Введение в искусственный интеллект. Современное компьютерное зрение

Лекция 9. Методы сжатия и ускорения нейронных сетей

Бабин Д.Н., Иванов И.Е., Петюшко А.А.

кафедра Математической Теории Интеллектуальных Систем

10 декабря 2019

План лекции

- Прунинг
- Квантование
- Дистилляция знаний и совместное обучение
- Ручное построение архитектуры нейронной сети
- Методы автоматического поиска архитектуры (NAS)

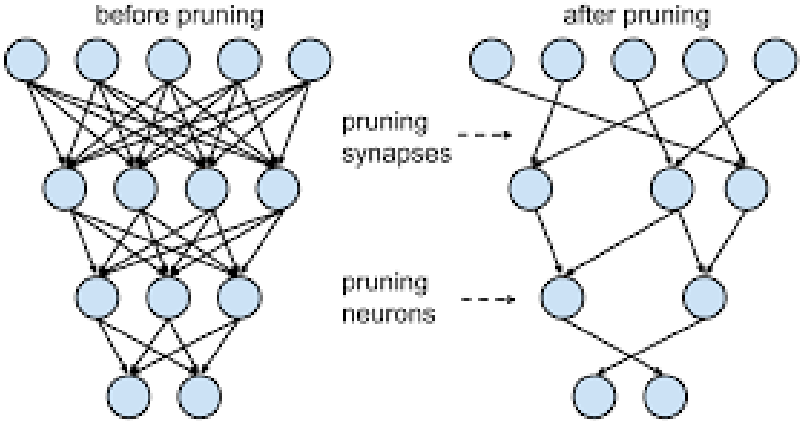
Постановка задачи и приложения

- Ускорение модели
- Оптимизация модели под конкретное устройство
- Сжатие модели

Прунинг

Определение

Прунинг — это удаление связей в нейронных сетях



Как выбирать удаляемые связи?

Ответ

Чтобы обнуление веса наименьшим образом влияло на качество, надо удалять наименее значимые связи, например, те, у которых наименьшие веса

Проблема удаления связей

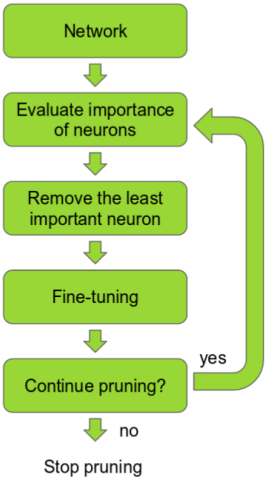
Проблема

При удалении связей может существенно падать точность модели

Возможные решения

- Для того, чтобы появлялось больше весов с маленьким весом, можно добавить ℓ_1/ℓ_2 - регуляризацию
- Производить дообучение модели после обнуления весов

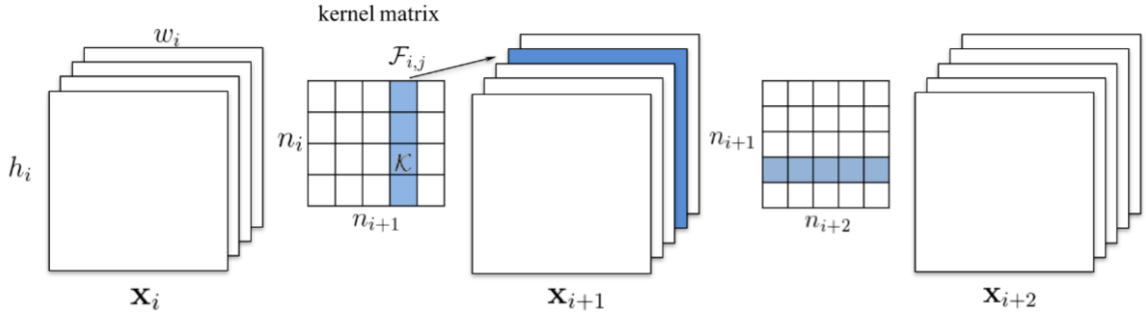
Типичная схема прунинга [1]



