Компьютерное зрение, часть 1

Сверточные сети: backbone архитектуры

Константин Архипенко

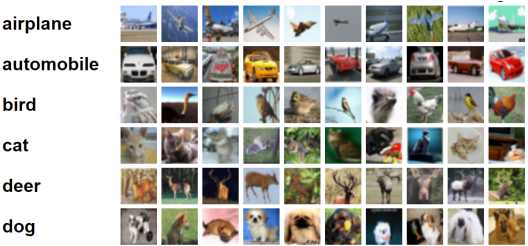
21 октября 2020 г.

ИСП РАН

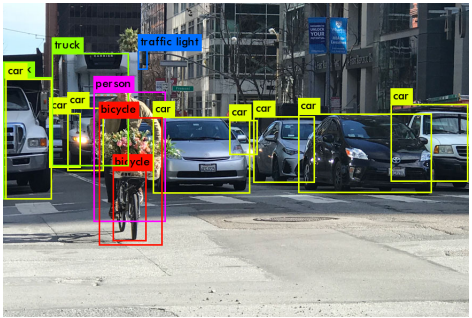
Задачи компьютерного зрения

ИСП РАН

- Классификация:



- Детекция объектов:

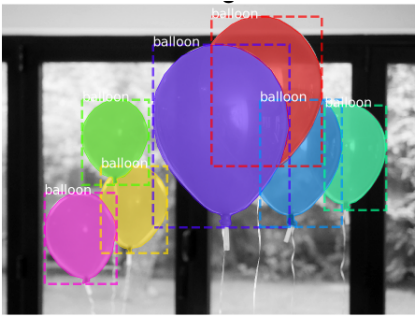


Задачи компьютерного зрения

ИСП РАН

- Semantic segmentation:

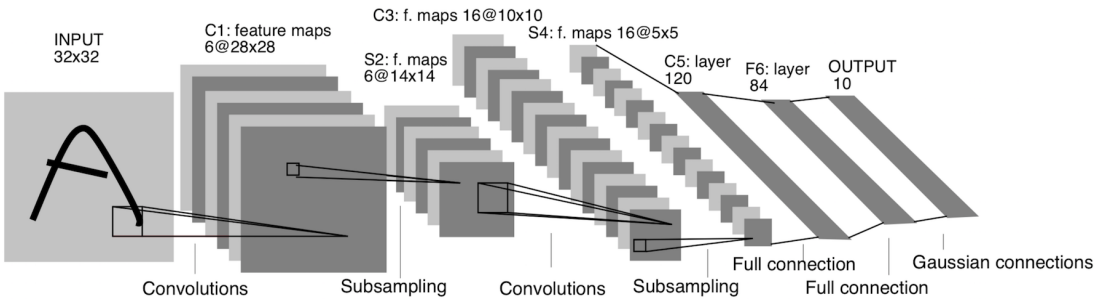
- Instance segmentation:



- И другие (pose estimation, генерация картинок, super-resolution, ...)
- State-of-the-art — сверточные нейросети

Эволюция сверточных сетей

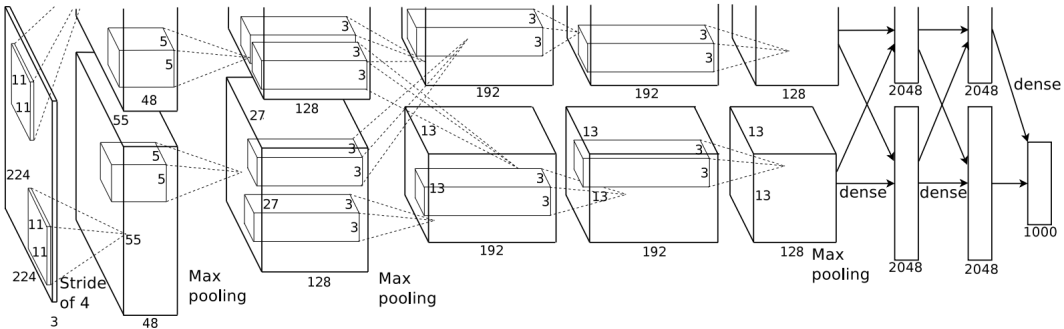
- **LeNet** (LeCun et al., 1998):



