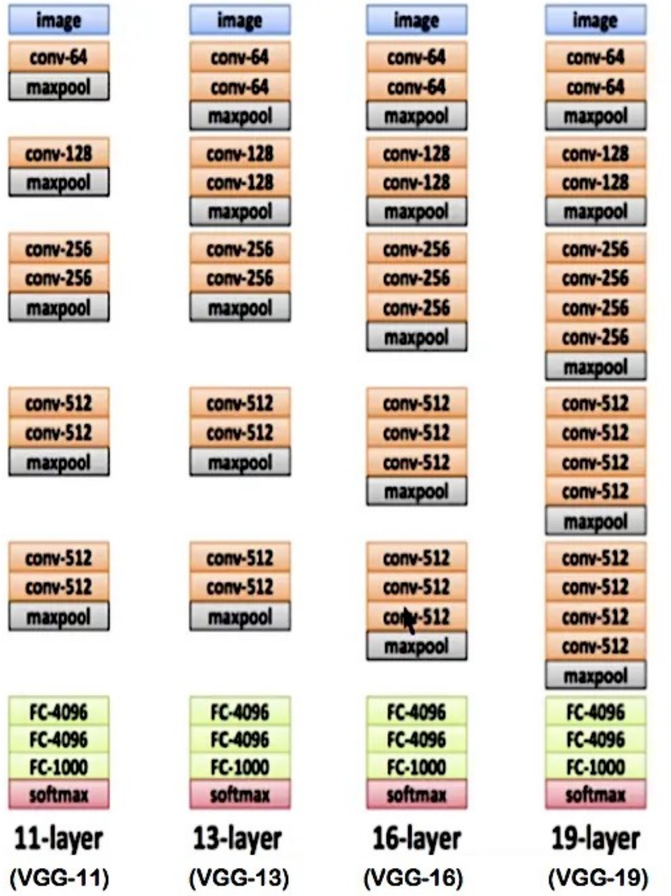
RepVGG

Предпосылки
Feed-forward models

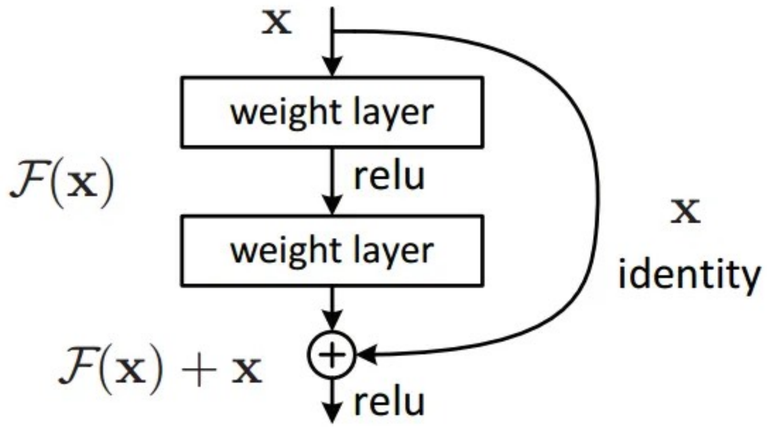
- Достаточно быстрые:
  - 3x3 conv (aka Winograd conv)
  - Feed-forward
- Улучшение метрик происходило увеличением количества слоёв