[1] https://arxiv.org/pdf/1611.06440.pdf

Прунинг карт [2]

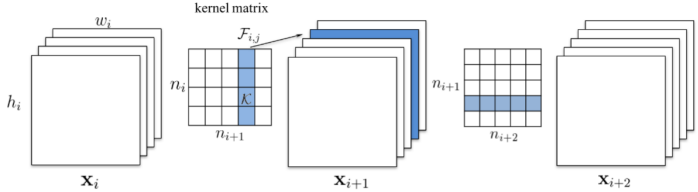
- С точки зрения вычислений выгоднее удалять не связи из графа, а нейроны целиком или карты или даже слои



[2] https://arxiv.org/pdf/1608.08710.pdf

Детали алгоритма

- Удаление карты происходит путем удаления весов с минимальной нормой весов (сумма квадратов или сумма абсолютных значений)
- Эта же норма добавляется к функции потерь с некоторым весом
- Удаление происходит постепенно и итеративно вместе с дообучением
- Критерий остановки либо достижение попрога по качесту модели, либо достижение необходимого вычислительного бюджета



Квантование

Определение

Уменьшение битности параметров нейронной сети

Интуиция

- При уменьшении битности параметров, вычисления можно проводить быстрее
- Существенно уменьшается размер модели

Методы квантования

- Обучение модели в полной точности, а потом округление до нужного количества бит
- Как правило после округления идет дообучение
- Есть методы позволяющие обучать сразу квантизованные модели

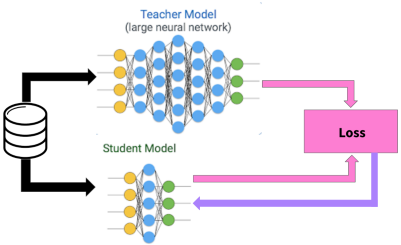
Бинарные нейронные сети

- Бинарные нейронные сети — экстремальный случай квантизации, когда веса имеют только два значения
- В таком виде обучение модели градиентном спуском затруднено, так как пространство дискретное
- Довольно полезная задача для индустрии, которая ждем своего решения

Дистилляция знаний [4]

Идея

Передать знания от Учителя (большая нейронная сеть или ансамбль моделей) к Студенту (маленькая нейронная сеть)



3

[3] https://towardsdatascience.com/knowledge-distillation-simplified-dd4973dbc764

[4] CristianBucila,RichCaruana,andAlexandruNiculescu-Mizil.ModelCompression.KDD,2006

Дистилляция знаний: интуиция

Допущение

Мы будем рассматривать **классификационную** нейронную сеть

Вопрос

Почему использование учителя может работать лучше, чем просто обучение?

Ответ

- В задаче классификации разметка осуществляется путем выбора одного наиболее подходящего класса. Обученная нейронная сеть же выдаёт распределение по классам, которое содержит дополнительную информацию
- Можно расширить обучающий датасет, используя выход Учителя, как аннотацию

Дистилляция знаний: функция потерь [5]

Softmax с температурой

$$q_i = \frac{exp(\frac{z_i}{T})}{\sum_j exp(\frac{z_j}{T})}$$

Свойства

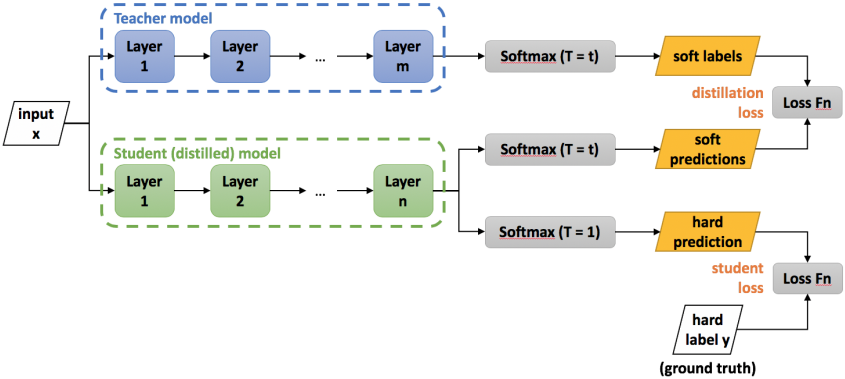
- При $T = 1$ обычный Softmax
- При $T \to \infty$ вероятности будут всё больше выравниваться
- Где в промежутке станут доступны для обучения дополнительная информация из распределения (сам Хинтон называл это "dark knowledge")