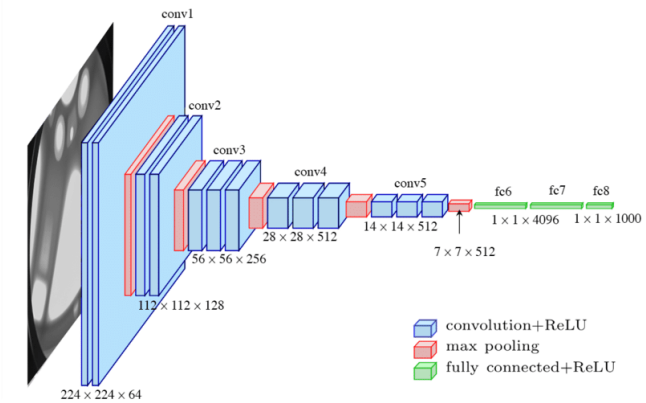
Эволюция сверточных сетей

ИСП РАН

- **AlexNet** (Krizhevsky et al., 2012):



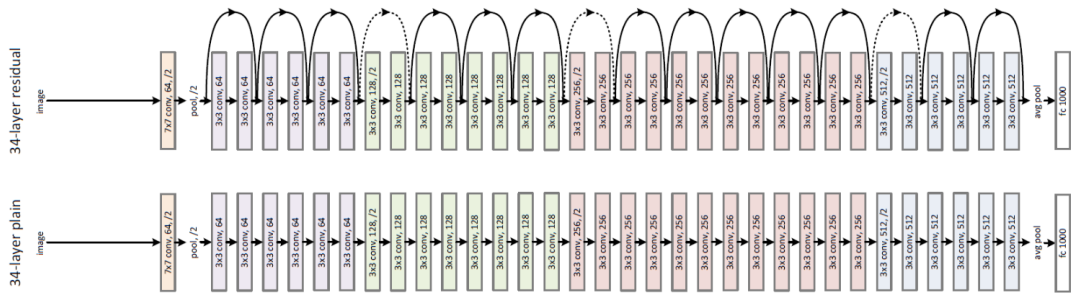
- **VGG** (Simonyan et al., 2014)[1]:



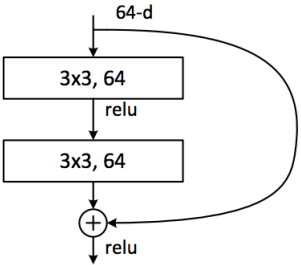
[1]картинка: mc.ai (VGG16)

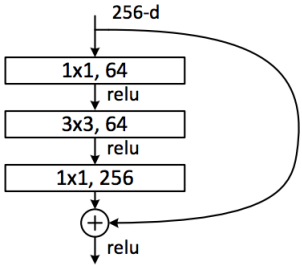
ResNet (He et al., 2015)

- **Skip connections** упрощают градиентную оптимизацию
- Пунктиром — свертки 1×1, изменяющие количество каналов
- ResNet-34 и plain 34-слойная сеть:



ResNet (He et al., 2015)





- **Bottleneck**: для ResNet-50 и глубже, более эффективный блок

Backbone и task-specific архитектуры

ИСП РАН

- SOTA архитектуры для задач компьютерного зрения часто можно разбить на **backbone** (извлечение признаков) и **task-specific** части (например, предсказание bounding box)
- Transfer learning: backbone можно обучать на больших датасетах для классификации (ImageNet)
- Сегодня рассмотрим популярные backbone[2]
- Следующая лекция: task-specific модели (детекторы, адаптации backbone для сегментации)

[2] https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

ResNeXt (Xie et al., 2016)

