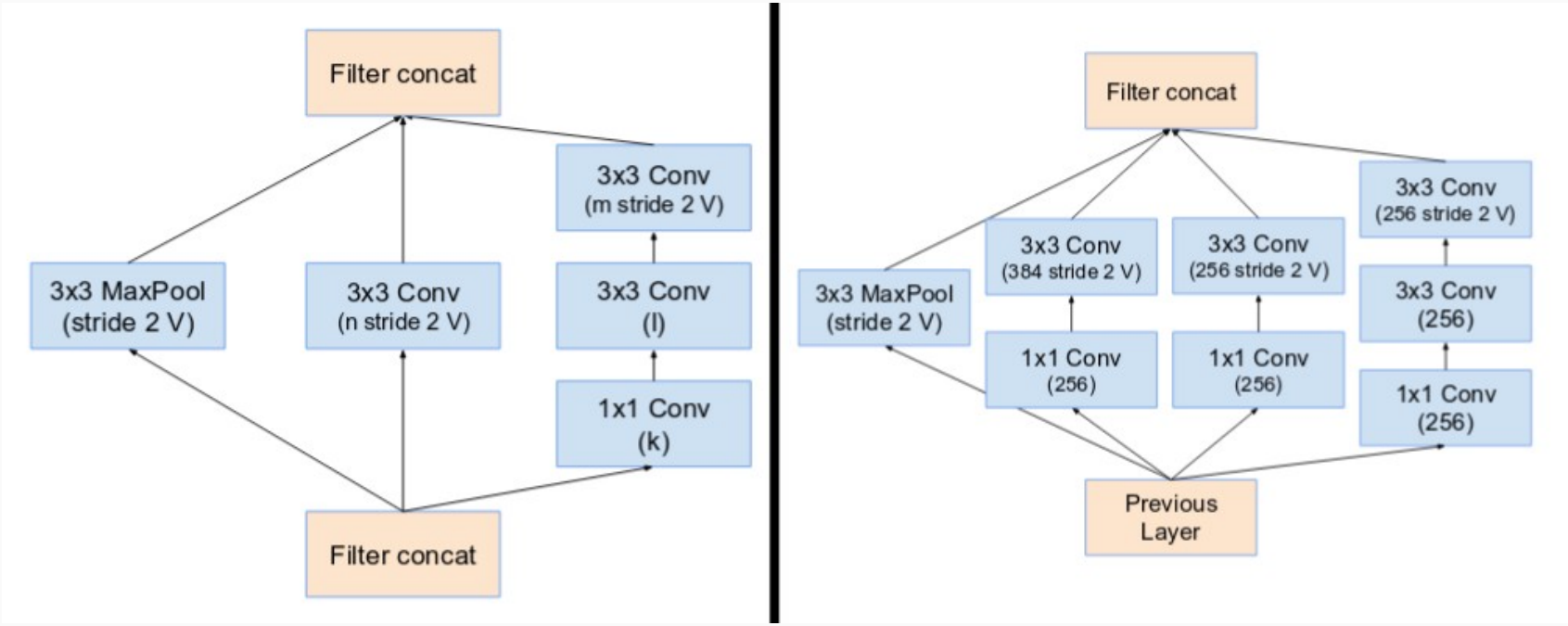
Inception v4 и Inception ResNet

- В Inception ResNet к тем же блокам добавляются остаточные связи:



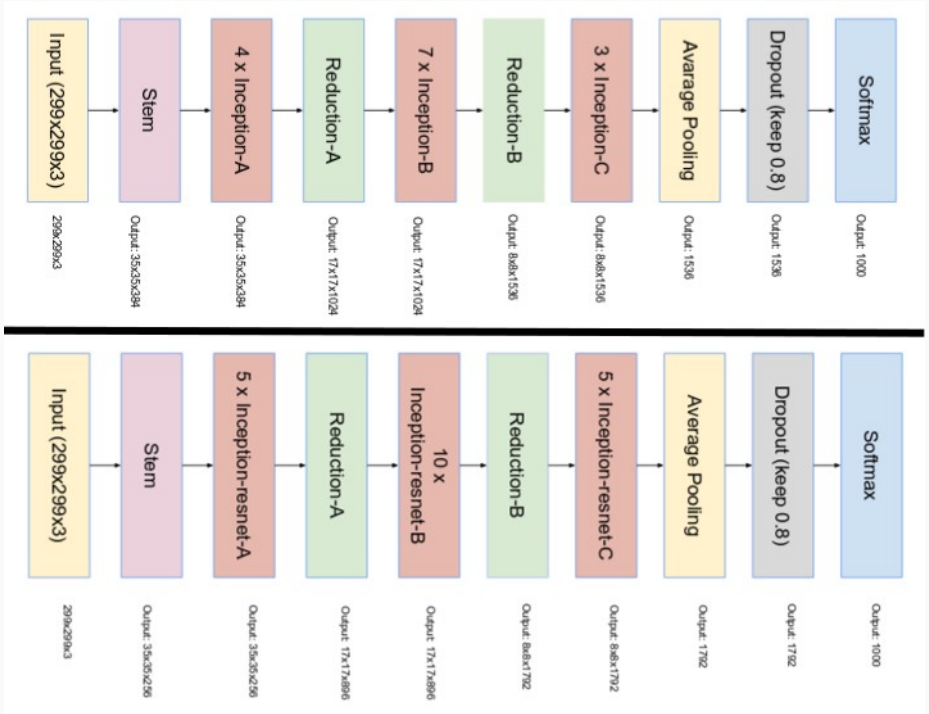
Inception v4 и Inception ResNet

- Субдискретизации теперь нет, но по-прежнему есть reduction blocks:



