基于深度残差学习的图像识别

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

更深的神经网络更难训练。本文提出一种残差学习框架，以简化网络的训练，这些网络比以前使用的网络要深得多。我们明确地根据层的输入将层重新表述为学习残差函数，而不是学习未引用的函数。提供了全面的经验证据，表明这些残差网络更容易优化，并可以从大幅增加的深度中获得精度。在*ImageNet*数据集上，我们评估了深度高达*152*层的残差网络——比*VGG*网络*[41]*深*8* ×，但仍然具有较低的复杂度。这些残差网络的集成在*ImageNet*测试集上取得了*3.57*%的误差。该结果在*ILSVRC 2015*分类任务中获得第一名。本文还对具有*100*层和*1000*层的*CIFAR-10*进行了分析。

表示的深度对许多视觉识别任务至关重要。仅仅由于极其深度的表示，在*COCO*目标检测数据集上获得了*28*%的相对改进。深度残差网络是我们提交的*ILSVRC* &*COCO 2015*竞赛[1]的基础，在那里我们还在*ImageNet*检测、*ImageNet*定位、*COCO*检测和*COCO*分割任务上获得了第一名。

1. 简介

深度卷积神经网络[22, 21]为图像分类带来了一系列的突破[21, 50, 40]。深度网络自然地以端到端的多层方式集成低/中/高级特征[50]和分类器，并且特征的“层次”可以通过堆叠层的数量(深度)来丰富。最近的证据[41, 44]揭示了网络深度是至关重要的，而在具有挑战性的ImageNet数据集[36]上的领先结果[41, 44, 13, 16]都利用了“非常深”[41]模型，深度为16 [41]到30 [16]。许多其他重要的视觉识别任务[8, 12, 7, 32, 27]也从非常深的模型中受益匪浅。

受深度重要性的驱动，一个问题出现了:学习更好的网络是否容易堆叠更多的层? 回答这个问题的一个障碍是臭名昭著的梯度消失/爆炸问题[1, 9]，它从一开始就阻碍了收敛。然而，这个问题已经在很大程度上通过规范化初始化[23, 9, 37, 13]和中间规范化层[16]得到了解决，这使得具有数十层的网络开始收敛于反向传播的随机梯度下降(SGD) [22]。

[1]http://image-net.org/challenges/LSVRC/2015/ 和http://mscoco.org/dataset/#detections-challenge2015。

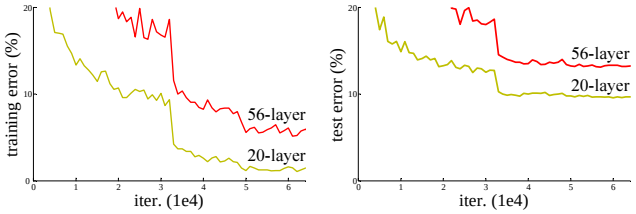

Figure 1. 在具有20层和56层“普通”网络的CIFAR-10上的训练误差(左)和测试误差(右)。网络越深，训练误差越大，测试误差也越大。ImageNet上的类似现象如图4所示。

当更深的网络能够开始收敛时，一个退化问题就暴露出来了:随着网络深度的增加，精度达到饱和(这可能并不奇怪)，然后迅速退化。意外的是，这种退化不是由过拟合引起的，向适当深度的模型添加更多层会导致更高的训练误差，如[11, 42]所报道的，并通过我们的实验进行了彻底验证。图1展示了一个典型的例子。

(训练精度)的下降表明并非所有系统都同样容易优化。让我们考虑一个较浅的体系结构，以及在其上添加更多层的较深的对应结构。存在通过构建较深模型的解决方案:添加的层是身份映射，其他层是从学习的较浅模型复制的。这种构造的解决方案的存在表明，较深的模型不应该产生比较浅的对应模型更高的训练误差。但是实验表明，我们现有的求解器无法找到比构建的解决方案更好或更好的解决方案(或无法在可行时间内做到这一点)。

本文通过引入深度残差学习框架来解决退化问题。我们不希望每个堆叠的层直接适合所需的基础映射，而是明确地让这些层适合残差映射。形式上，将所需的底层映射表示为$\mathcal{H}(\mathbf{x})$，我们让堆叠的非线性层适合$\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$的另一个映射。原始映射被重铸为$\mathcal{F}(\mathbf{x}) + \mathbf{x}$。我们假设优化残差映射比优化原始的、未引用的映射更容易。在极端情况下，如果单位映射是最优的，那么将残差推到零比通过一堆非线性层来拟合单位映射更容易。

