RepVGG:2021

RepVGG:Making VGG-style ConvNets Great Again

code:https://github.com/megvii-model/RepVGG

论文出发点或背景

以ResNet和Inception Net的多分支结构虽然能达到很高的精度，但是缺点也十分明显，比如模型难以实现或者是客制化，降低了模型推理的速度，同时也降低了内存的利用率，增加了内存的访问成本。

ResNet通过多分支，使得模型成为了许多较浅模型的集成，从而避免了梯度消失问题

先前的工作主要是为了让非常深的模型能够以一个不错的精度收敛，但是忽略了模型对推理的影响

3×3卷积核在GPU等硬件设备上得到了很好的支持

论文创新思路

多分支的训练方式对于训练是有益的，但是不利于推理

通过结构重参数化来解决问题

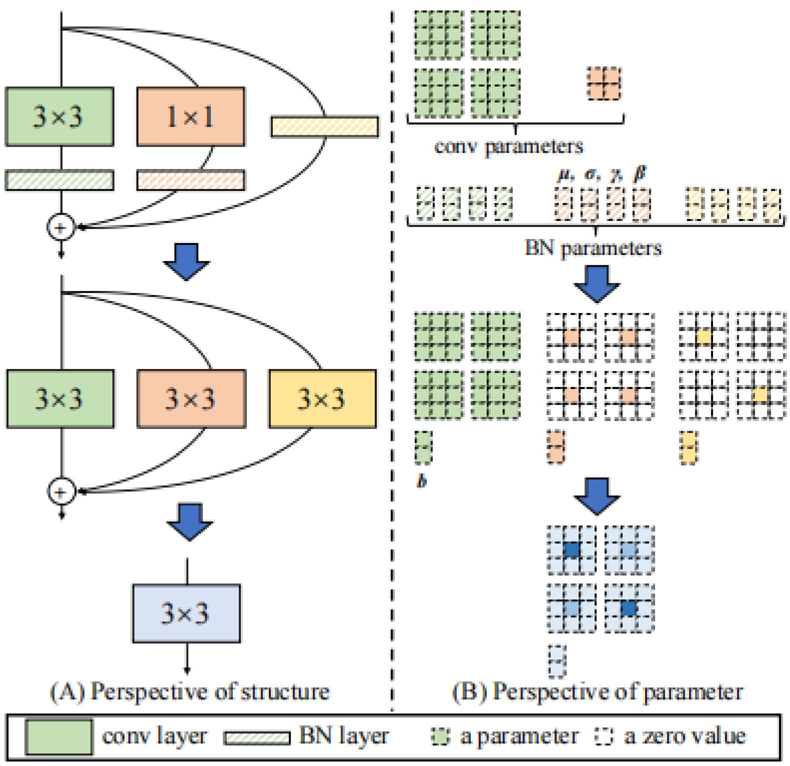

Figure 4: Structural re-parameterization of a RepVGG block. For the ease of visualization, we assume $C_2 = C_1 = 2$, thus the 3×3 layer has four 3×3 matrices and the kernel of 1×1 layer is a 2×2 matrix.

论文方法大概介绍

训练的时候采用多分支结构进行训练，但是推理的时候只使用单路径进行推理，由多路径变为单路经使用结构重参数化

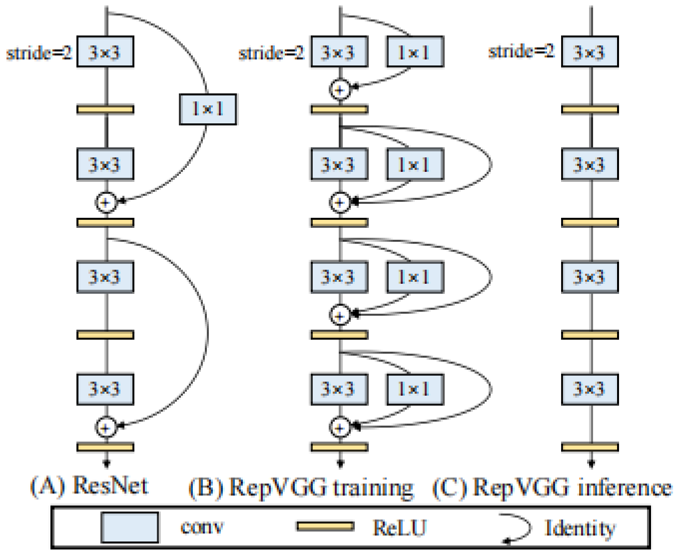

Figure 2: Sketch of RepVGG architecture. RepVGG has 5 stages and conducts down-sampling via stride-2 convolution at the beginning of a stage. Here we only show the first 4 layers of a specific stage. As inspired by ResNet [12], we also use identity and 1×1 branches, but only for training.

