RepVGG

2021 CVPR

RepVGG: Making VGG-style ConvNets Great Again

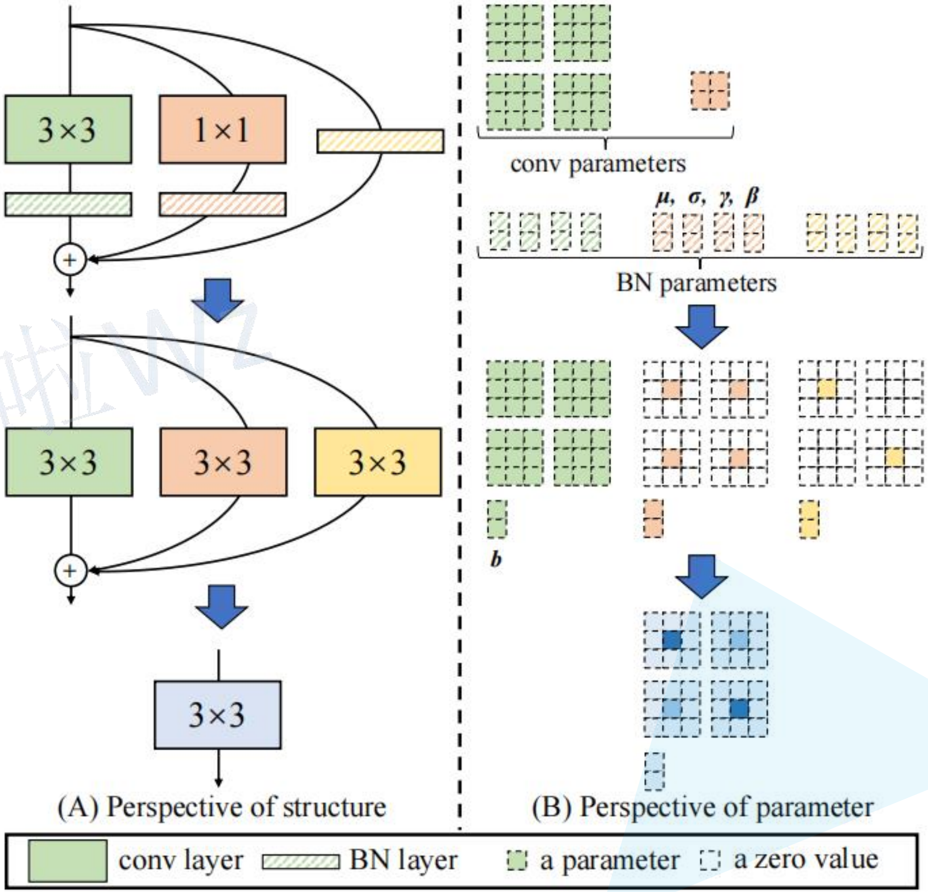
结构重参数化

Xiaohan Ding [1*] Xiangyu Zhang [2] Ningning Ma [3]
Jungong Han [4] Guiguang Ding [1†] Jian Sun [2]
[1] Beijing National Research Center for Information Science and Technology (BNRist);
School of Software, Tsinghua University, Beijing, China
[2] MEGVII Technology
[3] Hong Kong University of Science and Technology
[4] Computer Science Department, Aberystwyth University, SY23 3FL, UK
dxh17@mails.tsinghua.edu.cn zhangxiangyu@megvii.com nmaac@cse.ust.hk
jungonghan77@gmail.com dinggg@tsinghua.edu.cn sunjian@megvii.com

论文下载地址：https://arxiv.org/abs/2101.03697
博文：https://blog.csdn.net/qq_37541097/article/details/125692507
公众号“阿喆学习小记”输入RepVGG获取

RepVGG

目录



3×3
1×1
3×3
3×3
3×3
3×3
conv parameters
μ, σ, γ, β
BN parameters
b
(A) Perspective of structure
(B) Perspective of parameter
conv layer
BN layer
a parameter
a zero value

