通用 Vision Backbone 超详细解读 (七)：重参数化你的优化器：VGG 型架构 + 特定的优化器 = 快速模型训练 + 强悍性能

科技猛兽
清华大学 自动化系硕士在读

专栏目录

本文目录

> 论文名称：Re-parameterizing Your Optimizers rather than Architectures

论文地址：

https://arxiv.org/pdf/2205.15242.pdf
arxiv.org/pdf/2205.15242.pdf

重参数化技术延伸阅读：

科技猛兽：解读模型压缩6：结构重参数化技术：进可暴力提性能，退可无损做压缩
74 赞同 · 7 评论 文章

- **1.1 RepOptimizer 原理分析**

1.1.1 你有多久没换过优化器了？

在现代计算机视觉任务中，大家关注的点一般专注于如何设计更好，表征能力更强的神经网络架构，以提升视觉任务的性能。

当大家的目光从卷积神经网络 (CNN) 的深度 (Depth) 、宽度 (Width) 、组数 (Group)、输入分辨率，再转到 Kernel Size 的时候，却从没有想过调节一下模型的优化器，总是被默认设为 SGD 或 AdamW?

当大家的目光从视觉 Transformer 模型的架构设计，宽度剪枝，再转到自动搜索的时候，是否想过要换一换 ViT 等模型默认的 AdamW 优化器？是否希望有一种简单、训练快速的优化器，它可以快速、低显存消耗、巧妙地把模型训好，同时性能不输 SGD 等主流 Optimizer?

RepVGG 一作联合旷视，推出 RepOptimizer (RepOpt)。该工作表明，优化器的泛化性导致它是一个非常重要但总是被人忽略的设计维度，在现代模型设计的加持下，人们往往会根据经验去习惯性地使用一些 "已经被验证过有效" 的优化器。本文作者将重参数化技术重新在优化器中使用，提出 RepOptimizer。我们知道，优化过程最重要的变量就是梯度，RepOptimizer 根据一组模型特定的超参数修改梯度来添加先验知识，这个过程也称为梯度重参数化 (Gradient Re-parameterization)。作者仅仅使用了一个极简的 VGG-Style 模型，通过 RepOptimizer 训练出的 RepOpt-VGG 推理速度快，训练效率高，同时性能依旧强悍，是一个很好的基础模型。

1.1.2 设计动机和背景

要获得一个性能优良的视觉模型，目前主要是两种思路：**神经网络的架构设计，优化方法的设计。**

神经网络的架构设计是将人类的先验知识结合到模型中，比如 ResNet[1] 将残差先验引入模型中，并优于没有此类先验知识的简单模型，例如 VGGNet[2]；EfficientNet[3] 通过网络架构搜索和复合缩放获得了一组结构超参数，作为构建模型的先验知识；RegNet[4] 通过改变架构的超参数获得一系列的模型，它们帮助提升架构的性能。但是，**① 尽管更高级的模型架构不断地将我们的最新理解纳入模型，可是我们实际使用的优化器却还是 SGD 和 AdamW，它们都是模型无关 (model-agnostic) 的。**也就是说，我们早已习惯了 **"特定结构+通用优化器"** 的优化范式。

优化方法的设计就是指设计高效的优化器，例如广泛用于各种 CNN，Transformer 和 MLP 模型的 SGD[5]，Adam[6]，AdamW[7] 等。还有一些比较高阶的优化器[8][9][10]，但是不舒服的地方是还要计算复杂的 Hessian 矩阵。但是，**② 尽管更高级的优化器以不同的方式改进了训练过程，但是它们没有特定于被优化的模型的先验知识。**

通过阅读以上两段话中的黑体字 ① ② 我们做出总结：**"特定结构+通用优化器"** 的确是一种可行的优化范式。但是，若能在优化的过程中，**将先验知识更多地放在优化器中，而不是模型中，是否是一种新的优化模型的思路？这也就意味着我们会不再去把我们的先验知识都用在模型架构上面，而**

本文的贡献主要有3点：

1. 通用的简单 VGG 架构 + 模型相关的特定优化器 (generic structure + specific optimizer)：通用的简单 VGG 架构是指模型架构设计时，使用的人类先验知识越少越好。模型相关的特定优化器意味着在优化过程中，优化器的行为是与模型的架构密切相关的，训练的动态以特定于模型的方式来改变，而不是诸如 SGD，Adam 那样一成不变。

2. 一种针对 SGD 型优化器的方法：SGD 的核心是用梯度更新参数，作者建议在更新可训练参数之前，通过**一组特定于模型的超参数 "修改" 梯度**，使得优化器成为 model-specific 的。作者将这种方法称之为梯度重参数化 (Gradient Re-parameterization, GR)。为了获得这样的超参数，作者提出了一种叫做超参数搜索的方法。

3. 一个相对不错的基础模型：为了证明通用架构 + 模型相关的特定优化器的有效性，作者希望模型尽可能简单，所以和 RepVGG[11] 一样选择了极简 VGG 风格的架构，只使用 3×3 卷积这一种操作，经过梯度重参数化的 RepOptimizer 训练之后，得到的 RepOpt-VGG 模型的精度可以与 RegNet 和 EfficientNet 这种有更丰富的结构先验模型的精度相媲美。

1.1.4 本文做了哪些具体的工作?