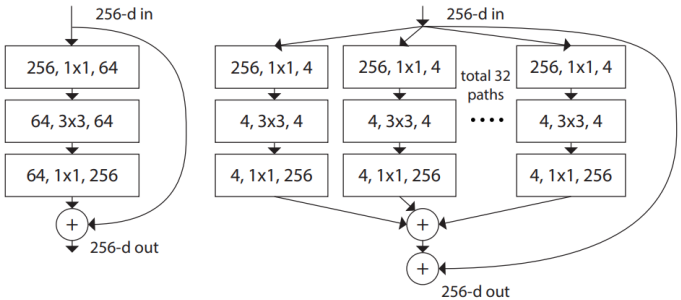
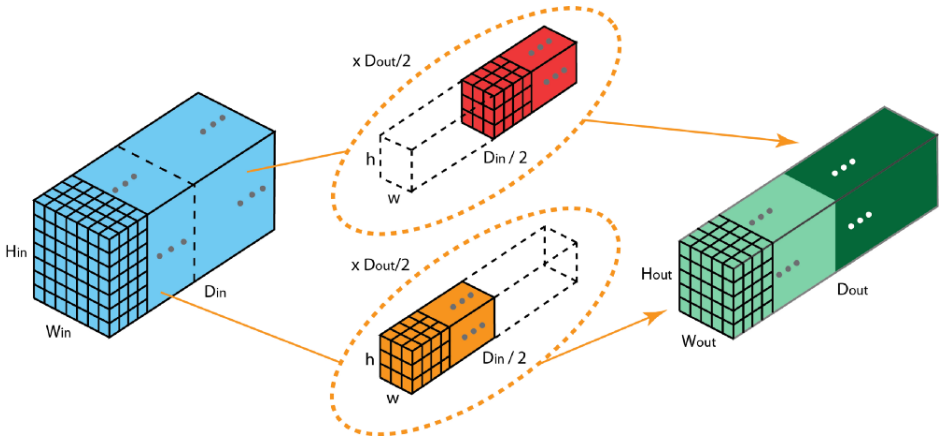


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

Количество параметров сверток:

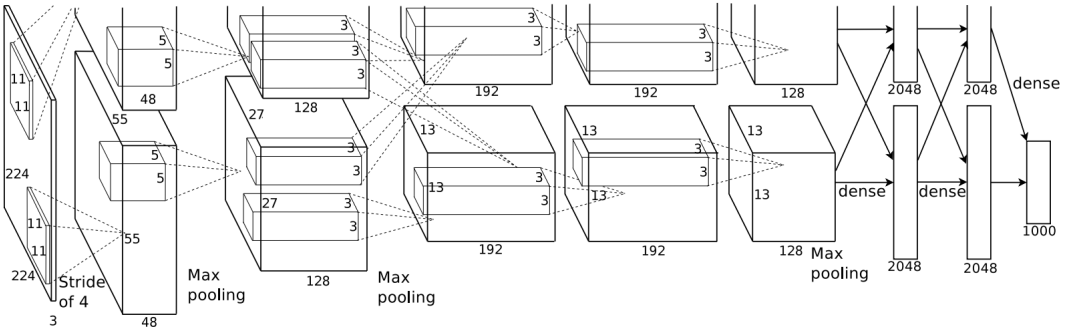
$256 \cdot 64 + 3 \cdot 3 \cdot 64 \cdot 64 + 64 \cdot 256$ VS $32 \cdot (256 \cdot 4 + 3 \cdot 3 \cdot 4 \cdot 4 + 4 \cdot 256)$