RepVGG模型的优点：
1.类VGG的架构，每一层的输出都作为下一层的唯一输入

2.模型全部只使用了3×3的卷积核和ReLU的模型架构

3.没有自动搜索、手动细化、复合缩放或者其他类型的复杂设计

网络重参数化：

训练时的结构对应一组参数，推理时我们想要的结构对应另一组参数；只要能把前者的参数等价转换为后者，就可以将前者的结构等价转换为后者。

受到ResNet方法启迪，显式构造了一个模型将信息流建模为$y = g(x) + f(x)$,其中$g(x)$是通过一个1×1卷积实现的，目的是为了维度匹配。为了让更多含有更多浅层信息，我们最终构建了一个$y = x + g(x) + f(x)$的信息流

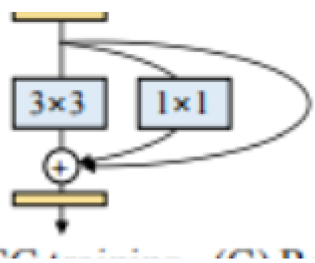

Table 2: Architectural specification of RepVGG. Here $2 \times 64a$ means stage2 has 2 layers each with $64a$ channels.

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

实际效果

与最先进的技术相比较，具有良好的速度-精度权衡

在图像分类和语义分割方面效率高，准确率高，方便

证明了普通的模型也可以很好的收敛，可以简单地用最常用的组件和简单的代数来实现

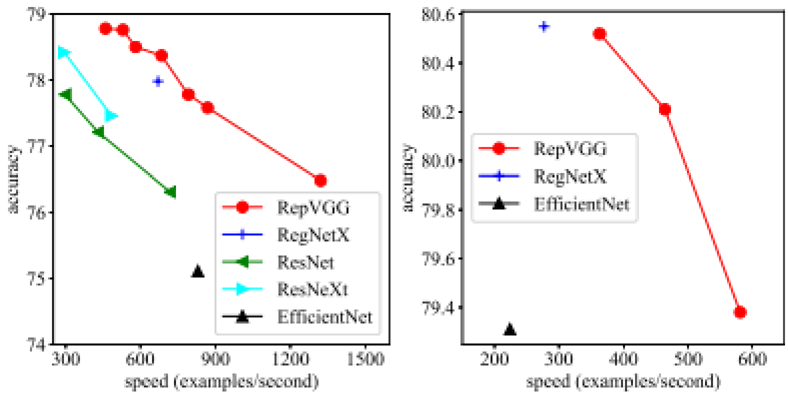

Figure 1: Top-1 accuracy on ImageNet *vs*. actual speed. Left: lightweight and middleweight RepVGG and baselines trained in 120 epochs. Right: heavyweight models trained in 200 epochs. The speed is tested on the same 1080Ti with a batch size of 128, full precision (fp32), single crop, and measured in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

Table 4: Results trained on ImageNet with simple data augmentation in 120 epochs. The speed is tested on 1080Ti with a batch size of 128, full precision (fp32), and measured in examples/second. We count the theoretical FLOPs and Wino MULs as described in Sect. 2.4. The baselines are our implementations with the same training settings.

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 5: Results on ImageNet trained in 200 epochs with Autoaugment [5], label smoothing and mixup.

Model	Acc	Speed	Params	FLOPs	MULs
RepVGG-B2g4	79.38	581	55.77	11.3	6.0
RepVGG-B3g4	80.21	464	75.62	16.1	8.4
RepVGG-B3	80.52	363	110.96	26.2	12.9
RegNetX-12GF	80.55	277	46.05	12.1	10.9
EfficientNet-B3	79.31	224	12.19	1.8	-

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

Table 7: Comparison with variants and baselines on RepVGG-B0 trained in 120 epochs.

Variant and baseline	Accuracy
Identity w/o BN	74.18
Post-addition BN	73.52
Full-featured reparam	**75.14**
+ReLU in branch	75.69
DiracNet [39]	73.97
Trivial Re-param	73.51
ACB [10]	73.58
Residual Reorg	74.56

Table 8: Semantic segmentation on Cityscapes [4] tested on the *validation* subset. The speed (examples/second) is tested with a batch size of 16, full precision (fp32), and input resolution of 713×713 on the same 1080Ti GPU.

Backbone	Mean IoU	Mean pixel acc	Speed
RepVGG-B1g2-fast	78.88	96.19	10.9
ResNet-50	77.17	95.99	10.4
RepVGG-B1g2	78.70	96.27	8.0
RepVGG-B2-fast	79.52	96.36	6.9
ResNet-101	78.51	96.30	6.7
RepVGG-B2	80.57	96.50	4.5

优点：模型简单、快速、实用

缺点：在嵌入式设备和移动设备上的表现不是很好

个人理解

1.多分支模型对训练友好，对推理不友好，直筒型模型对推理友好，但是对训练不友好，所以我可以通过多分支模型训练网络，利用卷积的可加性，将每个分支的卷积加起来，得到一个单支路的网络，这样我的单支路模型是和多支路模型在推理的时候是等价的，但是推理性能上要优于分支模型，感觉就是来通过等价来取得分支模型和直筒模型的优点，各取所长。

2.3×3卷积核在很多计算设备上都有一定的加速，所以网络选取了3×3卷积进行主要架构，模型等效的时候将1×1卷积和identity支路等效为3×3卷积，实现了单支路向多支路的转化。