$$\mathcal{L}(x; W) = \alpha * \mathcal{H}(y, \sigma(z_s; T = 1)) + \beta * \mathcal{H}(\sigma(z_t; T = \tau), \sigma(z_s, T = \tau))$$

[5] https://arxiv.org/abs/1503.02531

Дистилляция знаний

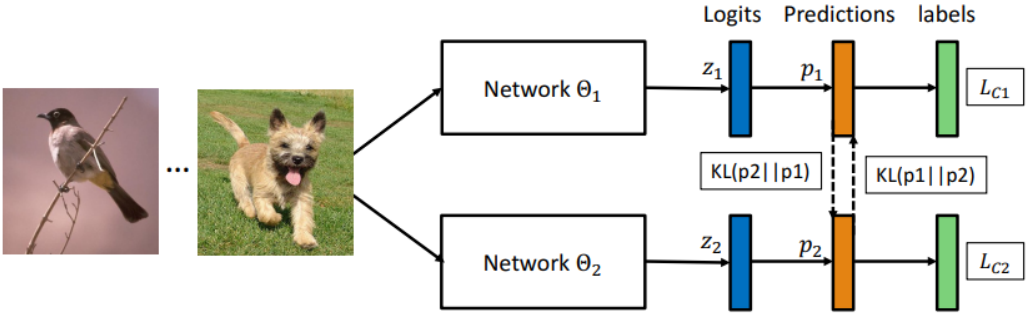
$$\mathcal{L}(x;W) = \alpha * \mathcal{H}(y, \sigma(z_s; T=1)) + \beta * \mathcal{H}(\sigma(z_t; T=\tau), \sigma(z_s, T=\tau))$$



Совместное обучение [6]

Идея

Учиться вместе помогает достичь более высоких результатов



[6] https://arxiv.org/abs/1706.00384

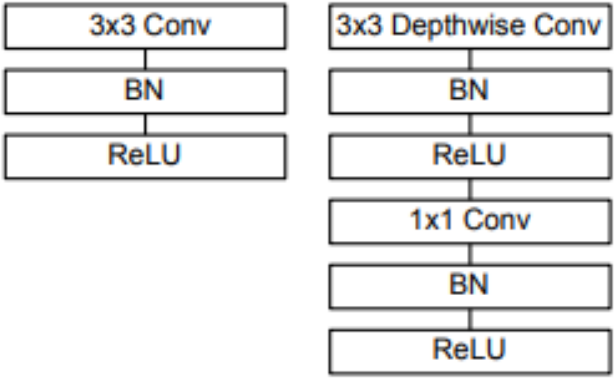
Эффективные архитектуры

- MobileNet
- ShuffleNet
- EfficientNet

MobileNet [7]

Идея

Замена всёрток 3×3s на depth-wise separable convolution



[7] https://arxiv.org/abs/1704.04861

Архитектура MobileNet

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	$1 \times 1 \times 1024$

Table 2. Resource Per Layer Type

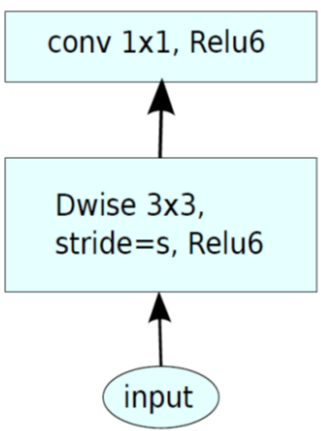
Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