$\mathcal{F}(\mathbf{x})+\mathbf{x}$的公式化可以通过具有“快捷连接”的前馈神经网络实现(图2)。快捷连接[2, 34, 49]是指跳过一层或多层的连接。在我们的例子中，快捷连接只是执行身份映射，它们的输出被添加到堆叠层的输出中(图2)。标识快捷连接既不增加额外参数，也不增加计算复杂性。整个网络仍然可以通过SGD的反向传播进行端到端的训练，并且可以使用公共库(*e.g.*, Caffe [19])轻松实

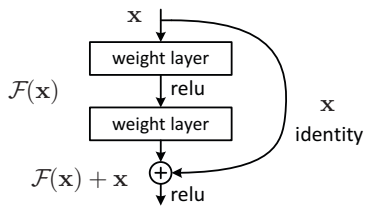

Figure 2. 残差学习:构建模块。

现，而无需修改求解器。

在ImageNet [36]上进行了全面的实验，以显示退化问题，并评估了所提出方法。实验表明:1)极深的残差网络易于优化，但对应的“普通”网络(只是堆叠层)在深度增加时表现出更高的训练误差;2)深度残差网络可以很容易地从深度的大幅增加中获得精度增益，产生的结果大大优于之前的网络。

在CIFAR-10集[20]上也显示了类似的现象，这表明优化困难和我们方法的效果不仅类似于特定的数据集。本文提出了在这个数据集上成功训练的模型，有100多层，并探索了有1000多层的模型。

在ImageNet分类数据集[36]上，我们通过极深的残差网络获得了出色的结果。我们的152层残差网络是ImageNet上有史以来最深的网络，同时仍然比VGG网络[41]的复杂度低。该集成在ImageNet测试集上有**3.57**%的top-5错误，并在*ILSVRC 2015*分类竞赛中获得了第一名。极深的表示在其他识别任务上也有出色的泛化性能，并使我们在ILSVRC &*COCO 2015*比赛中进一步赢得了第一名:*ImageNet*检测，*ImageNet*定位，*COCO*检测和*COCO*分割。这一强有力的证据表明，残差学习原理是通用的，并期望它适用于其他视觉和非视觉问题。

2. 相关工作

残差表示。在图像识别中，VLAD [18]是一种通过相对于字典的残差向量进行编码的表示，Fisher向量[30]可以表示为VLAD的概率版本[18]。它们都是用于图像检索和分类的强大的浅层表示[4, 48]。对于矢量量化，编码残差矢量[17]比编码原始矢量更有效。

在低级视觉和计算机图形学中，为了求解偏微分方程(PDEs)，广泛使用的多重网格方法[3]将系统重新表述为多个尺度的子问题，其中每个子问题负责较粗和较细尺度之间的残差解决方案。多重网格的一个替代方案是分层基预处理[45, 46]，它依赖于表示两个尺度之间的残差向量的变量。已经表明[3, 45, 46]这些求解器比不知道解决方案的剩余性质的标准求解器收敛得快得多。这些方法表明，良好的重构或预处理可以简化优化。

快捷连接。导致捷径连接的 实践和理论[2, 34, 49]已经被研究了很长时间。训练多层感知器(MLPs)的早期实践是添加一个从网络输入连接到输出的线性层[34, 49]。在[44, 24]中，一些中间层直接连接到辅助分类器，以解决梯度消失/爆炸问题。[39, 38, 31, 47]的论文提出了通过快捷连接实现对层响应、梯度和传播误差进行中心化的方法。在[44]中，一个“inception”层由一个快捷分支和几个更深的分支组成。

与我们的工作同时，“高速公路网络”[42, 43]提供了与门控功能的快捷连接[15]。这些门依赖于数据，并且有参数，而我们的标识快捷方式是无参数的。当门控捷径被“关闭”(接近零)时，高速公路网络中的层代表非残差函数。相反，我们的公式总是学习残差函数;我们的身份快捷方式永远不会关闭，所有信息总是被传递，还有额外的残差功能需要学习。此外，高速公路网络并没有表现出深度极大增加(*e.g.*，超过100层)带来的精度提高。

3. 深度残差学习

3.1. 残差学习

让我们考虑$\mathcal{H}(\mathbf{x})$作为底层映射，由几个堆叠的层(不一定是整个网络)拟合，$\mathbf{x}$表示对第一个层的输入。如果假设多个非线性层可以渐进地逼近复杂函数[2]，那么就等价于假设它们可以渐进地逼近残差函数*i.e.*, $\mathcal{H}(\mathbf{x}) - \mathbf{x}$(假设输入和输出具有相同的维度)。因此，与其期望堆叠的层近似$\mathcal{H}(\mathbf{x})$，我们显式地让这些层近似残差函数$\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$。原来的功能因此变成了$\mathcal{F}(\mathbf{x}) + \mathbf{x}$。尽管两种形式都应该能够渐进地逼近所需的函数(如假设的那样)，但学习的难度可能不同。