RepVGG

前言

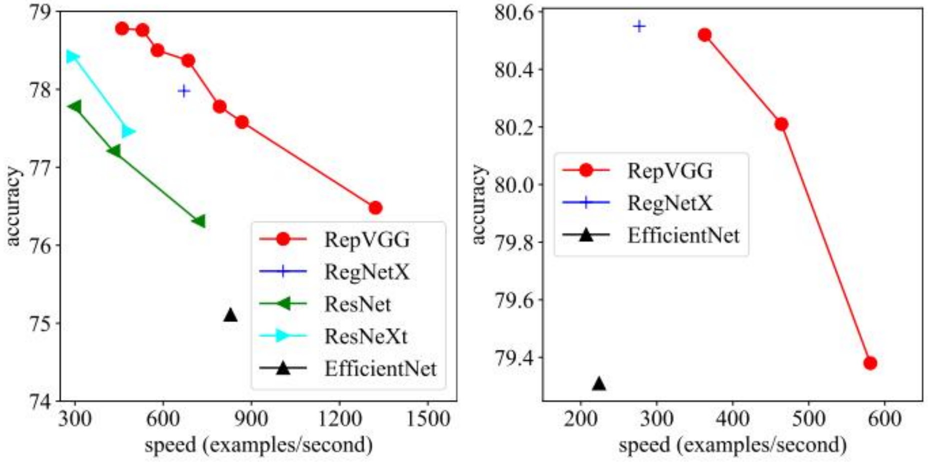
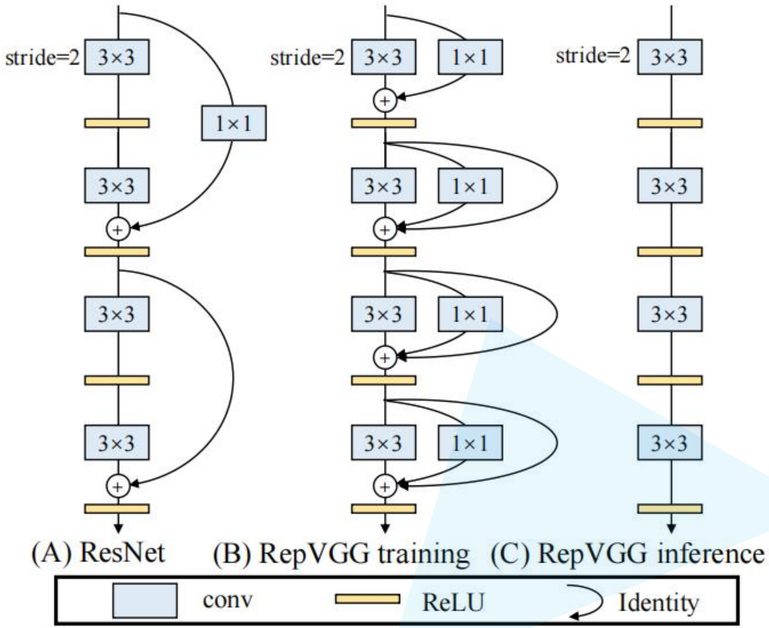


Figure 1: Top-1 accuracy on ImageNet *vs*. actual speed. Left: lightweight and middleweight RepVGG and baselines trained in 120 epochs. Right: heavyweight models trained in 200 epochs. The speed is tested on the same 1080Ti with a batch size of 128, full precision (fp32), single crop, and measured in examples/second. The input resolution is 300 for EfficientNet-B3 [35] and 224 for the others.

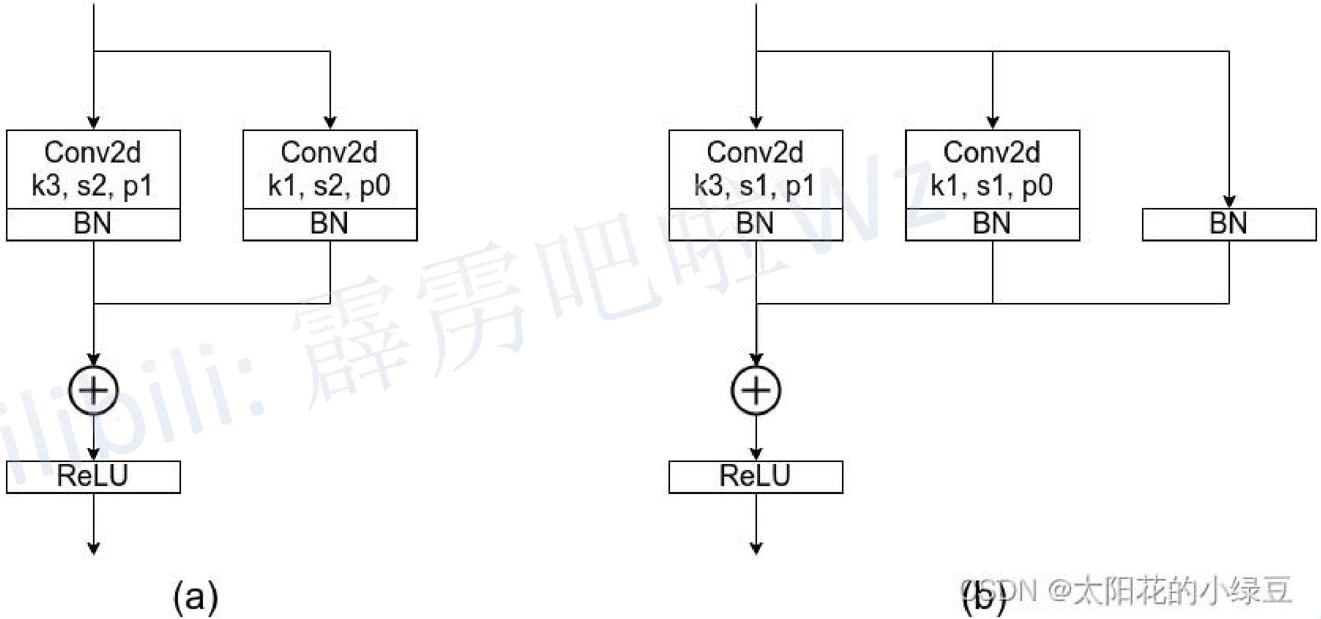
structural re-parameterization technique