MobileNet: Результаты

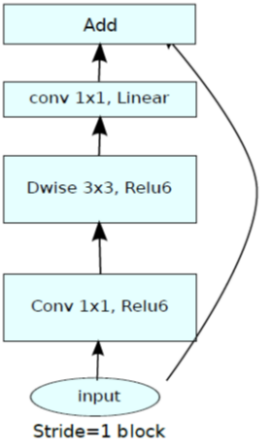
Table 4. Depthwise Separable vs Full Convolution MobileNet

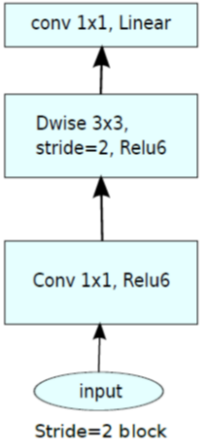
Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

MobileNetV2: Новые блоки [8]



MobileNetV1





MobileNetV2

[8] https://arxiv.org/abs/1801.04381

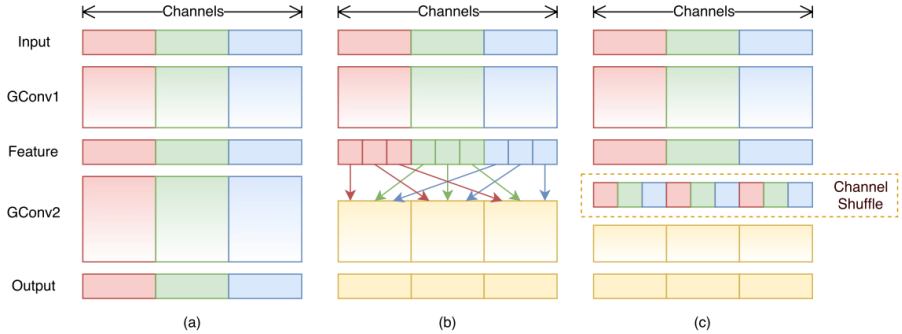
MobileNetV2: Архитектура

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	-	

ShuffleNet [9]

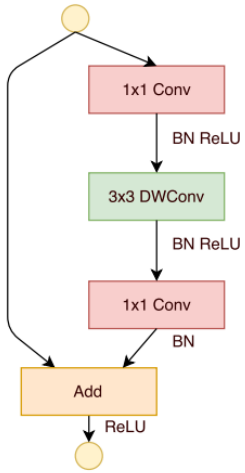
Идея

Использование групповой свёртки и операцию перемешивания (Shuffle)

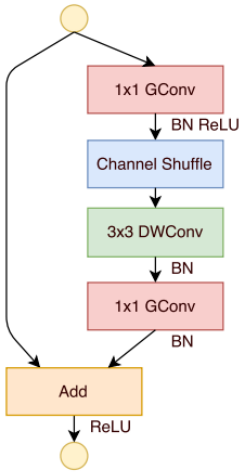


[9] https://arxiv.org/abs/1707.01083

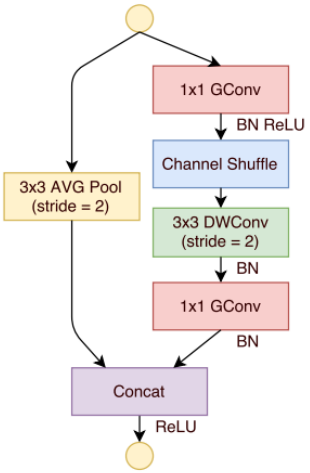
Блоки ShuffleNet



(a)



(b)



(c)