> **a) 理论方面：**

这个工作与 RepVGG 相当于是两个不同的改进方向：

RepVGG 是在训练的时候，将模型设计为一个多分支的架构，使用常规的 SGD 等优化器进行训练。训练完以后，把模型的参数等价转换为 VGG 型直筒型网络，用于后续的推理过程。

a1) RepVGG 与 ResNet 等模型类似，也在设计训练时的模型架构时，融入人类先验知识，并利用通用优化器，但是 **RepOpt 将人类先验知识添加进优化器中**。

a2) RepVGG 虽然在推理时拥有极简的 VGG 类型架构，但是训练时却是一个较为复杂的模型，并且需要更多的时间和内存来训练。相比之下，**RepOpt-VGG 在训练时也是一个极简的 VGG 类型架构**。

a3) **RepOpt 扩展和深化了结构重参数化技术**，本文表明即使训练时候的模型也是一个极简的 VGG 类型架构，我们依然能够通过定制的优化器实现等效的训练动态。这就意味着：当输入和其他训练的设置相同时，RepVGG 和 RepOpt 在训练过程中的任何迭代中产生理论上相同的输出。

> **b) 应用方面：**

RepOpt-VGG 也是一个很好的基础模型，它具有高效推理和高效训练的特点：

b1) **RepOpt-VGG 继承了 RepVGG 的极简架构**，具有低内存消耗 (因为它没有分支)、高并行度，并极大地受益于高度优化的 3×3 Conv (例如 Winograd 算法[12]) 。而且，大量的 3×3 Conv 单元集成到一个定制的芯片上，也有利于硬件的专门化。

b2) **RepVGG 需要更多的时间和内存来训练**，这在计算资源有限或我们希望快速交付或快速迭代模型的应用场景下并不占优势。比如我们可能需要每隔几天用新收集的数据重新训练模型。而 RepOpt-VGG 的训练速度约为 RepVGG 的1.7倍。而且 RepOpt-VGG 由于在训练过程中间只用到了 3×3 Conv，可以去专门定制高通量训练芯片训练这样的简单模型。

b3) **解决 RepVGG 的后量化问题**：RepVGG 的推理模型很难使用后量化方法 (Post-Training Quantization, PTQ)。比如，使用简单的 INT8 PTQ，ImageNet 上的 RepVGG 模型的准确性降低到 54.55%。但是 RepOpt-VGG 却对量化是友好的，**量化 RepVGG 的问题源于 "训练模型到推理模型时的等价变换"**。优化器的重参数化可以很好地解决这个问题，因为根本没有进行 "训练模型到推理模型时的等价变换"。

在以往的架构设计中，我们一般将人类先验知识融入模型架构的设计中。比如 ResNet 添加残差连接，把输出和输入加起来。RepVGG 使用多个分支，并把 BN 层之后的输出加起来得到该层的输出。以上种种都可以视为是架构的先验知识。

而 RepOpt 的第一步就是将架构的先验知识转移到你的优化器中。也就是说，架构方面使用极简的 VGG 类型，而是把关注的重点放在优化器上。下面介绍本文提出的一个概念，即：

> **一个包含了多个分支的 Block，如果其每个分支都有一个可选择的 scale 常数，且还有一个可学习的线性变换算子 (卷积，或者 FC)，而没有 BN，Dropout等非线性算子，则将这样的 Block 称之为 Constant-Scale Linear Addition (CSLA)。**

CSLA 当然可以在部署的时候转化成一个 3×3 卷积，本文假设在训练的时候有一个**等效的单算子**与 CSLA 块是等效的，且有着相同的训练动态过程，即：如果给它们提供相同的训练数据，它们在任意数量的训练迭代后总是会产生相同的输出。那么究竟怎么保持训练过程的等效？本文假设通过将训练过程的**梯度**乘以一个**与 scale 常数相关的 Gradient Masks**。用 Grad Masks 修改梯度可以看作是 Gradient Re-parameterization 的具体实现，简而言之，则有：

