清华&旷视：RepVGG，更佳的速度-精度 trade-off！

文章题目：RepVGG：Make VGG-style ConvNets outstanding again
代码地址：https://github.com/DingXiaoH/RepVGG
摘要：

本文提出一种简单而强有力的 CNN 架构 RepVGG，在推理阶段，它具有与 VGG 类似的架构，而在训练阶段，它则具有多分支架构体系，这种训练-推理解耦的架构设计被称为“重参数化”。

研究现状：
尽管许多复杂的 ConvNet 都比简单的 ConvNet 提供更高的准确性，但缺点很明显。1）复杂的多分支设计（例如 ResNet 中的残差加法和 Inception 中的分支级联）使模型难以实现和定制，减慢了推理速度并降低了内存利用率。2）一些组件增加了内存访问成本，并且缺乏对各种设备的支持。此外，还有影响的因素，浮点运算（FLOP）的数量不能准确反映实际速度。因此，VGG 和 ResNets 的原始版本仍然在学术界和工业界广泛用于现实世界的应用程序。

研究贡献：
基于上述研究现状，本文的研究人员提出了一种简单有强有的 CNN 架构 RepVGG，相比其他架构，具有更佳的精度-速度均衡；对 plain 架构采用重参数化技术；并在图像分类、语义分割等任务上验证了 RepVGG 的有效性。

研究细节：
1. 选择 ConvNet 的原因：

快速：相比 VGG，现有的多分支架构理论上具有更低的计算速度，但推理速度并未更快。计算速度与推理速度的矛盾主要源自两个关键因素：(1)内存访问消耗，比如多分支结构的运算很小，但内存访问消耗很高；(2)并行度，并行度高的模型要比并行度低的模型推理速度更快。

节省内存：多分支结构是一种内存低效的架构，这是因为每个分支的结构都需要在运算之前保存，这会导致更大的峰值内存占用；而 plain 模型则具有更好的内存高效特征。

灵活：多分支结构会限制 CNN 的灵活性，与此同时，多分支结构对于模型剪枝不够友好。

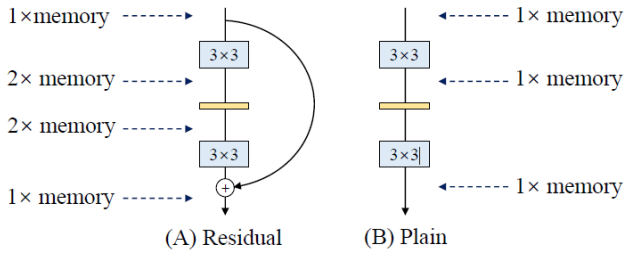

Figure 3: Peak memory occupation in residual and plain model. If the residual block maintains the size of feature map, the peak value of memory occupied by feature maps will be 2× as the input. The memory occupied by the model parameters is small compared to the features hence ignored.

2. 训练中的多分支架构体系

Palin 模型存在性能差的缺点。本文设计的 RepVGG，其 ResNet 的 ResBlock 构建了一个短连接模型信息流$y = x + f(x)$，当x，$f(x)$的维度不匹配时，则转变为$y = g(x) + f(x)$。尽管多分支结构对于推理不友好，但适合于训练，研究人员将 RepVGG 设计为训练时多分支，推理时单分支结构。研究人员设计了如下形式模块：

$$y = x + g(x) + f(x)$$

其中，$g(x), f(x)$分别对应1×1，3×3卷积。在训练阶段，通过简单的堆叠上述模块构建 CNN 架构；而在推理阶段，上述模块可以轻易转换为$y = h(x)$形式，且$h(x)$的参数可以通过线性组合方式从已训练好的模型中转换得到。