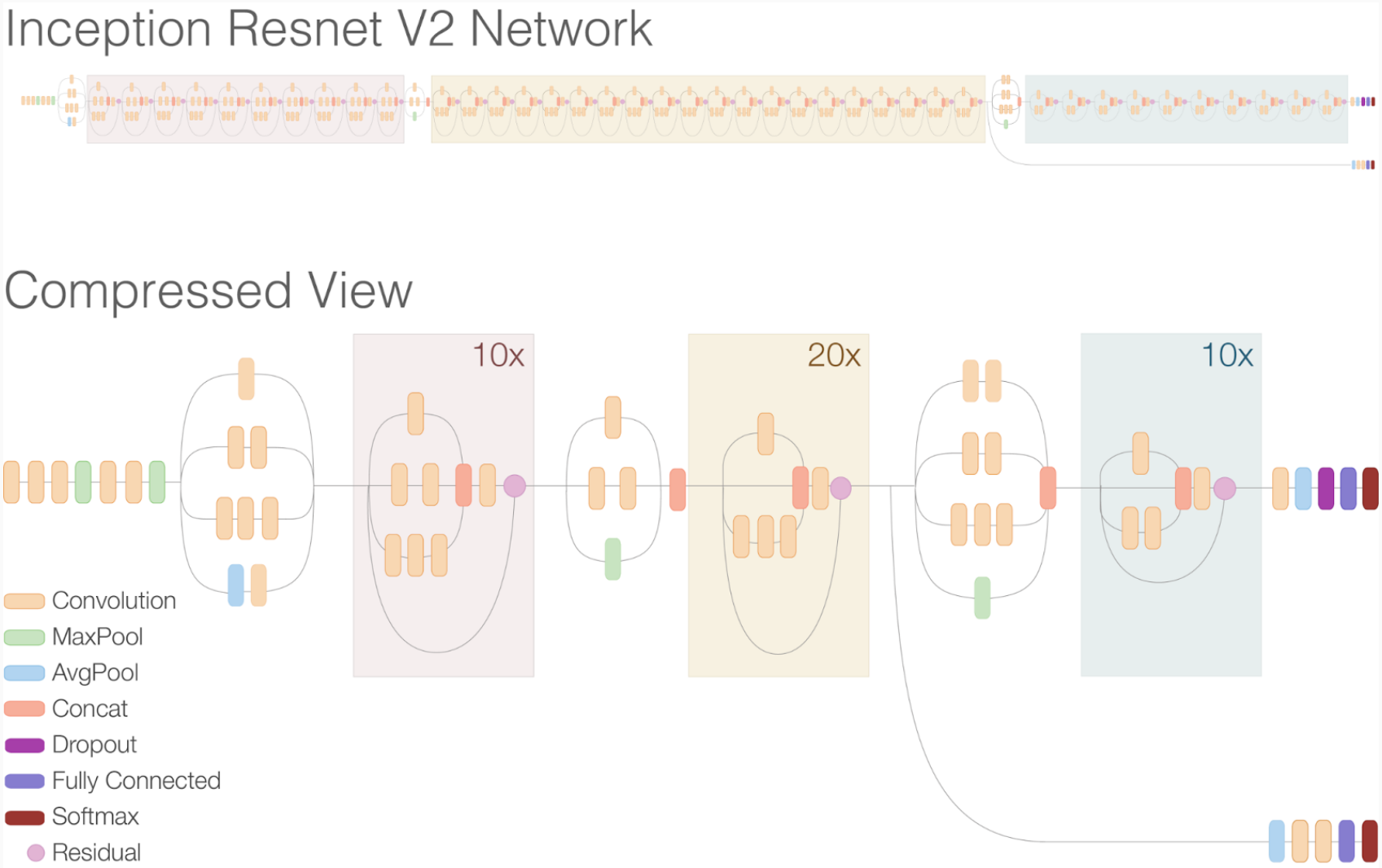
Inception v4 и Inception ResNet

- В итоге архитектура даже упростилась; сверху Inception v4, снизу Inception ResNet:



Inception v4 и Inception ResNet

- Inception ResNet v2:



Inception v4 и Inception ResNet

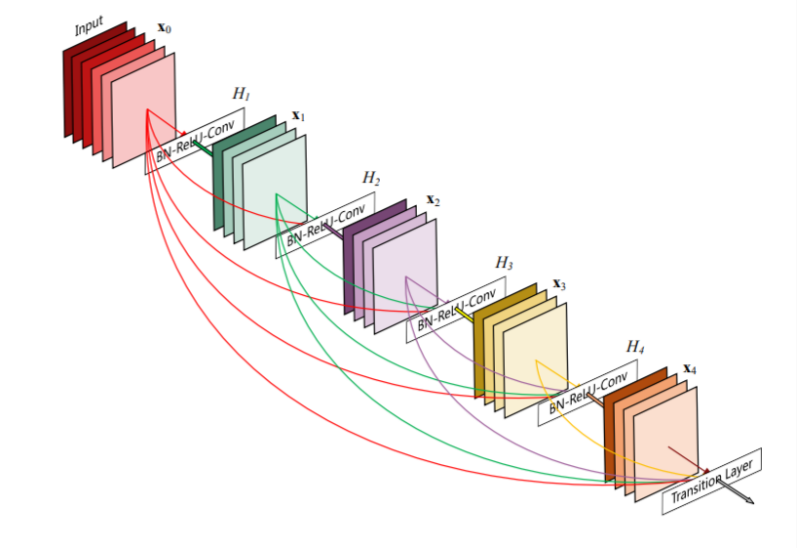
- И работает неплохо:

Network	Crops	**Top-1 Error**	**Top-5 Error**
ResNet-151 [5]	10	21.4%	5.7%
Inception-v3 [15]	12	19.8%	4.6%
Inception-ResNet-v1	12	19.8%	4.6%
Inception-v4	12	18.7%	4.2%
Inception-ResNet-v2	12	18.7%	4.1%

Network	Crops	**Top-1 Error**	**Top-5 Error**
ResNet-151 [5]	dense	19.4%	4.5%
Inception-v3 [15]	144	18.9%	4.3%
Inception-ResNet-v1	144	18.8%	4.3%
Inception-v4	144	17.7%	3.8%
Inception-ResNet-v2	144	17.8%	3.7%

DenseNet

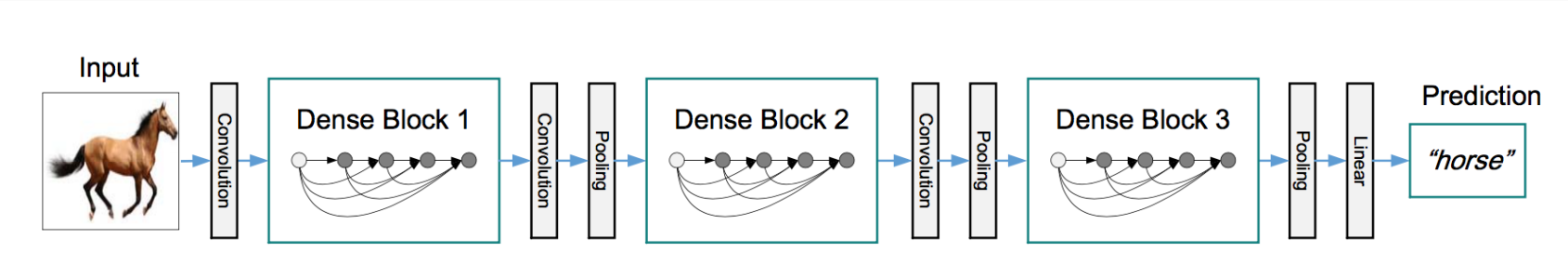
- DenseNet (Huang et al., 2017, best paper at CVPR 2017) – полезно включать не только последовательные соединения между слоями:



- Каждый с каждым! Выглядит странно – сколько ж весов нужно?..