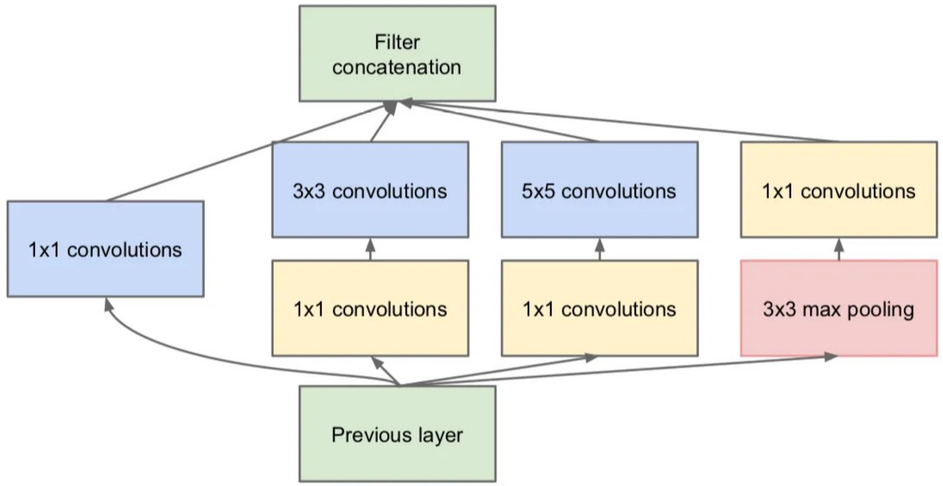


Предпосылки

Multi-branch models

- Меньшее количество параметров при лучшем качестве
- Теоретически выше скорость



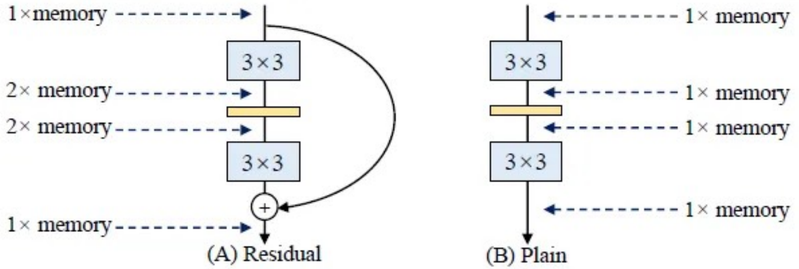


Comparison					
Network	Year	Salient Feature	top5 accuracy	Parameters	FLOP
AlexNet	2012	Deeper	84.70%	62M	1.5B
VGGNet	2014	Fixed-size kernels	92.30%	138M	19.6B
Inception	2014	Wider - Parallel kernels	93.30%	6.4M	2B
ResNet-152	2015	Shortcut connections	95.51%	60.3M	11B

Предпосылки

Почему multi-branch не так хороши, как кажется?

- Скорость
  - Не учитывается memory-acces cost
  - Степень параллелизма: объединение операций в блоки влечет за собой расходы GPU на запуск ядер и их синхронизацию
  - Winograd convs: теоретическая вычислительная плотность 3×3 conv примерно в 4 раза выше, чем у остальных, что говорит о том, что общее количество теоретических FLOP не является сопоставимым показателем
- Память
  - Результаты каждой ветви необходимо хранить до сложения или конкатенации, что значительно увеличивает пиковое значение занимаемой памяти
  - Видно, что вход в residual блок необходимо хранить до момента сложения. Блок сохраняет размер карты признаков, то пиковое значение дополнительной занимаемой памяти составляет 2× от входной.



RepVGG

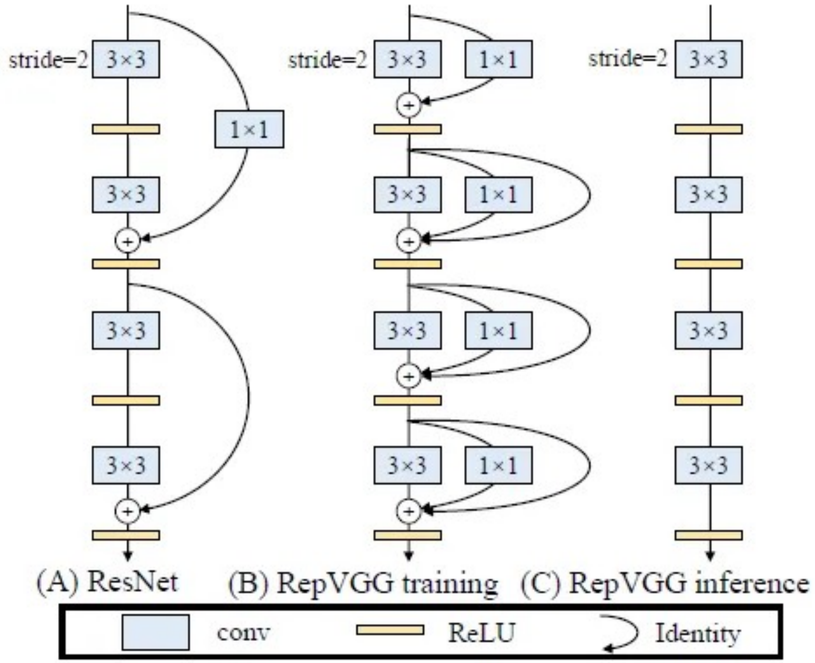
Идея

- Использовать feed-forward топологию, без ветвей. Как следствие меньшие расходы на память и MAC
- Использовать только «дешёвые» 3x3 conv, ReLU, BatchNorm
- Конкретная архитектура без автоматического поиска