Grouped convolutions



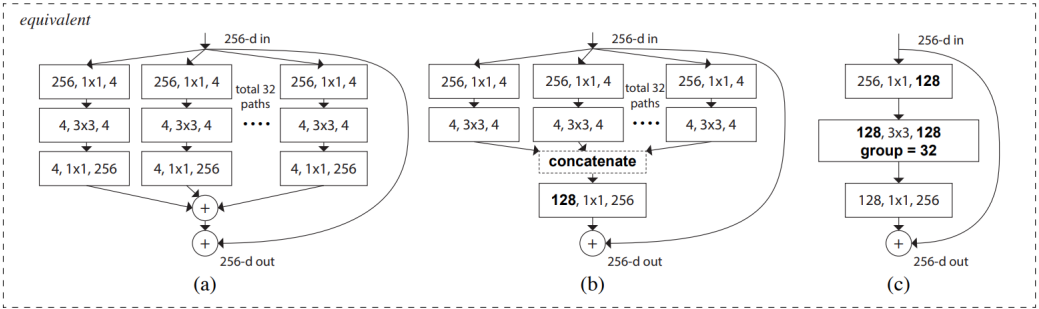
картинка: mc.ai

Grouped convolutions в AlexNet (Krizhevsky et al., 2012)



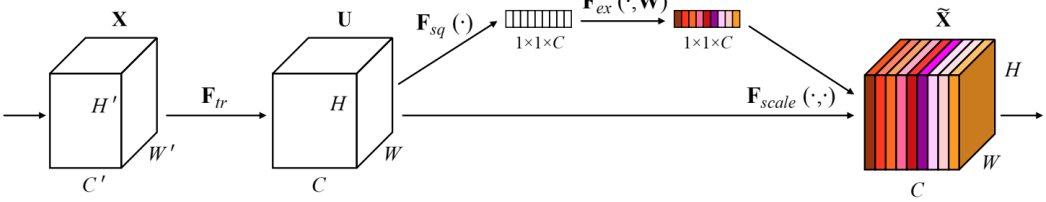
- Благодаря grouped conv смогли обучить на двух GPU с 3 GB памяти
- В статье ResNeXt показывается, что grouped conv могут и улучшать качество

Эквивалентные представления ResNeXt блока



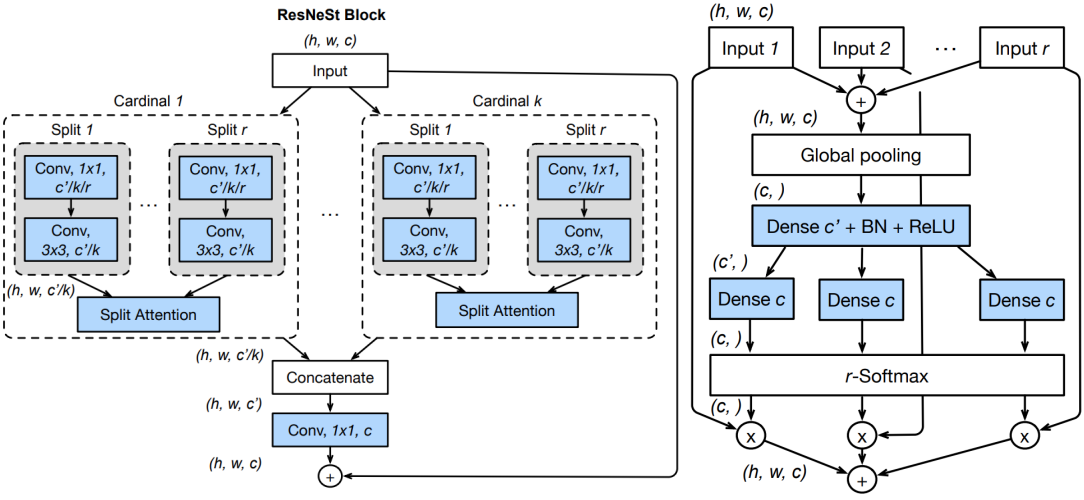
- Лучше, чем ResNet, на ImageNet, CIFAR, COCO minival

- Механизм внимания (self-attention)



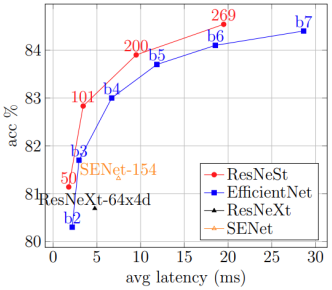
- Можно для F_{tr} брать ResNeXt (получится SE-ResNeXt), Inception и др.
- F_{sq} — global average pooling: $F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j)$
- F_{ex} — двухслойная feedforward с bottleneck (ReLU слой + sigmoid слой)