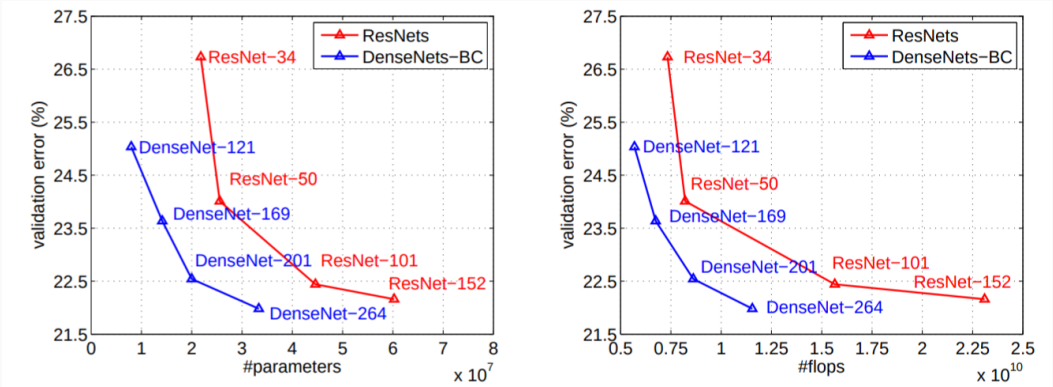
DenseNet

- Но на самом деле нужно не так уж много; такие соединения позволяют мало фич использовать.
- Например, $k = 12$ фильтров на каждый уровень – это значит, что на уровне m будет $k_0 + k(m-1)$ фич доступно.
- Такие конструкции объединяют в блоки:



DenseNet

- В DenseNet меньше весов, обучается быстрее и лучше:



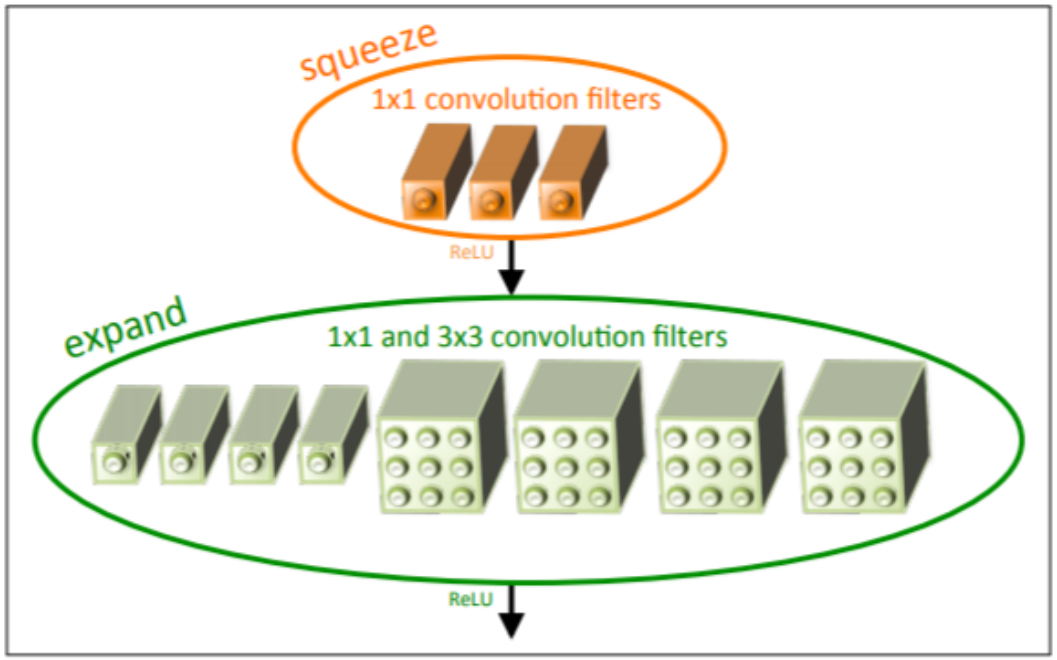
Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [32]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [34]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [42]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

SqueezeNet

- SqueezeNet (Iandola et al., 2017) – как уменьшить число параметров:
  - заменять 3×3 фильтры на 1×1;
  - уменьшать число входов для 3×3;
  - делать downsampling как можно позже, чтобы карты активаций были побольше.