> **CSLA 块+ 常规优化器 = 等效的单算子 + 梯度重参数化优化器**

假设 CSLA 块有两个相同形状的并行卷积核 $\mathbf{W}^{(A)}, \mathbf{W}^{(B)}$，每个核都由一个常数标量缩放 α_A, α_B，输入和输出分别是 $\mathbf{X}$ 和 $\mathbf{Y}$，则 CSLA 的输出是：$\mathbf{Y}_{CSLA} = \alpha_A(\mathbf{X} * \mathbf{W}_A) + \alpha_B(\mathbf{X} * \mathbf{W}^{(B)})$，其中 $*$ 代表卷积操作。对于 GR 这部分，为单分支操作 $\mathbf{Y}_{GR} = \mathbf{X} * \mathbf{W}'$。假设损失函数为 L，梯度为 $\frac{\partial L}{\partial \mathbf{W}}$，GR 分支对梯度进行变换为 $F(\frac{\partial L}{\partial \mathbf{W}'})$，则有下面的命题成立。

命题：当满足以下两个条件时，存在仅由 α_A 和 α_B 确定的变换 F，使得更新 $\mathbf{W}'$ 时可以确保：

$$\mathbf{Y}^{(i)}_{CSLA} = \mathbf{Y}^{(i)}_{GR} \quad \forall i \geq 0 . \tag{1}$$

条件1： $\mathbf{W}'$ 应该被初始化为 $\mathbf{W}'^{(0)} \leftarrow \alpha_A \mathbf{W}^{(A)(0)} + \alpha_B \mathbf{W}^{(B)(0)}$。换句话说就是 GR 的初始化应该和 CSLA 的初始化保持一致。

条件2：CSLA 块使用常规 SGD 进行更新，GR 对应的梯度应乘以 $(\alpha_A^2 + \alpha_B^2)$，可以写成：

$$\mathbf{W}'^{(i+1)} \leftarrow \mathbf{W}'^{(i)} - \lambda(\alpha_A^2 + \alpha_B^2)\frac{\partial L}{\partial \mathbf{W}'^{(i)}} . \tag{2}$$

证明：

根据卷积的叠加性和齐次性，需要确保在任何 iteration i，都有下式成立：

$$\alpha_A \mathbf{W}^{(A)(i)} + \alpha_B \mathbf{W}^{(B)(i)} = \mathbf{W}'^{(i)} \quad \forall i \geq 0 . \tag{3}$$

在第0个 iteration 时，初始权重应满足：

$$\mathbf{W}'^{(0)} = \alpha_A \mathbf{W}^{(A)(0)} + \alpha_B \mathbf{W}^{(B)(0)} , \tag{4}$$

因此有：

$$\mathbf{Y}^{(0)}_{CSLA} = \mathbf{Y}^{(0)}_{GR} . \tag{5}$$

然后使用数学归纳法来证明：设 λ 为学习率，更新规则为：

$$\mathbf{W}^{(i+1)} = \mathbf{W}^{(i)} - \lambda \frac{\partial L}{\partial \mathbf{W}^{(i)}} \quad \forall i \geq 0 . \tag{6}$$

当更新了 CSLA 块之后，有：

$$-\lambda(\alpha_A\frac{\partial L}{\partial \mathbf{W}^{(A)(i)}}+\alpha_B\frac{\partial L}{\partial \mathbf{W}^{(B)(i)}}))\,.$$

使用 $F(\frac{\partial L}{\partial \mathbf{W}})$ 来更新 $\mathbf{W}'$，有：

$$\mathbf{W}'^{(i+1)}=\mathbf{W}'^{(i)}-\lambda F(\frac{\partial L}{\partial \mathbf{W}'^{(i)}})\,. \tag{8}$$

根据式 2,6,7 有：

$$F(\frac{\partial L}{\partial \mathbf{W}'^{(i)}})=\alpha_A\frac{\partial L}{\partial \mathbf{W}^{(A)(i)}}+\alpha_B\frac{\partial L}{\partial \mathbf{W}^{(B)(i)}}\,. \tag{9}$$

3式若成立，则下面两个偏导数就得成立：

$$\frac{\partial \mathbf{W}'^{(i)}}{\partial \mathbf{W}^{(A)(i)}}=\alpha_A\,,\quad \frac{\partial \mathbf{W}'^{(i)}}{\partial \mathbf{W}^{(B)(i)}}=\alpha_B\,. \tag{10}$$

则有：

$$\begin{aligned}F(\frac{\partial L}{\partial \mathbf{W}'^{(i)}})&=\alpha_A\frac{\partial L}{\partial \mathbf{W}'^{(i)}}\frac{\partial \mathbf{W}'^{(i)}}{\partial \mathbf{W}^{(A)(i)}}+\alpha_B\frac{\partial L}{\partial \mathbf{W}'^{(i)}}\frac{\partial \mathbf{W}'^{(i)}}{\partial \mathbf{W}^{(B)(i)}}\\&=(\alpha_A^2+\alpha_B^2)\frac{\partial L}{\partial \mathbf{W}'^{(i)}}\,,\end{aligned} \tag{11}$$