RepVGG

RepVGG Block详解

Training mode



RepVGG Block详解

为什么训练时要采用多分支结构？

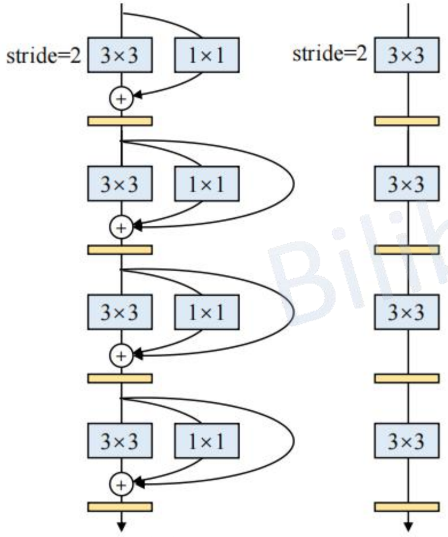
Table 6: Ablation studies with 120 epochs on RepVGG-B0. The inference speed w/o re-param (examples/s) is tested with the models before conversion (batch size=128). Note again that all the models have the same final structure.

Identity branch	1×1 branch	Accuracy	Inference speed w/o re-param
		72.39	1810
✓		74.79	1569
	✓	73.15	1230
✓	✓	**75.14**	1061

RepVGG Block详解

为什么推理时要将多分支模型转换成单路模型？

1. 更快
2. 省内存
3. 更加灵活



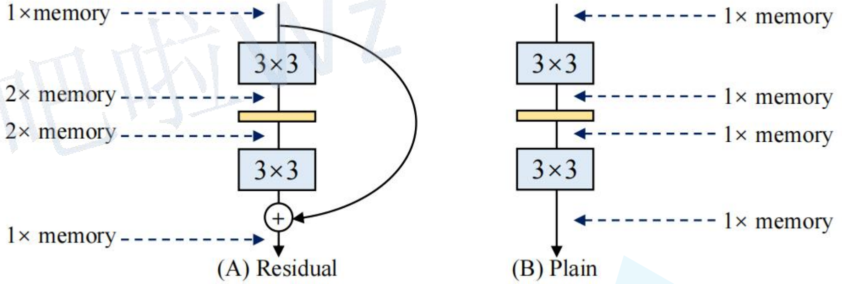


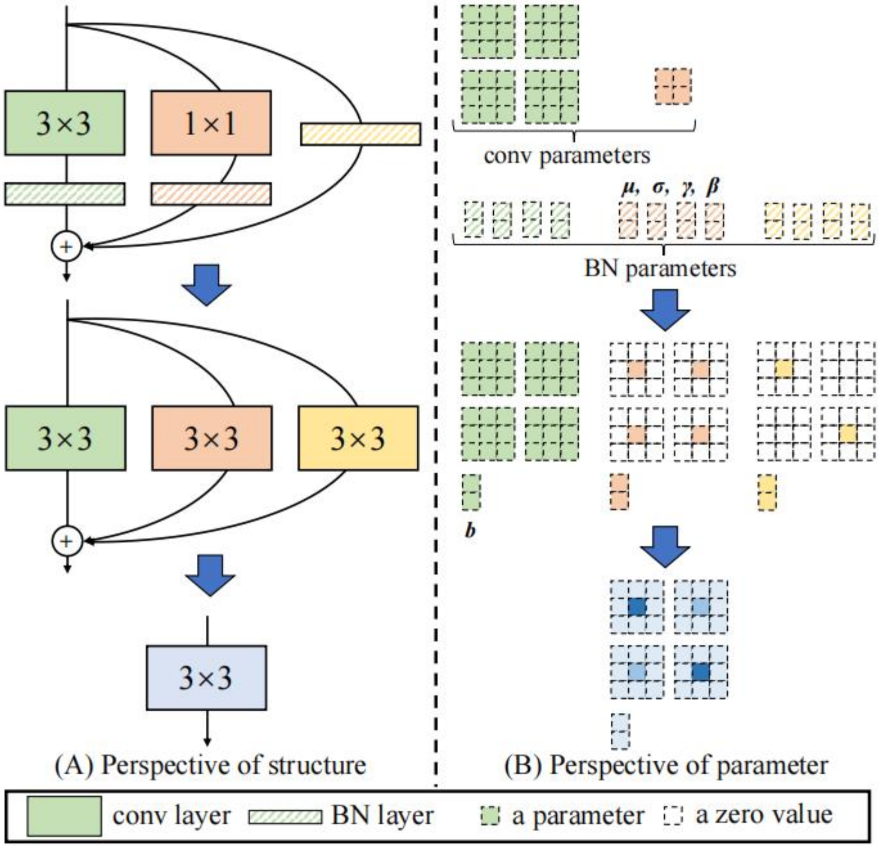
Figure 3: Peak memory occupation in residual and plain model. If the residual block maintains the size of feature map, the peak value of extra memory occupied by feature maps will be 2× as the input. The memory occupied by the parameters is small compared to the features hence ignored.