3. Plain 架构的重新参数化

研究中将已训练模块转换成单一的3×3卷积用于推理。下图给出了参数转换示意图。

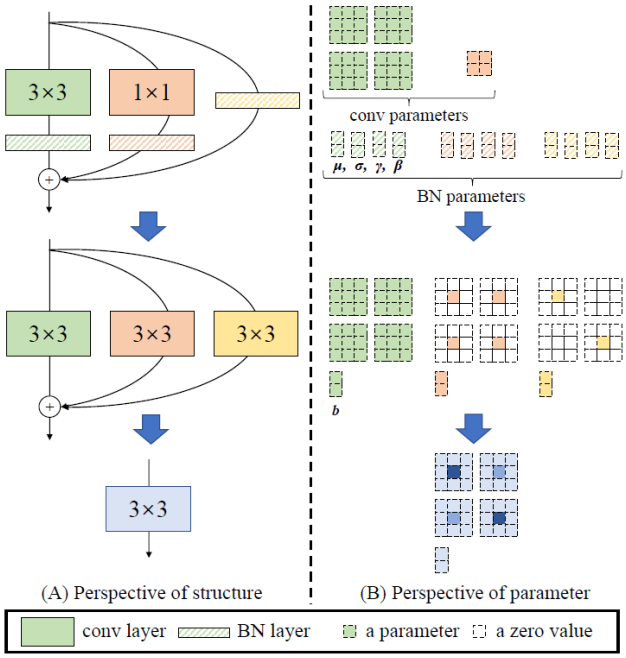

Figure 4: Structural re-parameterization of a RepVGG block. For the ease of visualization, we assume $C_2 = C_1 = 2$, thus the 3×3 layer has four 3×3 matrices and the kernel of 1×1 layer is a 2×2 matrix.

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

模块中仅具有一个3×3卷积核，两个 1×1卷积核以及三个 bias 参数，三个 bias 参数相加即可合并为一个 bias，卷积核是通过将1×1卷积核参数与3×3卷积核的中心点相加获取的。

4. RepVGG 网络设计

RepVGG 是一种类 VGG 的架构，文中对于每个阶段的层数按照如下规则进行相应的设计。研究人员遵循三个简单的准则来决定每个阶段的层数。1）第一阶段以高分辨率运行，很耗时，因此仅使用一层来降低延迟。2）最后一级应具有更多通道，因此仅使用一层来保存参数。3）紧随 ResNet 及其最新版本之后，研究人员将最多的层放到倒数第二级（在 ImageNet

上具有的14×14输出分辨率），设置五个阶段分别具有 1、2、4、14、1 层，以构造一个名为 RepVGG-A 的实例。此外还构建了更深的 RepVGG-B，在 stage2、3 和 4 中又增加了 2 层。使用 RepVGG-A 与其他轻型和中等重量型号竞争，包括 ResNet-18 / 34/50，而 RepVGG-B 与高性能机型竞争

基于上述规则，RepVGG-A 中的层数对应为 1，2，4，14，1；RepVGG-B 中的层数对应为 1，4，6，16，1，下图为两者的对比。

Table 2: Architectural specification of RepVGG. *E.g.*, $2 \times 64a$ 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	$1 \times 512b$	$1 \times 512b$

5. 实验：

实验中，研究人员将 RepVGG 的性能与 ImageNet 上的基线进行比较，通过一系列消融研究和比较证明结构重新参数化的重要性，并验证 RepVGG 在语义分割上的泛化性能

本文主要是在 ImageNet 图像分类任务上进行了实验，实验结果如下图所示，主要反映了 RepVGG 和不同计算量的 ResNe 和变体在精度、速度、参数量上的对比，从中不难发现，RepVGG 具有更好的精度-速度均衡。在 ImageNet 数据集上，RepVGG 取得了超过 80%的 top-1 精度，这是 plain 模型首次达到如此高的精度。在 NVIDIA 1080TiGPU 上，RepVGG 比 ResNet50 快 83%，比 ResNet101 快 101%，同时具有更高的精度。

Table 4: RepVGG models and baselines trained in 120 epochs with simple data augmentation on ImageNet. 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

与此同时，RepVGG 在参数量和推理速度上也具有较好的性能。

Table 5: RepVGG models and baselines trained in 200 epochs with Autoaugment [4], 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 8: Semantic segmentation on Cityscapes [3] 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

不足：RepVGG 模型是快速，简单和实用的 ConvNet 架构，在 GPU 和专用硬件上以最快的速度运行，而无需考虑参数等。尽管 RepVGG 模型的参数效率比 ResNets 高，但在低功耗设备应用上，它们可能不如 MobileNets 和 ShuffleNets 等移动系统模型。

总结：本文提出了 RepVGG，一个简单的体系结构，适用于 GPU 和专用推理芯片。研究人员通过结构重新参数化方法，使其在 ImageNet 上可以达到 80%的 top-1 精度，并且与最新的复杂模型相比，它显示出良好的速度-精度性能。

发布链接：https://mp.weixin.qq.com/s/6v7-sW7mD1qQdBxJ6eJYnw