ResNeSt (Zhang et al., 2020): split-attention



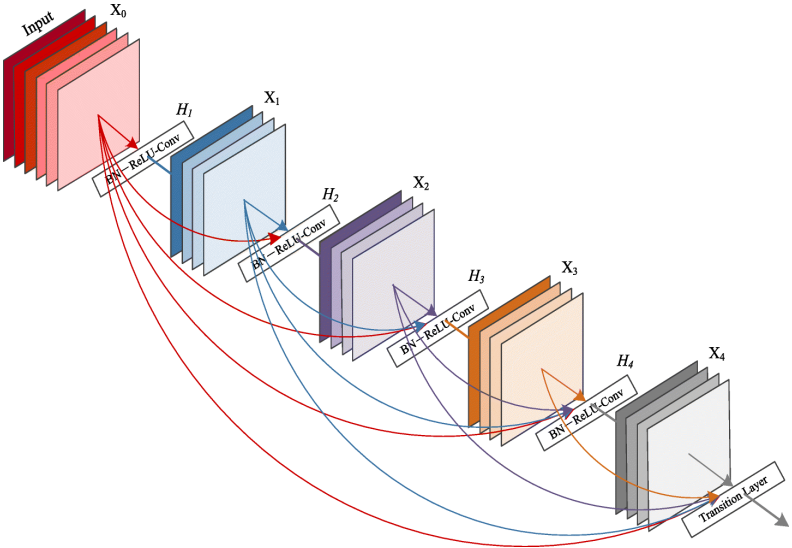
ResNeSt (Zhang et al., 2020): split-attention

- Отличие от ResNeXt: количество групп в свертке 3×3 равно KR, а не K (гиперпараметр R — radix)
- Но выход 3×3 для группы делают с $\frac{c'}{k}$ каналами, а не $\frac{c'}{KR}$ (видимо, чтобы поднять качество)
- Перед self-attention суммируют R таких выходов

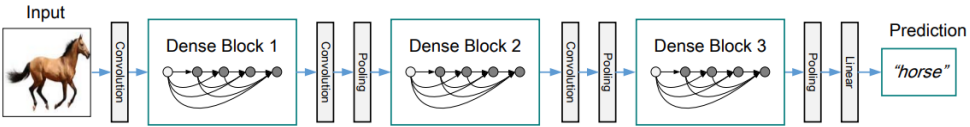


	#P	crop	img/sec	acc(%)
ResNeSt-101(ours)	48M	256	**291.3**	**83.0**
EfficientNet-B4 [55]	19M	380	149.3	83.0
SENet-154 [29]	146M	320	133.8	82.7
NASNet-A [74]	89M	331	103.3	82.7
AmoebaNet-A [45]	87M	299	-	82.8
ResNeSt-200 (ours)	70M	320	**105.3**	**83.9**
EfficientNet-B5 [55]	30M	456	84.3	83.7
AmoebaNet-C [45]	155M	299	-	83.5
ResNeSt-269 (ours)	111M	416	**51.2**	**84.5**
GPipe	557M	-	-	84.3
EfficientNet-B7 [55]	66M	600	34.9	84.4

DenseNet (Huang et al., 2016): dense block



DenseNet (Huang et al., 2016)



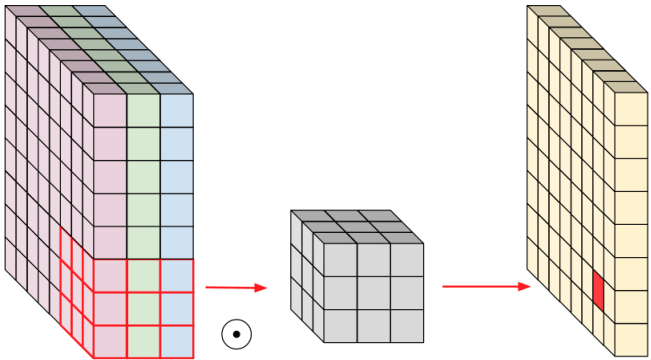
- Dense block — несколько повторений пары сверток (1×1 и 3×3, каждая с BN и ReLU)
- Каждая пара внутри dense block увеличивает количество каналов на growth rate
- Свертка 1×1 внутри dense block является bottleneck: на выходе $4 \cdot$ growth rate каналов
- Transition layer (между dense block): уменьшает количество каналов (conv) и разрешение (pool)