RepVGG

结构重参数化

in_channels=2

out_channels=2



Conv2d(no bias)+BN

RepVGG

结构重参数化-融合Conv2d和BN

第i个通道的值

$$bn(M,\mu,\sigma,\gamma,\beta)_{:,i,:,:} = (M_{:,i,:,:} - \mu_i)\frac{\gamma_i}{\sigma_i} + \beta_i$$

$$y_i = \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \cdot \gamma_i + \beta_i$$

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

第i个卷积核的权重

RepVGG

结构重参数化-融合Conv2d和BN

Channel:1

x_1^1	x_2^1	x_3^1
x_4^1	x_5^1	x_6^1
x_7^1	x_8^1	x_9^1

k_1^1	k_2^1	k_3^1
k_4^1	k_5^1	k_6^1
k_7^1	k_8^1	k_9^1

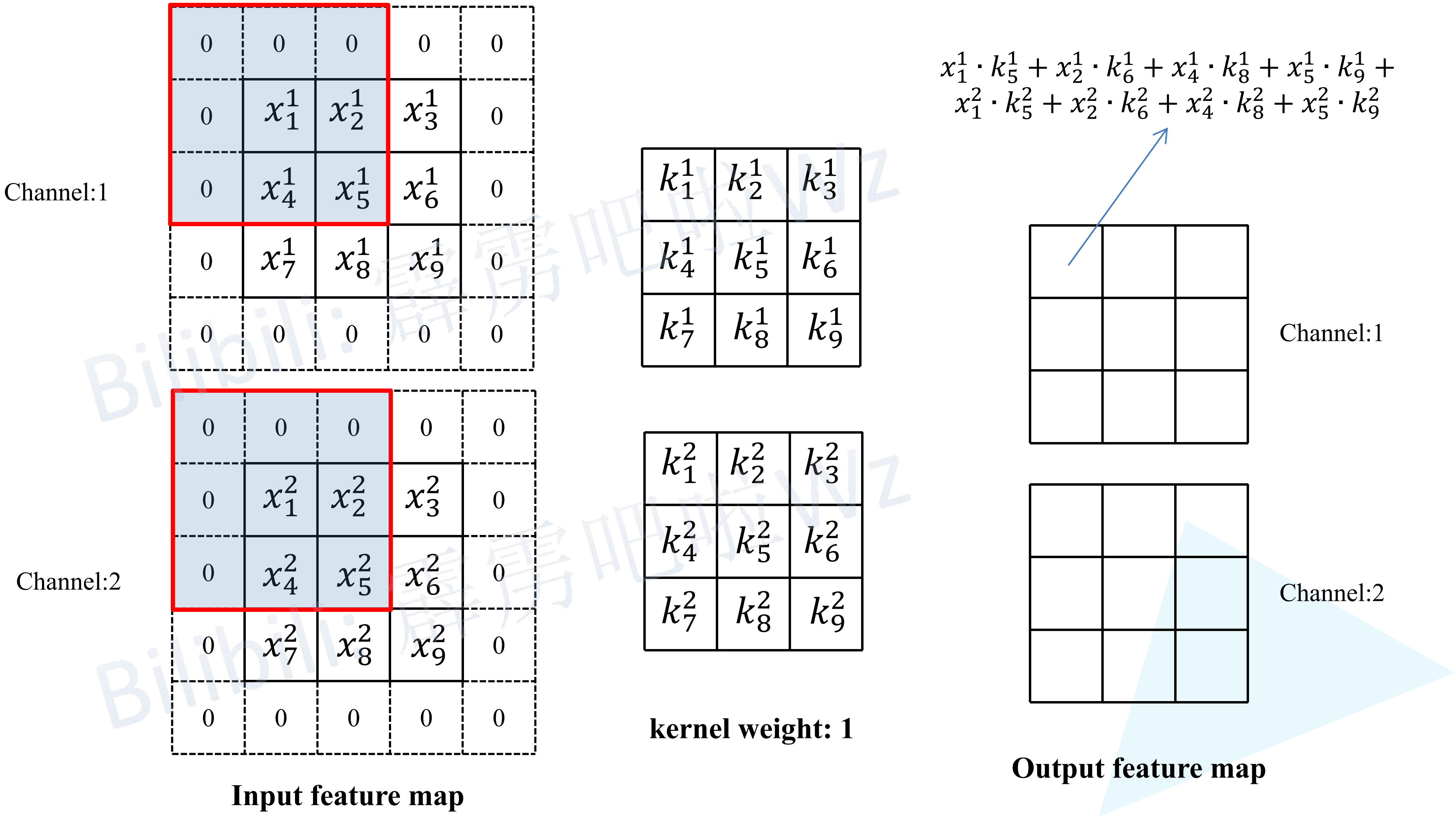
Channel:2

x_1^2	x_2^2	x_3^2
x_4^2	x_5^2	x_6^2
x_7^2	x_8^2	x_9^2

k_1^2	k_2^2	k_3^2
k_4^2	k_5^2	k_6^2
k_7^2	k_8^2	k_9^2

Input feature map

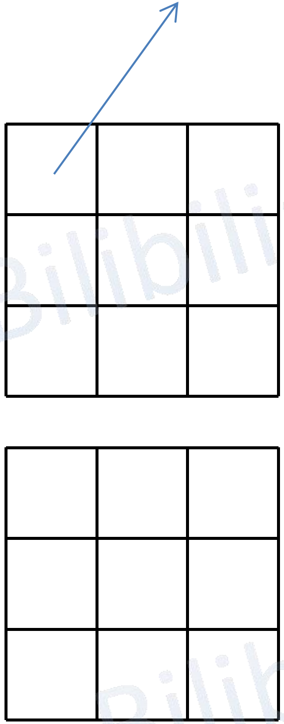
kernel weight: 1



Channel:1
Channel:2
Input feature map
kernel weight: 1
$x_1^1 \cdot k_5^1 + x_2^1 \cdot k_6^1 + x_4^1 \cdot k_8^1 + x_5^1 \cdot k_9^1 +$
$x_1^2 \cdot k_5^2 + x_2^2 \cdot k_6^2 + x_4^2 \cdot k_8^2 + x_5^2 \cdot k_9^2$