Архитектура ShuffleNet

Table 1: ShuffleNet architecture

Layer	Output size	KSize	Stride	Repeat	Output channels (g groups)				
					$g = 1$	$g = 2$	$g = 3$	$g = 4$	$g = 8$
Image	224×224				3	3	3	3	3
Conv1	112×112	3×3	2	1	24	24	24	24	24
MaxPool	56×56	3×3	2						
Stage2[1]	28×28		2	1	144	200	240	272	384
	28×28		1	3	144	200	240	272	384
Stage3	14×14		2	1	288	400	480	544	768
	14×14		1	7	288	400	480	544	768
Stage4	7×7		2	1	576	800	960	1088	1536
	7×7		1	3	576	800	960	1088	1536
GlobalPool	1×1	7×7							
FC					1000	1000	1000	1000	1000
Complexity[2]					143M	140M	137M	133M	137M

ShuffleNet: Результаты

Table 2: Classification error vs. number of groups g (*smaller number represents better performance*)

Model	Complexity (MFLOPs)	Classification error (%)				
		$g = 1$	$g = 2$	$g = 3$	$g = 4$	$g = 8$
ShuffleNet 1×	140	35.1	34.2	**34.1**	34.3	34.7
ShuffleNet 0.5×	38	46.1	45.1	44.4	**43.7**	43.8
ShuffleNet 0.25×	13	56.7	56.3	55.6	54.5	**53.7**
ShuffleNet 0.5× (arch2)	40	45.7	44.3	43.8	43.2	**42.7**
ShuffleNet 0.25× (arch2)	13	56.5	55.3	55.5	54.3	**53.3**

Model	Cls err. (%, no shuffle)	Cls err. (%, shuffle)	Δ err. (%)
ShuffleNet 1x ($g = 3$)	34.5	**32.6**	1.9
ShuffleNet 1x ($g = 8$)	37.6	**32.4**	5.2
ShuffleNet 0.5x ($g = 3$)	45.7	**43.2**	2.5
ShuffleNet 0.5x ($g = 8$)	48.1	**42.3**	5.8
ShuffleNet 0.25x ($g = 3$)	56.3	**55.0**	1.3
ShuffleNet 0.25x ($g = 8$)	56.5	**52.7**	3.8

ShuffleNet: Сравнение с другими моделями

Table 6: Complexity comparison

Model	Cls err. (%)	Complexity (MFLOPs)
VGG-16 [27]	28.5	15300
ShuffleNet $2\times$ ($g = 3$)	29.1	**524**
PVANET [18] (*our impl.*)	35.3	557
ShuffleNet $1\times$ ($g = 3$)	34.1	**140**
AlexNet [19]	42.8	720
SqueezeNet [13]	42.5	833
ShuffleNet $0.5\times$ (arch2, $g = 8$)	42.7	**40**

ShuffleNet: Сравнение с другими моделями

Table 5: ShuffleNet vs. MobileNet [12] on ImageNet Classification

Model	Complexity (MFLOPs)	Cls err. (%)	Δ err. (%)
1.0 MobileNet-224	569	29.4	-
ShuffleNet 2× ($g = 3$)	524	**29.1**	0.3
0.75 MobileNet-224	325	31.6	-
ShuffleNet 1.5× ($g = 3$)	292	**31.0**	0.6
0.5 MobileNet-224	149	36.3	-
ShuffleNet 1× ($g = 3$)	140	**34.1**	2.2
0.25 MobileNet-224	41	49.4	-
ShuffleNet 0.5× (arch2, $g = 8$)	40	**42.7**	6.7
ShuffleNet 0.5× (shallow, $g = 3$)	40	45.2	4.2

Сравнение моделей

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Этапы развития искусственного интеллекта