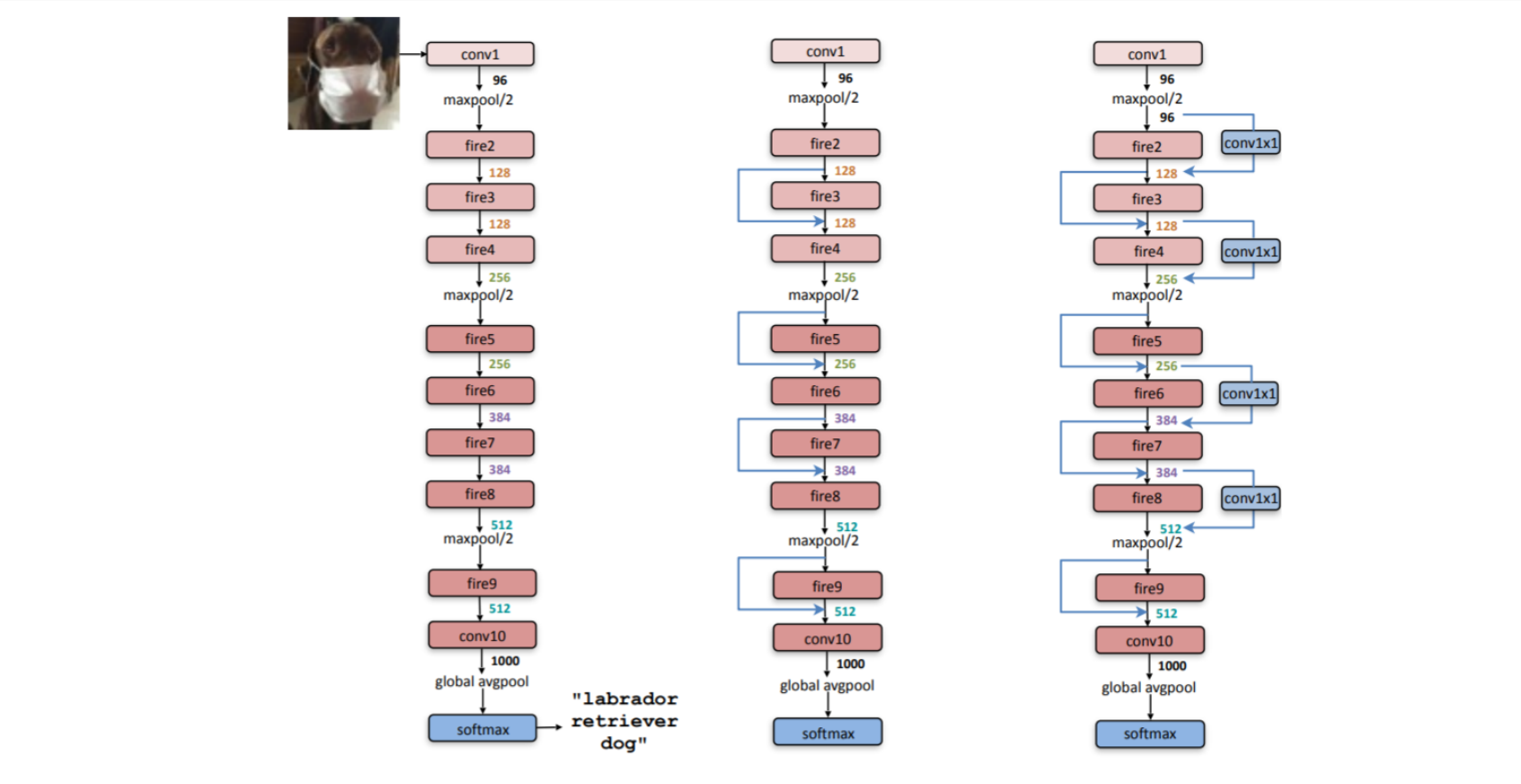
SqueezeNet

- Fire module:
  - squeeze convolutional layer (только 1×1);
  - expand layer (1×1 и 3×3).