通过上式11，我们有：

$$\begin{aligned}\alpha_A\mathbf{W}^{(A)(i+1)}+\alpha_B\mathbf{W}^{(B)(i+1)}&=\alpha_A\mathbf{W}^{(A)(i)}+\alpha_B\mathbf{W}^{(B)(i)}\\&-\lambda(\alpha_A\frac{\partial L}{\partial \mathbf{W}^{(A)(i)}}+\alpha_B\frac{\partial L}{\partial \mathbf{W}^{(B)(i)}}))\\&=\mathbf{W}'^{(i)}-\lambda F(\frac{\partial L}{\partial \mathbf{W}'^{(i)}})=\mathbf{W}'^{(i+1)}.\end{aligned} \tag{12}$$

即：

$\alpha_A\mathbf{W}^{(A)(i+1)}+\alpha_B\mathbf{W}^{(B)(i+1)}=\mathbf{W}'^{(i+1)}$：。通过初始条件和数学归纳，证明了任何 $i\geq 0$ 的等价性。

得证。

此命题可以得出结论：用于更新 GR 对应算子的梯度应该简单地按一个常数因子缩放，在这种情况下为 $(\alpha_A^2+\alpha_B^2)$。下面两种训练的流程能够得到完全相同的输出：

> **CSLA**：构造两个卷积层，分别初始化为 $\mathbf{W}^{(A)(0)}$ 和 $\mathbf{W}^{(B)(0)}$ 并训练 CSLA 块 $\mathbf{Y}_{CS}=\alpha_A(\mathbf{X}*\mathbf{W}_A)+\alpha_B(\mathbf{X}*\mathbf{W}_B)$。
> **GR**：构造一个单卷积算子 $\mathbf{W}'$，并初始化为 $\mathbf{W}'^{(0)}=\mathbf{W}^{(A)(0)}+\mathbf{W}^{(B)(0)}$，按照11式来计算梯度。

整个过程可以用下图1表示。

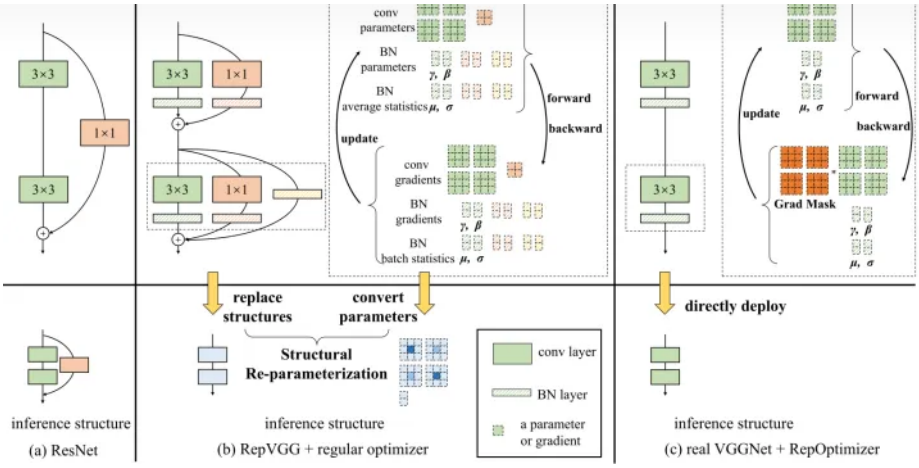


图1：常规的模型。RepVGG + 常规的优化器。VGG 模型 + RepOptimizer。

以上例子中的情况是最简单的情况，即 CSLA 不同分支卷积核的大小一致，且各个 channel 的 scale 值相同。更复杂一些的情况是不同分支卷积核有不同的尺寸，且各个 channel 的 scale 值不同，此时 Grad Mask 将是一个 Tensor，并且应分别使用相应位置上的 scale 值计算里面的内容。

作者提出了对应于这样一个 CSLA 块的 Grad Mask 的公式，这个公式用来训练单个 3×3 的卷积。令 C 为通道数， $\mathbf{s}, \mathbf{t} \in \mathbb{R}^C$ 分别为 3×3 和 1×1 层之后的 scale 向量。Grad Mask $\mathbf{M}^{C\times C\times 3\times 3}$ 通过下式来构建：

$$\mathbf{M}^{C\times C\times 3\times 3} = \begin{cases} 1+\mathbf{s}_c^2+\mathbf{t}_c^2 & \text{if } c=d, p=2 \text{ and } q=2, \\ \mathbf{s}_c^2+\mathbf{t}_c^2 & \text{if } c\neq d, p=2 \text{ and } q=2. \\ \mathbf{s}_c^2 & \text{elsewise} \end{cases} \tag{13}$$

以上13式其实就是在计算梯度的时候用了一个 Trick，也很好理解：就是假设每个 CSLA 模块有 3×3 分支，1×1 分支和 Identity mapping 分支，那么 $C\times C\times 3\times 3$ 的权重的梯度在计算时：

当 $c=d, p=2 \text{ and } q=2$ 时，对应的梯度是3个分支的梯度之和，所以按照以上命题的结论，就应该是 $1+\mathbf{s}_c^2+\mathbf{t}_c^2$ 。