Поколение 1: Старый добрый ИИ

- Конструирование алгоритмов под конкретные задачи
- Отсутствие обучения

Поколение 2: Машинное обучение

- Конструирование признаков
- Обучение модели на готовых признаках

Этапы развития искусственного интеллекта

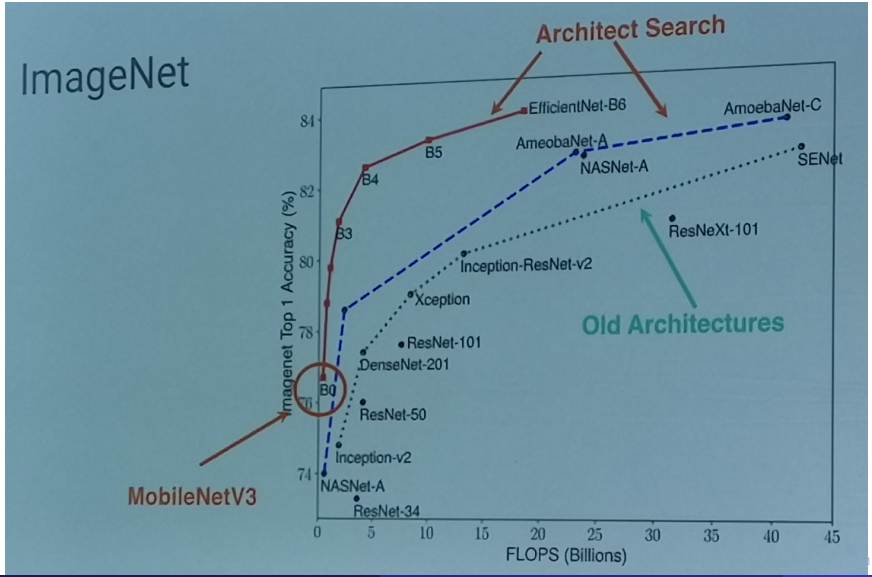
Поколение 3: Глубокое обучение

- Обучаемые признаки
- Обучаемые алгоритмы
- Гиперпараметры подбираются вручную

Поколение 4: Метаобучение

- Никакой ручной работы
- Всё обучаемое

NAS: Автоматический поиск архитектур



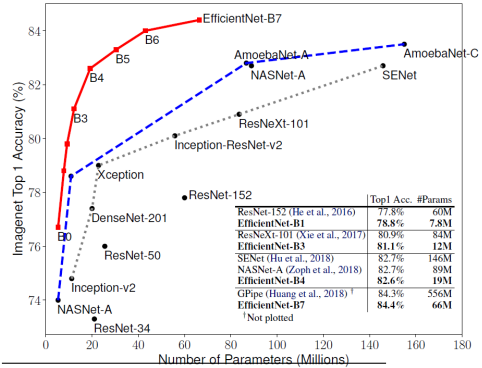
NAS: Схема [10]

[10] https://towardsdatascience.com/neural-architecture-search-nas-the-future-of-deep-learning-c99356351136

EfficientNet [11]

Идея

- Автоматический поиск сравнительно небольшой модели
- Эффективное масштабирование модели



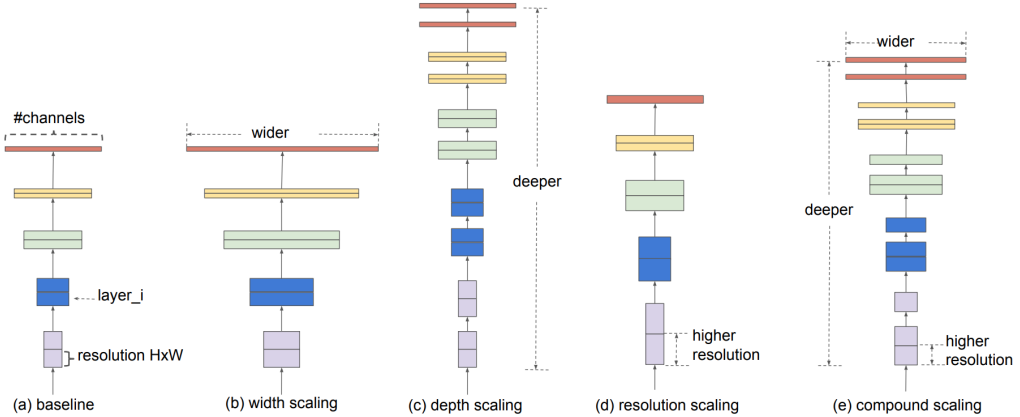
	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