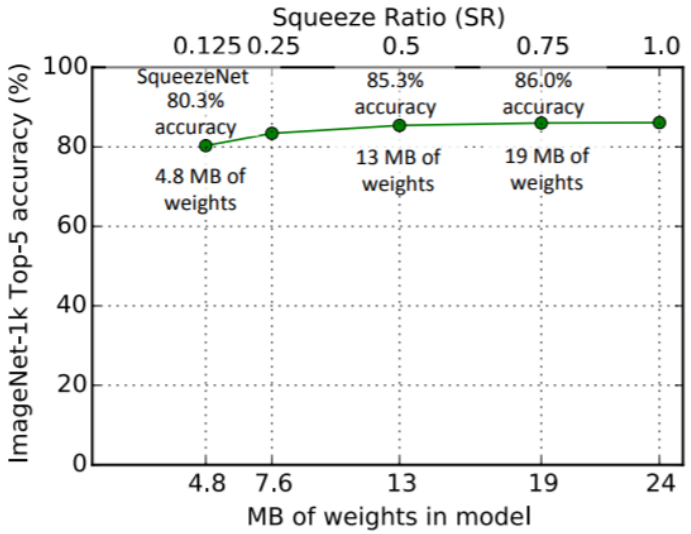
SqueezeNet

- И общая архитектура SqueezeNet:

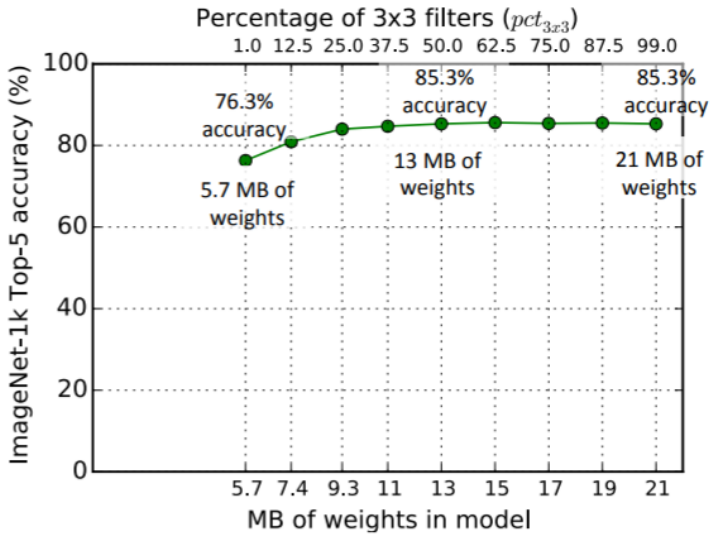


SqueezeNet

- Получается в 50 раз меньше параметров, чем у AlexNet, но:



(a) Exploring the impact of the squeeze ratio (SR) on model size and accuracy.

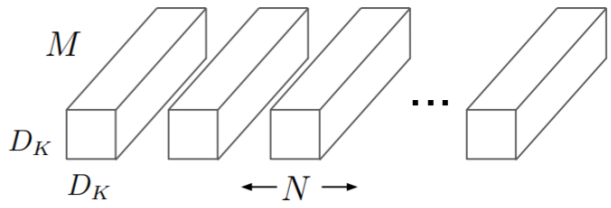


(b) Exploring the impact of the ratio of 3x3 filters in expand layers (pct_{3x3}) on model size and accuracy.

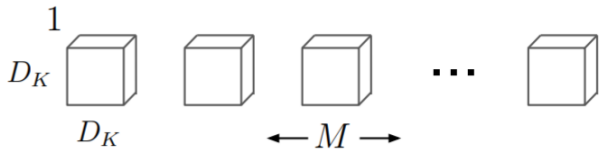
Architecture	Top-1 Accuracy	Top-5 Accuracy	Model Size
Vanilla SqueezeNet	57.5%	80.3%	4.8MB
SqueezeNet + Simple Bypass	**60.4%**	**82.5%**	4.8MB
SqueezeNet + Complex Bypass	58.8%	82.0%	7.7MB

MobileNet

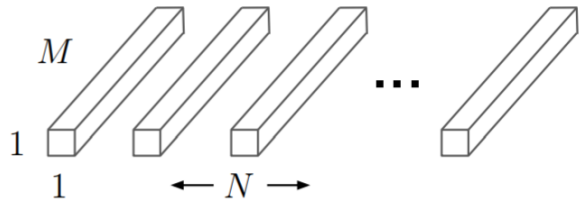
- MobileNet (Howard et al., 2017): сети для мобильных устройств.
- Depthwise separable convolutions: давайте разложим свёртку на depthwise (один фильтр к каждому каналу) и 1×1.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