当 $c\neq d, p=2 \text{ and } q=2$ 时，对应的梯度是 3×3 和 1×1 这两个分支的梯度之和，所以按照以上命题的结论，就应该是 $\mathbf{s}_c^2+\mathbf{t}_c^2$ 。

其他情况，对应的梯度只有 3×3 这个分支的梯度，按照以上命题的结论，就应该是 $\mathbf{s}_c^2$ 。

与 RepVGG 相比，CSLA 模块没有 BN 这种非线性层。此外，必须强调的是，CSLA 模块的结构，即 3×3 分支，1×1 分支和 Identity mapping 分支都是虚构的，因为虚构的多分支架构的梯度与 GR 的结果在数学上是一样的。

1.1.6 RepOpt 的第二步：通过超搜索获得超参数

下面的问题是：常数 scaling 向量 $\mathbf{s}$ 和 $\mathbf{t}$ 如何得到?

到目前为止我们知道的是：常数 scaling 向量 $\mathbf{s}$ 和 $\mathbf{t}$ 会直接影响 Grad Masks 的值，并且会影响优化器 RepOptimizer 的性能。但是这个常数向量 $\mathbf{s}$ 和 $\mathbf{t}$ 到底应该设置成多少？其实我们没办法从数学上面找到一个最优的解。因此作者借鉴了神经架构搜索的方法，将优化器的超参数 (向量 $\mathbf{s}$ 和 $\mathbf{t}$) 与 CSLA 模块的可训练参数相关联，因此被称为**超搜索 (Hyper-Search)**。

超搜索 (Hyper-Search) 具体是怎么做的呢?

作者借鉴了 DARTS[13] 的方法：给定一个 RepOptimizer，我们通过用**可训练**的向量 $\mathbf{s}$ 和 $\mathbf{t}$ 替换 RepOptimizer 对应的 CSLA 模型中的常数向量 $\mathbf{s}$ 和 $\mathbf{t}$，并在 CIFAR100 上用 DARTS 办法搜索它 (本 使用的值。即满足：可

的。

1.1.7 RepOpt 的第三步：使用 RepOpt 进行训练

在超搜索之后，作者使用搜索到的向量 $\mathbf{s}$ 和 $\mathbf{t}$ 构造 RepOptimizer 的 Grad mask，这些 mask 被存储在内存中。在训练的过程中把 Grad mask element-wise 地乘到梯度上面，整个过程的 pipeline 如下图2所示。

使用 RepOptimizer 开始训练之前，根据搜索的超参数 (α_A, α_B) 重新初始化模型的参数：$\mathbf{W}'^{(0)} \leftarrow \alpha_A \mathbf{W}^{(A)(0)} + \alpha_B \mathbf{W}^{(B)(0)}$ 。具体而言，初始化参数可以仿照13式设置为：

$$\mathbf{W}'^{(0)}_{c,d,p,q} = \begin{cases} 1 + \mathbf{s}_c \mathbf{W}^{(s)(0)}_{c,d,p,q} + \mathbf{t}_c \mathbf{W}^{(t)(0)}_{c,d,p,q} & \text{if } c = d, p = 2 \text{ and } q = 2, \\ \mathbf{s}_c \mathbf{W}^{(s)(0)}_{c,d,p,q} + \mathbf{t}_c \mathbf{W}^{(t)(0)}_{c,d,p,q} & \text{if } c \neq d, p = 2 \text{ and } q = 2. \\ \mathbf{s}_c \mathbf{W}^{(s)(0)}_{c,d,p,q} & \text{elsewise} \end{cases} \tag{14}$$

作者使用 $\mathbf{W}'^{(0)}$ 初始化 RepOpt-VGG 模型中的 3×3 Conv 层，这样训练动态就相当于训练一个用 $\mathbf{W}^{(s)(0)}_{c,d,p,q}$ 和 $\mathbf{W}^{(t)(0)}_{c,d,p,q}$ 初始化的 CSLA 块。注意初始化方法 (14式) 以及更新方式 (2，13式) 必须严格遵守才可以保证 CSLA=GR，否则对精度会有影响。

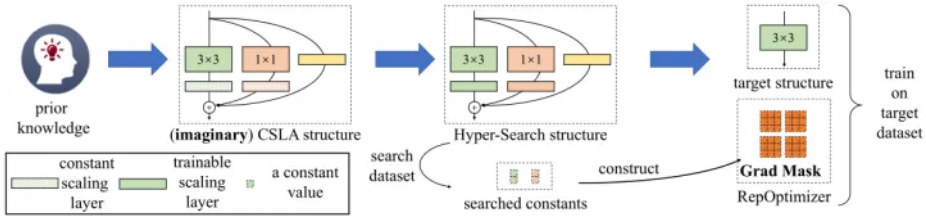


图2：使用 RepOpt 的整体 pipeline

1.1.8 RepOpt 实验设置

作者采用 RepVGG 作为主要基线模型，因为它是个简单而强大的 CNN。RepOpt-VGG 可以达到几乎与 RepVGG 相同的精度，但训练速度更快，内存消耗更低，并与其他 SOTA 的模型表现相当。

