EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Валерий Лобов

16 декабря 2020 г.

1 Введение

Масштабирование (*scaling)* нейронных сетей становится все более актуальным вопросом в сфере глубокого обучения. При правильном скейлинге нейросеть может показывать более высокую эффективность за счет увеличения сложности архитектуры или же наоборот, качественный скейлинг может сделать нейросеть более легковесной, более универсальной для применения на непроизводительных устройствах, при этом несильно теряя в целевых показателях. Прежде всего, нужно ответить, как масштабирование работает в случае сверточных нейронных сетей.

2 Масштабирование сверточных сетей

В архитектуре сверточных нейросетей есть три ключевые размерности, которые можно изменять: глубина (*depth*) - количество слоев, ширина (*width*) - подразумевает ширину карты фильтров в сверточных слоях, и входное разрешение (*resolution*) - разрешение входного изображения.

Увеличение глубины нейросети - классический способ масштабирования сверточных сетей. Оно может повлиять на генерацию более сложных признаков, что по идее должно улучшать целевые метрики. Но на практике исследователи сталкиваются с проблемой затухающих градиентов (*vanishing gradients*), что делает сильное масштабирование сверточных сетей нецелосообразным. К примеру, ResNet-1000 имеет

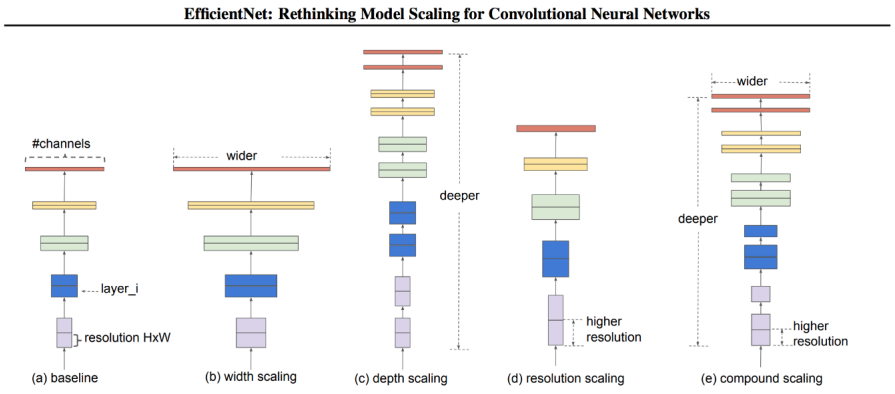

Рис. 1: Примеры возможного масштабирования

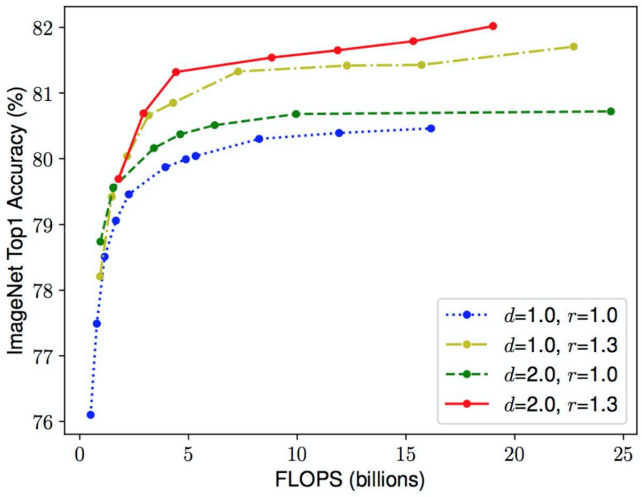

Рис. 2: Масштабирование ширины при заданных значениях глубины и разрешения

такую же точность, как и ResNet-100. Поэтому, можно предположить, что масштабирование по глубине нужно делать более осознанно.

Изменение ширины сверточной сети может повлиять на способность определять более тонкие паттерны на входном изображении, что естественным образом влияет на сложность обучения и его длительность. Если комбинировать вместе масштабирование ширины и глубины, возникнут дополнительные трудности. К примеру, при увеличении ширины и уменьшении глубины целевые метрики существенно падают.