Channel:1
Channel:2
Output feature map

$$x_1^1 \cdot k_5^1 + x_2^1 \cdot k_6^1 + x_4^1 \cdot k_8^1 + x_5^1 \cdot k_9^1 + x_1^2 \cdot k_5^2 + x_2^2 \cdot k_6^2 + x_4^2 \cdot k_8^2 + x_5^2 \cdot k_9^2$$

$$\frac{(x_1^1 \cdot k_5^1 + x_2^1 \cdot k_6^1 + x_4^1 \cdot k_8^1 + x_5^1 \cdot k_9^1 + x_1^2 \cdot k_5^2 + x_2^2 \cdot k_6^2 + x_4^2 \cdot k_8^2 + x_5^2 \cdot k_9^2) - \mu_1}{\sqrt{\sigma_1^2 + \epsilon}} \cdot \gamma_1 + \beta_1$$



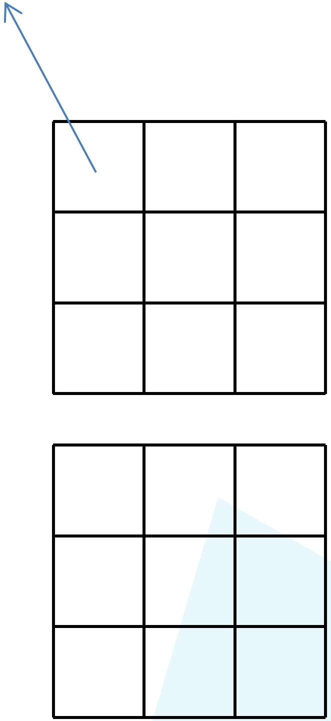
Channel:1

Channel:2

Input feature map

μ_1	σ_1^2	γ_1	β_1
μ_2	σ_2^2	γ_2	β_2

BN params



Channel:1

Channel:2

Output feature map

RepVGG

结构重参数化-融合Conv2d和BN

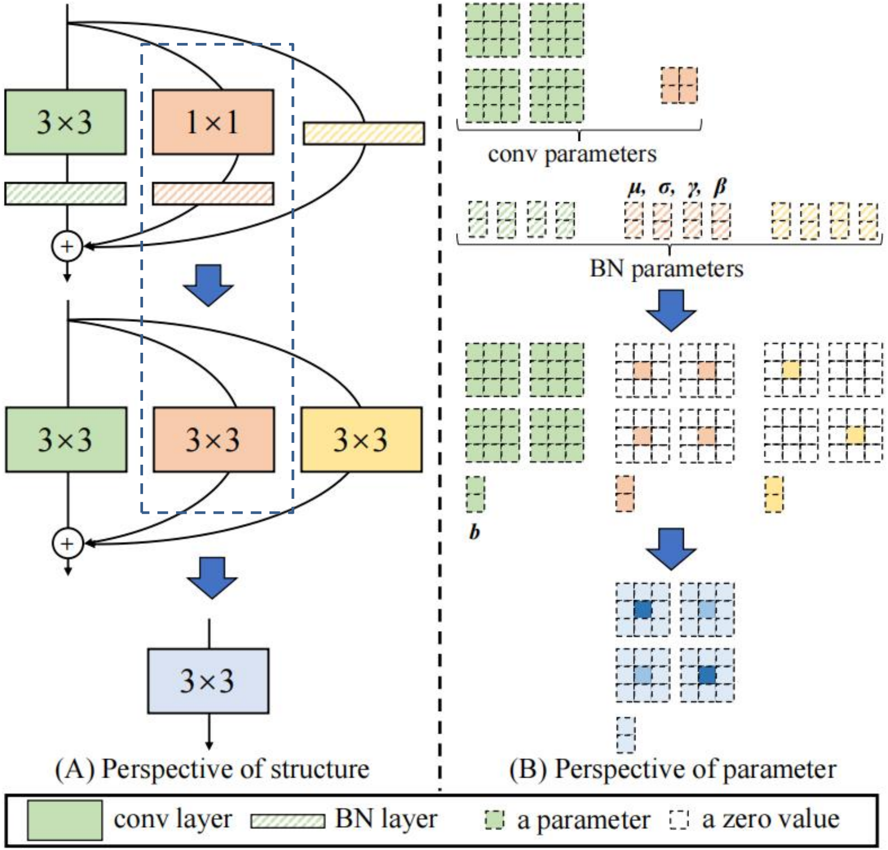
$$\frac{(x_1^1 \cdot k_5^1 + x_2^1 \cdot k_6^1 + x_4^1 \cdot k_8^1 + x_5^1 \cdot k_9^1 + \ x_1^2 \cdot k_5^2 + x_2^2 \cdot k_6^2 + x_4^2 \cdot k_8^2 + x_5^2 \cdot k_9^2) - \mu_1}{\sqrt{\sigma_1^2 + \epsilon}} \cdot \gamma_1 + \beta_1$$