RepOpt-VGG 的架构如下图3所示，各个架构都是由4个 stage 和每个 stage 里面的一堆 3×3 Conv 组成。为了公平比较，RepOpt-VGG 采用了与 RepVGG 相同的极简架构设计。

Model	3×3 layers	Channels	FLOPs (B)	Param (M)
B1	4, 6, 16, 1	128, 256, 512, 2048	11.9	51.8
B2	4, 6, 16, 1	160, 320, 640, 2560	18.4	80.3
L1	8, 14, 24, 1	128, 256, 512, 2048	21.0	76.0
L2	8, 14, 24, 1	160, 320, 640, 2560	32.8	118.1

图3：RepOpt-VGG 架构

使用 8 GPUs，每个 GPU 上的 Batch Size 为32，224×224 的输入分辨率，5-epoch warm-up，cosine annealing 的学习率衰减策略，一共训练 120 epoch。

作者使用 CIFAR100 完成超搜索的过程，在单个 GPU 上以 256 的 Batch Size，并且使用与 ImageNet 相同的学习率策略训练 240 个 Epoch。值得注意的是，HS 模型的可训练的向量 $\mathbf{s}$ 和 $\mathbf{t}$ 是根据深度 l 进行初始化的。具体来说，设 l 为层的深度索引 (即每个 stage 的第一层有 $l = 1$，下一层有 $l = 2$)，可训练的向量被初始化为 $\sqrt{\frac{2}{l}}$ 。

这种初始化方案来自这样的一个观察，就是对于一个包含残差连接的深度 CNN (ResNet, RepVGG) 而言，Shortcut 的效果应该在模型的深层发挥更大的效用。换句话说，更深层的功能应该更像一个 Identity mapping，以方便模型的训练。直观地说，在训练的早期阶段，浅层学习最多，而深层学习没那么多，因为如果没有来自浅层的良好特征，深层就无法很好地学习。

的方差/总和的方差。这个值越大就代表 Identity mapping 路径的值占据主导，即该层的 Identity mapping 分支比较重要；反之，这个值越小就代表另一个残差分支的值占据主导，即该层的 Identity mapping 路径分支不那么重要。

ResNet 模型的 identity variance ratio 是逐渐增加的，这种现象是意料之中的，因为每个残差块的输出均值为0，方差为1 (因为它通过了 BN 层)，在每次 Shortcut-addition 之后，identity 路径的方差增加，因此 Identity mapping 路径的方差/总和的方差也逐渐增加，故 identity variance ratio 也逐渐增加。

但是对于 Hyper Search 的模型，我们没办法观察到相似的模式，因为这里的3个分支都没有 BN 操作，三个分支的方差都随着深度的增加而变大，从而保持了恒等的 identity variance ratio。因此，为了产生类似的模式，作者根据深度进行模型的初始化，使得 Hyper Search 的模型拥有和 ResNet 相似的模式。

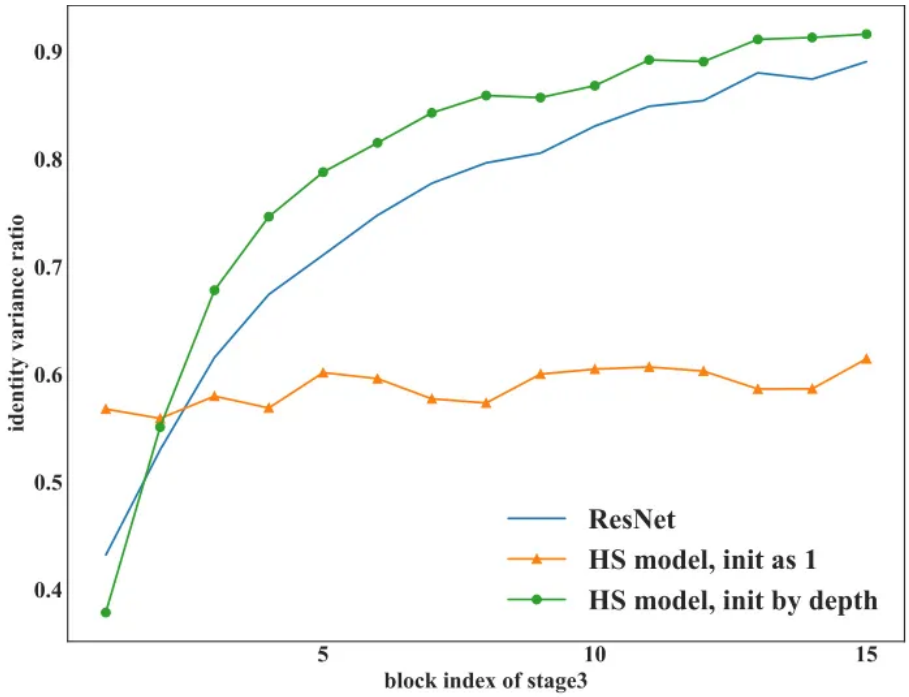


图4：不同模型的 identity variance ratio

1.1.9 RepOpt 实验结果

如下图5所示为 RepVGG 和 RepOpt 模型的训练速度对比，以及在 2080 Ti 上所允许的最大 Batch Size，和在不同 Batch Size 下的精度。