Осталось рассмотреть третью размерность масштабирования. Для более сложных задач, к примеру, детекция объектов на изображениях, необходимо подавать на вход изображения с более высоким разрешением, для того чтобы нейросеть могла распознавать более тонкие детали. Но сложно утверждать, по какому правилу нужно масштабировать это измерение для улучшения качества предсказания.

3 Предложенный метод

Авторы статьи представили эффективный алгоритм масштабирования сверточных сетей. Прежде всего, было показано, что масштабируемые размерности не являются независимыми. К примеру, на Рисунке 2 авторы провели результаты эксперимента, в котором показали, что изменение ширины нейросети показывает лучшие метрики качества при одновременном увеличении глубины и разрешения при одном и том же значении FLOPS.

Авторы пришли к выводу: для достижения наилучшего качества сверточная сеть должна масштабироваться одновременно в трех размерностях. Представленный метод *compound scaling* использует простую и эффективную технику для равномерного масштабирования размерностей, зависящая от единственного коэффициента ϕ

согласно правилу:

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

где α, β, γ это константы, которые могут быть определены с помощью небольшого поиска по сетке. ϕ это определенный пользователем коэффициент, который контролирует используемые нейросетью ресурсы. Отметим, что FLOPS регулярной операции свертки пропорционален d, w^2, r^2

Авторы разработали базовую модель *EfficientNet-B0* благодаря нейросетевому поиску архитектур (*neural architecture search*), где архитектурный поиск оптимизирует одновременно Accuracy и FLOPS. В качестве целевой оптимизируемой функции используется $ACC(m) \cdot [FLOPS(m)/T]^w$, где ACC(m) и FLOPS(m) означают accuracy модели m и FLOPS модели m, T это таргет FLOPS и $w = -0.07$ - это гиперпараметр для контролирования trade-off между accuracy и FLOPS. В отличие от других статей из данной сферы здесь оптимизируется FLOPS, а не время задержки прямого прохода нейросети (*latency*), так как авторы не берут в расчет определенное устройство, а рассматривают количество операций с плавающей точкой в целом.

На первый взгляд кажется нетривиальным подбирать сразу четыре коэффициента $\alpha, \beta, \gamma, \phi$, но авторы помимо прочего предложили несложный алгоритм поиска этих коэффициентов, благодаря которому возможно сузить пространство поиска и сделать операцию поиска менее вычислительно трудоемкой.

1. Фиксируем $\phi = 1$, делаем поиск гиперпараметров α, β, γ по небольшой сетке. Для базовой модели *EfficientNet-B0* оказалось, что оптимальными значениями являются $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$ такие, что $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$.

2. Далее найденные значения α, β, γ фиксируются, и производится подбор гиперпараметра ϕ. В зависимости от значений ϕ формируются архитектуры *EfficientNet* с идексом от B1 до B7.

4 Результаты и выводы

Сводная таблица с результатами, изображенная на Рисунке 3, показывает, что авторам удалось создать такие архитектуры, которые в сравнении с другими известными существующими решениями показывают более высокую точность, при этом используя меньшее количество FLOPS. В особенности, архитектура *EfficientNet-B7* достигает state-of-the-art результатов 84.3% top-1 accuracy на датасете ImageNet, но имея в 8.4 раза меньше параметров и работая в 6.1 раз быстрее чем архитектура из лучшего алгоритма нейросетевого поиска на тот момент.

В данной статье авторам удалось создать алгоритм поиска нейросетевых архитектур, который ищет не только наиболее точные сверточные нейронные сети, но и эффективные с точки зрения используемых ресурсов.

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPS	Ratio-to-EfficientNet
EfficientNet-B0	**77.3%**	**93.5%**	**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	**79.2%**	**94.5%**	**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	**80.3%**	**95.0%**	**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	**81.7%**	**95.6%**	**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	**83.0%**	**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	**83.7%**	**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	**84.2%**	**96.8%**	**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).

Рис. 3: Итоговая таблица с показателями метрик EfficientNet в сравнении с другими архитектурами