$$(x_1^1 \cdot k_5^1 + x_2^1 \cdot k_6^1 + x_4^1 \cdot k_8^1 + x_5^1 \cdot k_9^1 + \ x_1^2 \cdot k_5^2 + x_2^2 \cdot k_6^2 + x_4^2 \cdot k_8^2 + x_5^2 \cdot k_9^2) \cdot \frac{\gamma_1}{\sqrt{\sigma_1^2 + \epsilon}} + (\beta_1 - \frac{\mu_1 \cdot \gamma_1}{\sqrt{\sigma_1^2 + \epsilon}})$$

常数项

RepVGG

结构重参数化-将1x1卷积转换成3x3卷积



RepVGG

结构重参数化-将1x1卷积转换成3x3卷积

Channel:1

$$\begin{array}{|c|c|c|}\hline x_1^1 & x_2^1 & x_3^1 \\ \hline x_4^1 & x_5^1 & x_6^1 \\ \hline x_7^1 & x_8^1 & x_9^1 \\ \hline \end{array} \qquad \boxed{k_1^1} \qquad \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & k_1^1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

Channel:2

$$\begin{array}{|c|c|c|}\hline x_1^2 & x_2^2 & x_3^2 \\ \hline x_4^2 & x_5^2 & x_6^2 \\ \hline x_7^2 & x_8^2 & x_9^2 \\ \hline \end{array} \qquad \boxed{k_1^2} \qquad \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & k_1^2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

Input feature map

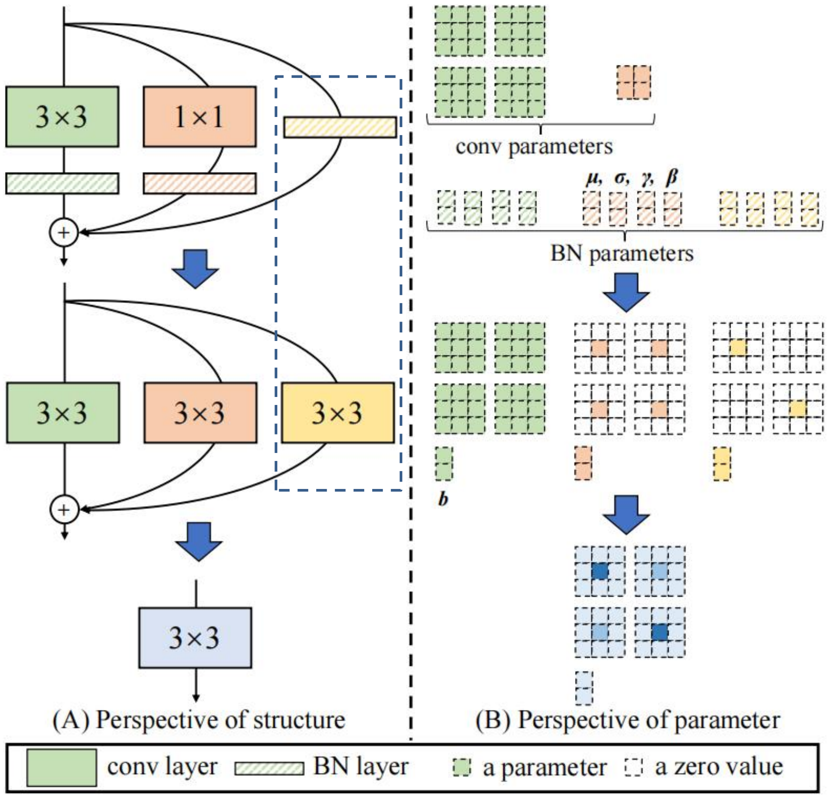
origin kernel weight: 1
stride=1, pad=0

converted kernel weight: 1
stride=1, pad=1

Bilibili: 霹雳吧啦Wz

RepVGG

结构重参数化-将BN转换成3x3卷积



RepVGG

结构重参数化-将BN转换成3x3卷积

Input feature map

Channel:1

$$\begin{bmatrix} x_1^1 & x_2^1 & x_3^1 \\ x_4^1 & x_5^1 & x_6^1 \\ x_7^1 & x_8^1 & x_9^1 \end{bmatrix}$$