MobileNet

- Тогда структура слоя будет чуть сложнее (но весов меньше), и общая структура не слишком глубокая:

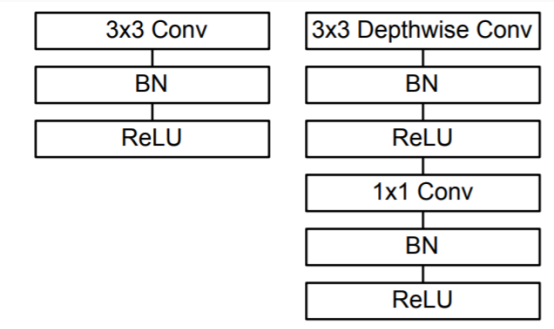


Table 1. MobileNet Body Architecture

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	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
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

- И получается, что можно сэкономить очень много параметров ценой небольшого ухудшения качества:

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Table 9. Smaller MobileNet Comparison to Popular Models

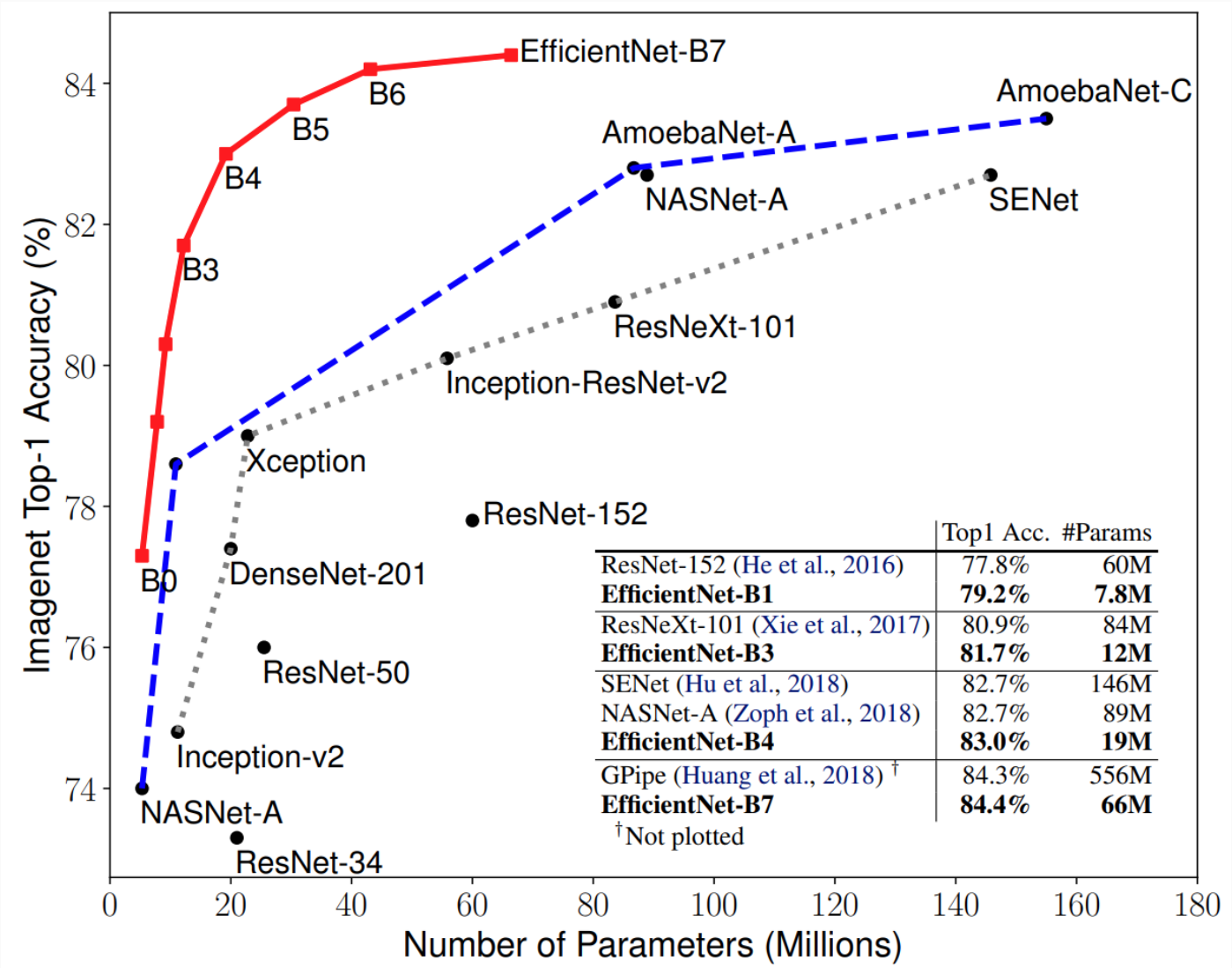
Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