RepVGG

Архитектура при обучении

- Используем multi-branch:
  - Такая архитектура превращает модель в ансамбль моделей
  - Ансамбль 3^n моделей
- Три ветви:
  - 3x3 conv
  - 1x1 conv
  - Identity используем когда размеры in и out одинаковые



RepVGG

Репараметризация

- W(3) размера C2×C1×3×3 обозначает ядро слоя 3×3 с входными каналами C1 и выходными каналами C2, а W(1) размера C2×C1 - ядро ветви 1×1.
- μ(3), σ(3), γ(3), β(3) - накопленные mean, std и выученные scaling factor и bias BatchNorm, следующего за 3×3 conv.
- μ(1), σ(1), γ(1), β(1) - аналогично для параметров BN, следующего за 1×1 conv, а μ(0),(0), γ(0), β(0) - для Identity.
- Пусть M(1) имеет размер N×C1×H1×W1, а M(2) - размер N×C2×H2×W2, которые являются входом и выходом соответственно, и пусть * - оператор свертки.

RepVGG

Репараметризация

$$\begin{aligned}\mathrm{M}^{(2)} &= \mathrm{bn}(\mathrm{M}^{(1)} * \mathrm{W}^{(3)}, \boldsymbol{\mu}^{(3)}, \boldsymbol{\sigma}^{(3)}, \boldsymbol{\gamma}^{(3)}, \boldsymbol{\beta}^{(3)}) \\ &+ \mathrm{bn}(\mathrm{M}^{(1)} * \mathrm{W}^{(1)}, \boldsymbol{\mu}^{(1)}, \boldsymbol{\sigma}^{(1)}, \boldsymbol{\gamma}^{(1)}, \boldsymbol{\beta}^{(1)}) \\ &+ \mathrm{bn}(\mathrm{M}^{(1)}, \boldsymbol{\mu}^{(0)}, \boldsymbol{\sigma}^{(0)}, \boldsymbol{\gamma}^{(0)}, \boldsymbol{\beta}^{(0)}) .\end{aligned}$$

Результат при inference

$$\mathrm{bn}(\mathrm{M}, \boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\gamma}, \boldsymbol{\beta})_{:,i,:,:} = (\mathrm{M}_{:,i,:,:} - \boldsymbol{\mu}_i)\frac{\boldsymbol{\gamma}_i}{\boldsymbol{\sigma}_i} + \boldsymbol{\beta}_i .$$

BatchNorm при inference

RepVGG

Репараметризация

- Можно преобразовать ядра свертки с уже примененным BatchNorm:

$$\mathrm{W}'_{i,:,:,:} = \frac{\gamma_i}{\sigma_i}\mathrm{W}_{i,:,:,:}\,, \quad \mathrm{b}'_i = -\frac{\mu_i\gamma_i}{\sigma_i} + \beta_i\,.$$