Channel:2

$$\begin{bmatrix} x_1^2 & x_2^2 & x_3^2 \\ x_4^2 & x_5^2 & x_6^2 \\ x_7^2 & x_8^2 & x_9^2 \end{bmatrix}$$

kernel weight: 1

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

kernel weight: 2

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

stride=1, pad=1

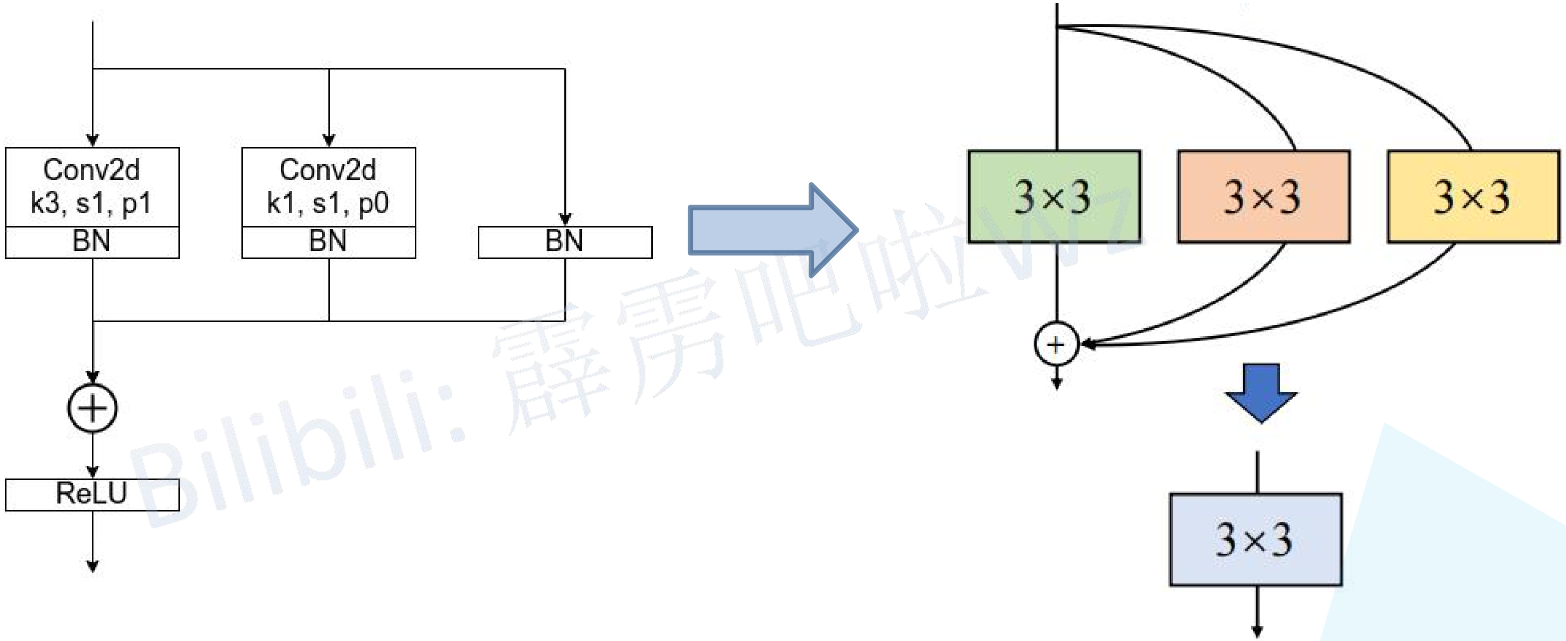
Output feature map

$$\begin{bmatrix} x_1^1 & x_2^1 & x_3^1 \\ x_4^1 & x_5^1 & x_6^1 \\ x_7^1 & x_8^1 & x_9^1 \end{bmatrix}$$

$$\begin{bmatrix} x_1^2 & x_2^2 & x_3^2 \\ x_4^2 & x_5^2 & x_6^2 \\ x_7^2 & x_8^2 & x_9^2 \end{bmatrix}$$