Model	Train MaxBS	Top-1 acc @BS=32	Top-1 acc @MaxBS	Train speed @BS=32	Train speed @MaxBS	Train param (M)	Inference param (M)
RepVGG-B1	198	78.42	78.67	243	213	57.4	51.8
RepOpt-VGG-B1	**260**	78.48	78.66	445	**380**	**51.8**	51.8
RepVGG-B2	153	79.50	79.76	163	142	89.0	80.3
RepOpt-VGG-B2	**200**	79.39	79.72	264	**246**	**80.3**	80.3
RepVGG-L1	106	79.49	79.84	136	124	84.3	76.0
RepOpt-VGG-L1	**140**	79.46	79.88	252	**221**	**76.0**	76.0
RepVGG-L2	77	80.58	80.68	86	82	131.0	118.1
RepOpt-VGG-L2	**103**	80.31	80.66	149	**138**	**118.1**	118.1

图5：RepVGG 和 RepOpt 模型的训练速度对比，以及在 2080 Ti 上所允许的最大 Batch Size，和在不同 Batch Size 下的精度

1) RepOpt-VGG 消耗的内存更少，训练速度更快：使用各自的 MaxBS，RepOpt-VGG-B1 的训练速度是 RepVGG-B1 的 1.7 倍。

2) 随着 Batch Size 的增大，每个模型的性能都有明显的提升，这可以解释为 BN 具有更好的稳定性。

Model	Train epochs	Train resolution	Test resolution	Top-1 acc	Test batch size	Throughput (samples/second)
RegNetX-3.2GF	120	224	224	78.43	128	988
RepOpt-VGG-B1	120	224	224	78.48	128	970
EfficientNet-B2	120	260	260	76.29	128	912
RepOpt-VGG-B2	120	224	224	79.39	128	645
RegNetX-6.4GF	120	224	224	79.67	128	589
RepOpt-VGG-L2	120	224	224	**80.31**	128	372
EfficientNet-B3	120	300	300	79.12	128	301
RepOpt-VGG-L2‡	**200**	**224**	**320**	**83.22**	64	180
EfficientNet-B4	300	380	380	82.02	64	183
EfficientNet-B5	300	456	456	82.96	64	91

图6：RepOpt-VGG 的与其他 SOTA 模型的精度比较

下图7是关于 RepOpt 的一些对比实验，只有在常数向量 $\mathbf{s}$ 和 $\mathbf{t}$ 是有 Hyper-Search 得到，且初始化改变，以及使用 Grad Mask 时，模型性能才是最高的，因为除此之外的其他情况，模型的 CSLA 都不完全等价于 GR。

Source of constants	Change initialization	Modify gradients	Top-1 acc of RepOpt-VGG-B1
Hyper-Search	✓	✓	**78.48**
Hyper-Search		✓	77.42
Hyper-Search	✓		77.07
None			76.91
All $\mathbf{1}$	✓	✓	77.59
Same as HS initialization	✓	✓	77.71
Average across channels	✓	✓	77.40

图7：RepOpt 对比实验

下图8是在下游任务上面验证 RepOpt-VGG 的性能，RepOpt-VGG 性能始终与 RepVGG 相当。

Backbone	COCO mAP	Cityscapes mIoU
RepOpt-VGG-B1	43.50	79.36
RepVGG-B1	43.60	79.15

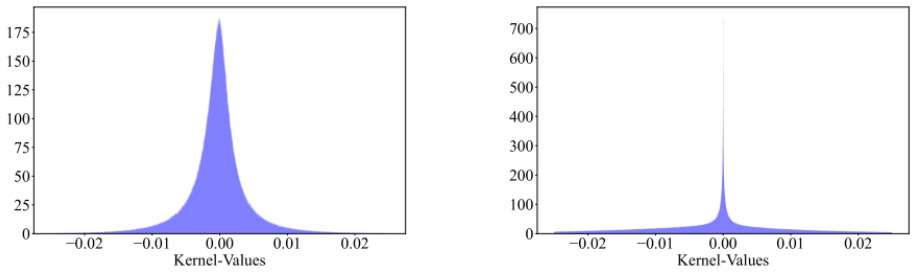
图8：下游任务实验结果

量化 RepVGG 模型会导致显著的精度下降，例如，使用简单的 PTQ (后训练量化) 方法将 RepVGG-B1 量化为 INT8 会将 Top-1 准确率降低到 54.55% 左右，从而使模型无法使用。但是，直接量化 RepOpt-VGG 只会导致 2.5% 的精度下降。作者分析了二者的权重分布，结果如下图9和10所示。