DenseNet: архитектура

ИСП РАН

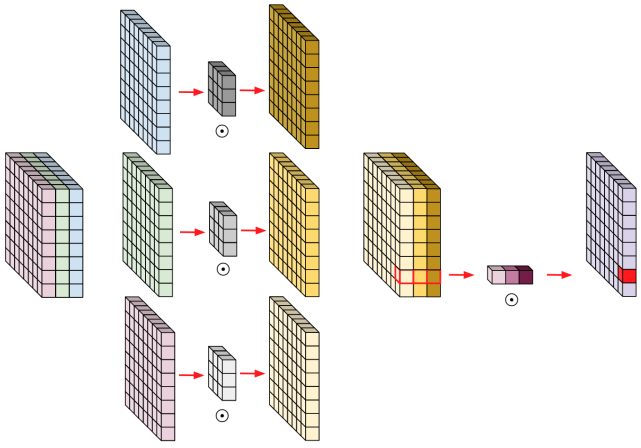
Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Обычная свертка (один фильтр)[3]



[3] картинка: eli.thegreenplace.net

Depthwise separable convolutions



Depthwise (= grouped с 1 каналом в группе) + pointwise свертка

MobileNet (Howard et al., 2016)

ИСП РАН

- Batch norm и ReLU после каждой свертки, в том числе между depthwise и 1×1
- 95% времени и 75% параметров занимают свертки 1×1

Модель	Acc	Mult-adds	Params
MobileNet	0.706	569M	4.2M
Conv MobileNet	0.717	4866M	29.3M

- Можно еще сильнее уменьшить модель (width/resolution multiplier), но потери в accuracy уже большие

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	$1 \times 1 \times 1000$

MobileNetV2 (Sandler et al., 2018)

ИСП РАН

- Применяются идеи ResNet — skip connections и bottleneck
- Вставим **линейный** bottleneck между depthwise и pointwise свертками (+ к эффективности)
- Skip connections **между линейными bottleneck** вместо expanded представлений (согласно экспериментам, улучшает качество)

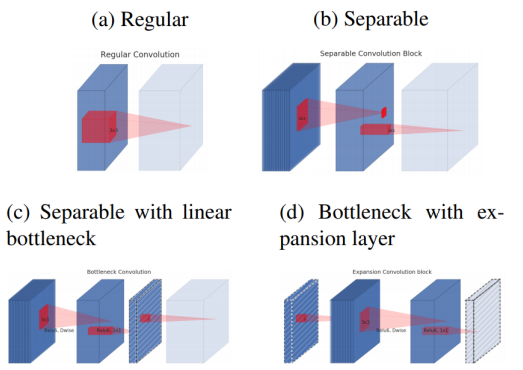


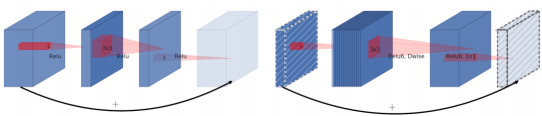
Figure 2: Evolution of separable convolution blocks. The diagonally hatched texture indicates layers that do not contain non-linearities. The last (lightly colored) layer indicates the beginning of the next block. Note: 2d and 2c are equivalent blocks when stacked. Best viewed in color.