[11] https://arxiv.org/abs/1905.11946

EfficientNet: Масштабирование модели

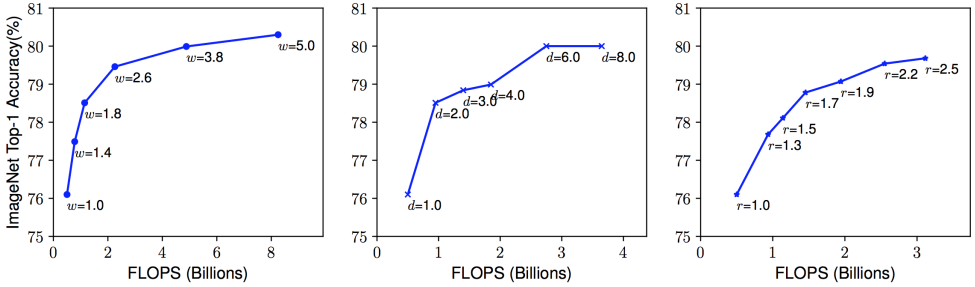
EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks



EfficientNet: Масштабирование модели

Масштабирование

Чем больше модель, тем меньше помагает мастабирование



EfficientNet: Масштабирование модели

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

Шаг 1

Для $\phi = 1.0$ параметры подбираются перебором по сетке:
$\alpha = 1.2,\ \beta = 1.1\ \gamma = 1.15$

Шаг 2

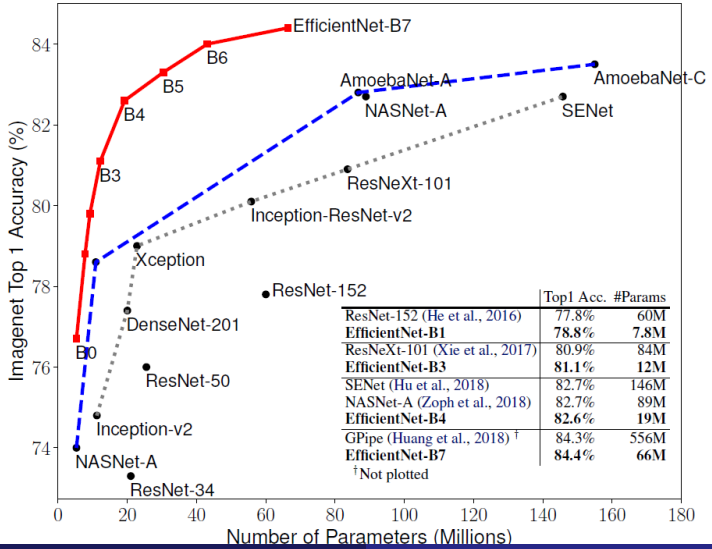
При фиксированных $\alpha = 1.2,\ \beta = 1.1\ \gamma = 1.15$, меняя ϕ, получаем большие модели

EfficientNet: Результаты

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPS	Ratio-to-EfficientNet
EfficientNet-B0	**76.3%**	**93.2%**	**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	**78.8%**	**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	**79.8%**	**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	**81.1%**	**95.5%**	**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
EfficientNet-B4	**82.6%**	**96.3%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.3%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.9%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.4%**	**97.1%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

We omit ensemble and multi-crop models (Hu et al., 2018), or models pretrained on 3.5B Instagram images (Mahajan et al., 2018).

EfficientNet: Результаты



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

Заключение

- Ускорение моделей и поиск более эффективных архитектур — тренд современного компьютерного зрения
- Очень часто для ускорения моделей используют сразу несколько способов
- Эпоха глубокого обучения подходит к концу, NAS открывает новую эпоху в искусственном интеллекте
- Учите математику, чтобы NAS не лишил вас работы

Спасибо за внимание!