通过图10可以看到 RepVGG 和 RepOpt-VGG 的参数分布明显不同，且前者的标准差是后者的4倍左右。这是由于多个分支相加所导致的，而 RepOpt-VGG 很自然地解决了这个问题，因为它根本没有多个分支相加。

Model	Quantized accuracy	Standard deviation of parameters
RepOpt-VGG-B1	75.89	0.0066
RepVGG-B1	54.55	0.0234

图9：RepOpt-VGG 和 RepVGG 参数的量化精度和标准差



(a) RepOpt-VGG. (b) RepVGG.

图10：RepOpt-VGG 和 RepVGG 的 3×3 核的参数分布

在现代模型设计的加持下，人们往往会根据经验去习惯性地使用一些 "已经被验证过有效" 的优化器，如 SGD，AdamW 等。本文作者将重参数化技术重新在优化器中使用，提出 RepOptimizer。我们知道，优化过程最重要的变量就是梯度，RepOptimizer 根据一组模型特定的超参数修改梯度来添加先验知识，这个过程也称为梯度重参数化 (Gradient Re-parameterization)。作者仅仅使用了一个极简的 VGG-Style 模型，通过 RepOptimizer 训练出的 RepOpt-VGG 推理速度快，训练效率高，同时性能依旧强悍，是一个很好的基础模型。

作者希望通过这篇论文，大家不仅仅是去精心设计模型架构，而是唤起对训练策略的研究。尽管该方法得到了实验上的验证，但是仍需进一步的理论研究。此外我们必须强调的是，本文列出的 CSLA 和梯度重参数化等效的情况仅仅是一种特殊的模型，对于更加泛化的模型 (比如有 BN 层，多个模块顺序连接的情况) 仍然不能直接使用梯度重参数化来替换。

参考

1. ^ Deep residual learning for image recognition
2. ^ V ery deep convolutional networks for large-scale image recognition
3. ^ Efficientnet: Rethinking model scaling for convolutional neural networks
4. ^ Designing network design spaces
5. ^ A stochastic approximation method
6. ^ Adam: A method for stochastic optimization
7. ^ Decoupled weight decay regularization
8. ^ Conditioning of quasi-newton methods for function minimization
9. ^ Structured quasi-newton methods for optimization with orthogonality constraints
10. ^ Compatible natural gradient policy search
11. ^ Repvgg: Making vgg-style convnets great again
12. ^ Fast algorithms for convolutional neural networks
13. ^ DARTS: Differentiable Architecture Search

编辑于 2022-10-06 18:10

优化

写下你的评论...

4 条评论 默认 最新

鹿小臻
不知道博主有没有看过git公开的代码，训练RepOptVGG的时候，是不是要保证参数初始化时要用训练csla block的参数去做修改初始化，但是我看代码的时候没有看到哪里用了csla模型的初始化参数😳
09-15 · IP 属地江苏 赞

鹿小臻
没事了，我又仔细去看了下论文附件部分Initialization of the Target Model，作者解释了用两种初始化方式也没有关系
09-15 · IP 属地江苏 赞

阳仔的下午茶
感谢博主，很棒的文章！我有个疑问想请教一下，为什么需要超参搜索s和t呢？我理解只要参数初始化满足式(3)，参数更新满足式(2)，两者使用相同的s和t就可以了。不同的s和t只是不同的参数初始化，应该不会对指标产生太大的影响吧
09-13 · IP 属地河北 赞

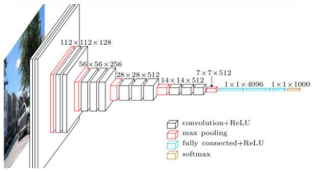
科技猛兽 作者
感谢赞同哈哈。我认同你的这句话 "我理解只要参数初始化满足式(3)，参数更新满足式(2)，两者使用相同的s和t就可以了"。但是这个s和t，随便设的我觉得不如作者搜出来的好。他后面也有做其他情况的s和t，都掉点了~
09-13 · IP 属地上海 赞

推荐阅读

迁移学习+VGG模型微调+测试集测试

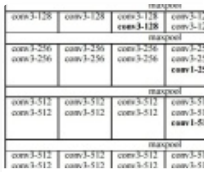
VGG+flowers photo(复现VGG模型，并用flower photo数据集微调参数，在新的验证集上验证网络模型的正确性) 目的：图像分类所用的模型：VGG16所用的数据集：flower_photo(数据集中有五种花…

玉米牛奶



《VGG网络学习（-），原理解析》。

时间之外的往事



从零开始搭建VGG网 Keras

胡卫雄 发表于