MobileNetV2 (Sandler et al., 2018)

ИСП РАН

- Работает лучше, чем V1, на ImageNet и как feature extractor для детекции и сегментации
- Для детекции взяли SSDLite (SSD с замененными на depthwise свертками) и присоединили к выходам MobileNetV2 — получилось лучше, чем YOLOv2, у которого **11x** больше параметров

(a) Residual block (b) Inverted residual block



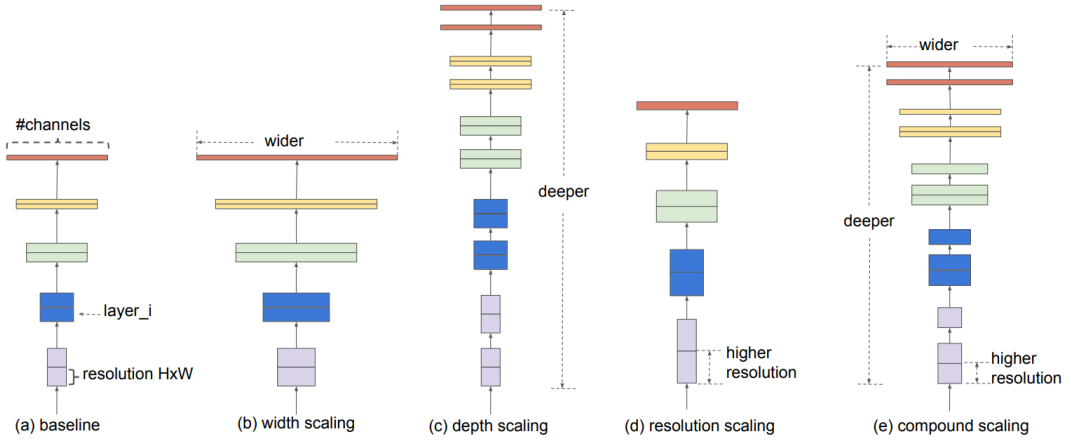
Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

EfficientNet (Tan & Le, 2019)

- **Neural architecture search**: можно почитать обзор Elsken et al.
- **Model scaling**: фиксируем baseline и просто пробуем пропорционально увеличивать/уменьшать ее части
- Один из тезисов статьи EfficientNet: увеличение количества слоев (d, depth), количества каналов (w, width) или разрешения (r) по отдельности быстро теряет смысл — надо увеличивать их **вместе**
- ResNetы и так уже большие — попробуем взять что-то похожее на MobileNetV2 в качестве baseline и поскейлить
- Оптимизация:

$$\max_{d,w,r} Accuracy(\mathcal{N}(d,w,r)) \quad Memory(\mathcal{N}) \leq T_1 \quad FLOPS(\mathcal{N}) \leq T_2$$

Варианты model scaling



Выбор baseline для EfficientNet

ИСП РАН

- Baseline модель **EfficientNet-B0** найдена с помощью алгоритма NAS, основанного на обучении с подкреплением (MnasNet, Tan et al. 2019, Zoph & Le 2017)
- Критерий эффективности в этом алгоритме (T — произвольно взятый эталон):

$$E(m) = Accuracy(m) \times [FLOPS(m)/T]^w \quad w = -0.07$$

- EfficientNet-B0 достаточно похожа на MobileNetV2 (хотя причина этому — сам алгоритм MnasNet)

Compound scaling

ИСП РАН

$$\text{depth: } d = \alpha^\phi$$
$$\text{width: } w = \beta^\phi$$
$$\text{resolution: } r = \gamma^\phi$$
$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

- ϕ — гиперпараметр алгоритма (насколько сильно можно скейлить)
- *d*, *w* и *r* здесь относительные (для baseline равны 1)
- FLOPS увеличится в $(\alpha \cdot \beta^2 \cdot \gamma^2)^\phi$ раз при $\phi > 0$ по сравнению с $\phi = 0$
- Ограничение (s.t.) выбрано так, чтобы разница была в 2^ϕ раз
- Grid search с $\phi = 1$, потом увеличение ϕ с фикс. найденными α, β, γ

Результаты статьи EfficientNet

ИСП РАН

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.1%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.1%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.6%**	**95.7%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x

Сейчас есть SOTA модель **EfficientDet** для детекции с EfficientNet в качестве backbone

Следующая лекция

ИСП РАН

- Feature pyramids
- Модели для сегментации
- Модели для object detection