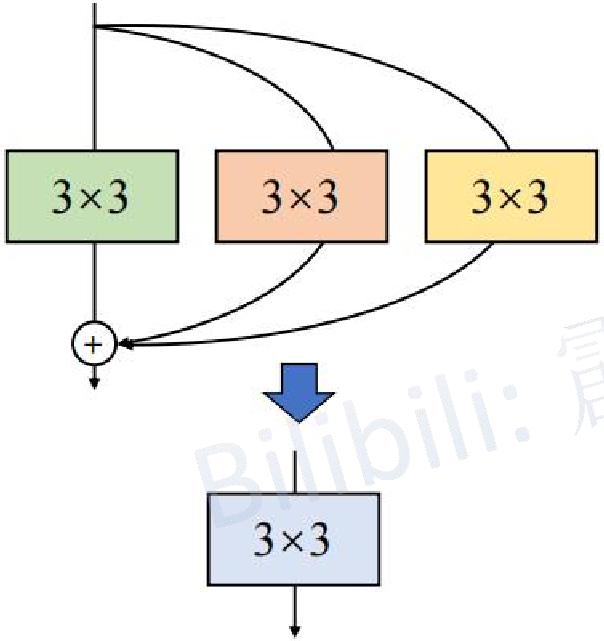
RepVGG

结构重参数化-多分支融合



RepVGG

结构重参数化-多分支融合

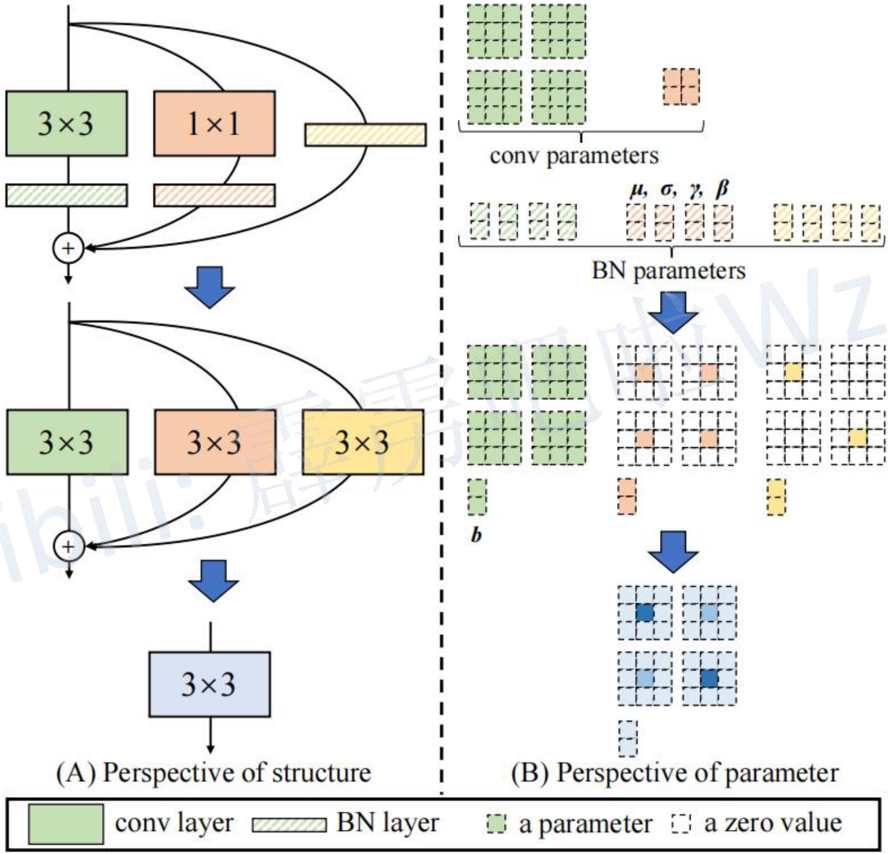


$$O = (I \otimes K_1 + B_1) + (I \otimes K_2 + B_2) + (I \otimes K_3 + B_3)$$
$$= I \otimes (K_1 + K_2 + K_3) + (B_1 + B_2 + B_3)$$

$\otimes$：表示卷积运算

RepVGG

结构重参数化



- 减少参数
- 加速推理

RepVGG

模型配置

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

Table 2: Architectural specification of RepVGG. Here $2 \times 64a$ means stage2 has 2 layers each with $64a$ channels.

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	$1 \times 512b$	$1 \times 512b$

Table 3: RepVGG models defined by multipliers a and b.

Name	Layers of each stage	a	b
RepVGG-A0	1, 2, 4, 14, 1	0.75	2.5
RepVGG-A1	1, 2, 4, 14, 1	1	2.5
RepVGG-A2	1, 2, 4, 14, 1	1.5	2.75
RepVGG-B0	1, 4, 6, 16, 1	1	2.5
RepVGG-B1	1, 4, 6, 16, 1	2	4
RepVGG-B2	1, 4, 6, 16, 1	2.5	5
RepVGG-B3	1, 4, 6, 16, 1	3	5

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26]

沟通方式

1.github

https://github.com/WZMIAOMIAO/deep-learning-for-image-processing

2.bilibili

https://space.bilibili.com/18161609/channel/index

3.CSDN

https://blog.csdn.net/qq_37541097/article/details/103482003