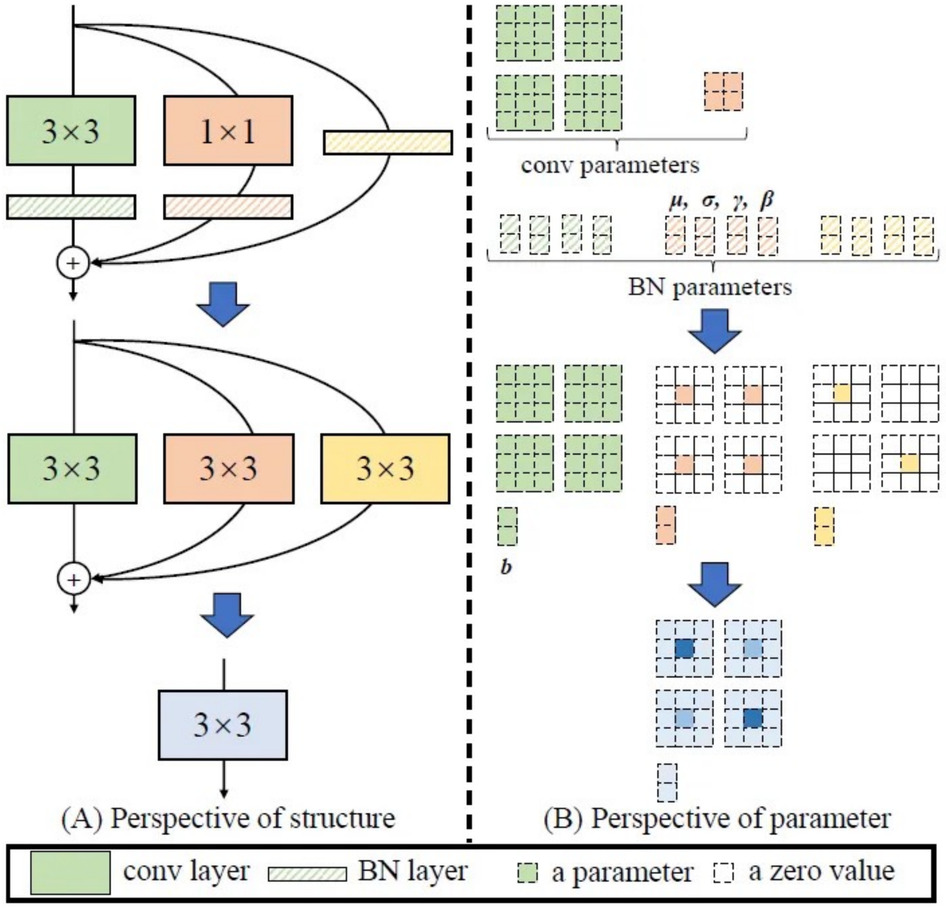
- Следовательно inference сводится к:

$$\mathrm{bn}(\mathrm{M} * \mathrm{W}, \mu, \sigma, \gamma, \beta)_{:,i,:,:} = (\mathrm{M} * \mathrm{W}')_{:,i,:,:} + \mathrm{b}'_i\,.$$

RepVGG

Branch-merging

- Это преобразование применимо и к ветви Identity, поскольку можно рассматривать как свертку 1×1 с I в качестве ядра.
- После таких преобразований мы получим одно ядро 3×3, два ядра 1×1 и три вектора смещения.
- Затем путем сложения трех векторов смещения получаем итоговое смещение.
- И окончательное ядро 3×3 путем сложения ядер 1×1 в центральной точке ядра 3×3, что можно легко реализовать, если сначала привести два ядра 1×1 к нулю в 3×3 и сложить эти три ядра, как показано на рисунке выше.



RepVGG

Архитектура

Stage	Output size	RepVGG-A	RepVGG-B
1	112×112	$1 \times \min(64, 64a)$	$1 \times \min(64, 64a)$
2	56×56	$2 \times 64a$	$4 \times 64a$
3	28×28	$4 \times 128a$	$6 \times 128a$
4	14×14	$14 \times 256a$	$16 \times 256a$
5	7×7	$1 \times 512b$	$1 \times 512b$

RepVGG

Эксперименты

Model	Top-1 acc	Speed	Params (M)	Theo FLOPs (B)	Wino MULs (B)
RepVGG-A0	72.41	3256	8.30	1.4	0.7
ResNet-18	71.16	2442	11.68	1.8	1.0
RepVGG-A1	74.46	2339	12.78	2.4	1.3
RepVGG-B0	75.14	1817	14.33	3.1	1.6
ResNet-34	74.17	1419	21.78	3.7	1.8
RepVGG-A2	76.48	1322	25.49	5.1	2.7
RepVGG-B1g4	77.58	868	36.12	7.3	3.9
EfficientNet-B0	75.11	829	5.26	0.4	-
RepVGG-B1g2	77.78	792	41.36	8.8	4.6
ResNet-50	76.31	719	25.53	3.9	2.8
RepVGG-B1	78.37	685	51.82	11.8	5.9
RegNetX-3.2GF	77.98	671	15.26	3.2	2.9
RepVGG-B2g4	78.50	581	55.77	11.3	6.0
ResNeXt-50	77.46	484	24.99	4.2	4.1
RepVGG-B2	78.78	460	80.31	18.4	9.1
ResNet-101	77.21	430	44.49	7.6	5.5
VGG-16	72.21	415	138.35	15.5	6.9
ResNet-152	77.78	297	60.11	11.3	8.1
ResNeXt-101	78.42	295	44.10	8.0	7.9