Table 10. MobileNet for Stanford Dogs

Model	Top-1 Accuracy	Million Mult-Adds	Million Parameters
Inception V3 [18]	84%	5000	23.2
1.0 MobileNet-224	83.3%	569	3.3
0.75 MobileNet-224	81.9%	325	1.9
1.0 MobileNet-192	81.9%	418	3.3
0.75 MobileNet-192	80.5%	239	1.9

EfficientNet

- EfficientNet (Tan, Le, 2019): а вот и у neural architecture search дошли руки до свёрточных архитектур

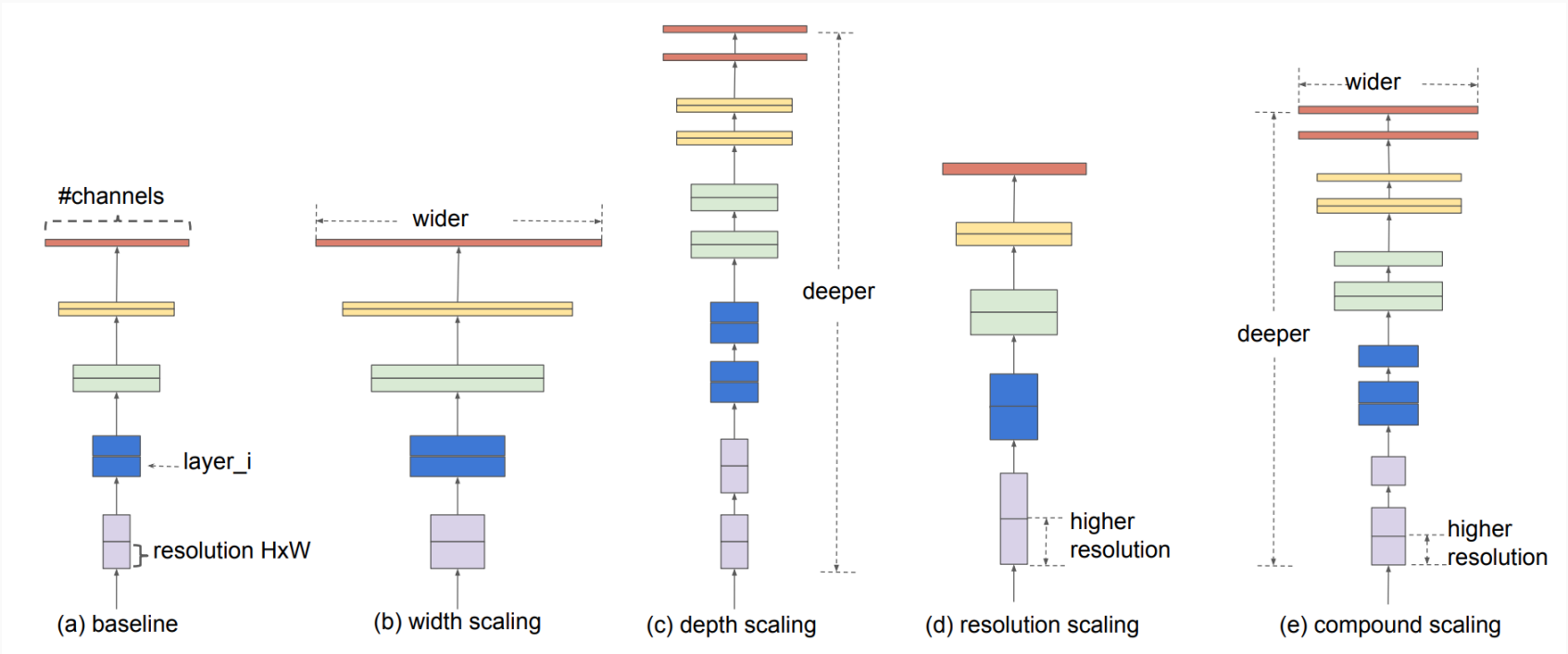


	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.2%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.7%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**83.0%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

EfficientNet

- Разные виды масштабирования для моделей:



Спасибо!

Спасибо за внимание!