这种重新表述的动机是关于退化问题的反直觉现象(图1，左)。正如我们在介绍中讨论的那样，如果添加的层可以构建为身份映射，则更深的模型的训练误差应该不大于较浅的模型。退化问题表明求解者可能难以通过多个非线性层逼近单位映射。通过残差学习重构，如果单位映射是最优的，求解器可以简单地将多个非线性层的权重趋近于零，以接近单位映射。

在实际情况中，标识映射不太可能是最优的，但我们的重新表述可能有助于解决问题。如果最优函数更接近恒等映射而不是零映射，则求解器应该更容易找到恒等映射的扰动，而不是学习一个新的函数。通过实验表明(图7)，学习到的残差函数通常具有较小的响应，这表明身份映射提供了合理的预处理。

3.2. 通过快捷方式进行身份映射

对每隔几层进行残差学习。一个构建块如图2所示。形式上，本文考虑一个构建块定义为:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}. \tag{1}$$

这里$\mathbf{x}$和$\mathbf{y}$是所考虑的层的输入和输出向量。函数$\mathcal{F}(\mathbf{x}, \{W_i\})$表示要学习的残差映射。对于图. 2中的示例，它有两层，$\mathcal{F} = W_2\sigma(W_1\mathbf{x})$其中$\sigma$表示ReLU [29]，为了简化符号，偏差被省略。$\mathcal{F} + \mathbf{x}$操作是通过快捷连接和元素添加来执行的。我们采用加法后的第二次非线性(*i.e.*, $\sigma(\mathbf{y})$，见图2)。

Eqn中的快捷连接。(1)不会引入额外的参数，也不会增加计算复杂度。这不仅在实践中很有吸引力，

[2]然而，这个假设仍然是一个开放的问题。参见[28]。

而且在我们比较普通网络和残差网络时也很重要。我们可以公平地比较同时具有相同参数数量、深度、宽度和计算成本的普通/残差网络(除了可忽略的元素添加)。

在Eqn中$\mathbf{x}$和$\mathcal{F}$的尺寸必须相等。(1)。如果不是这样(*e.g.*，当改变输入/输出通道时)，我们可以通过快捷连接执行线性投影W_s来匹配维度:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}. \tag{2}$$

我们也可以在Eqn中使用方阵W_s。(1)。但是，我们将通过实验表明，身份映射足以解决退化问题，并且是经济的，因此W_s仅在匹配尺寸时使用。

残差函数$\mathcal{F}$形式灵活。本文的实验涉及一个函数$\mathcal{F}$，它有两层或三层(图5)，也可以有更多层。但是如果$\mathcal{F}$只有一层，那么Eqn。(1)类似于线性层:$\mathbf{y} = W_1\mathbf{x} + \mathbf{x}$，对于它，我们没有观察到优势。

我们还注意到，尽管为简单起见，上述表示法是关于全连接层的，但它们适用于卷积层。$\mathcal{F}(\mathbf{x}, \{W_i\})$函数可以表示多个卷积层。在两个特征图上逐通道执行元素相加。

3.3. 网络体系结构

我们测试了各种普通/残差网络，并观察到一致的现象。为了提供讨论的实例，我们描述了ImageNet的两个模型，如下所示。

普通网络。我们简单的基线(图3，中)主要受到VGG nets [41](图3，左)的哲学启发。卷积层大多有3 × 3个滤波器，并遵循两个简单的设计规则:(i)对于相同的输出特征图大小，层具有相同数量的滤波器;(ii)如果特征图大小减半，滤波器的数量将增加一倍，以保持每层的时间复杂度。我们直接通过步幅为2的卷积层进行下采样。该网络以一个全局平均池化层和一个带有softmax的1000路全连接层结束。在图3(中间)中，加权层的总数为34。

值得注意的是，我们的模型比VGG网络[41]具有更少的过滤器和更低的复杂度(图3，左)。我们的34层基线有36亿FLOPs(乘法加法)，仅为VGG-19的18%(196亿FLOPs)。

残差网络。在上述普通网络的基础上，我们插入快捷连接(图3，右)，将网络转换为对应的残差版本。身份快捷方式(Eqn。(1))，当输入和输出尺寸相同时，可直接使用(图3中的实线快捷键)。当尺寸增加时(图3中的虚线快捷方式)，我们考虑两种选择: (A)快捷方式仍然执行恒等映射，为增加维度填充额外的零条目。这个选项不引入额外的参数; (B) Eqn中的投影捷径。(2)用于匹配尺寸(通过1 × 1个卷积完成)。对于这两个选项，当快捷方式跨越两个大小的特征映射时，它们以2的步幅执行。

3.4. 实现

我们对ImageNet的实现遵循[21, 41]中的实践。图像被调整大小，其较短的边在[256, 480]中随机采样以进

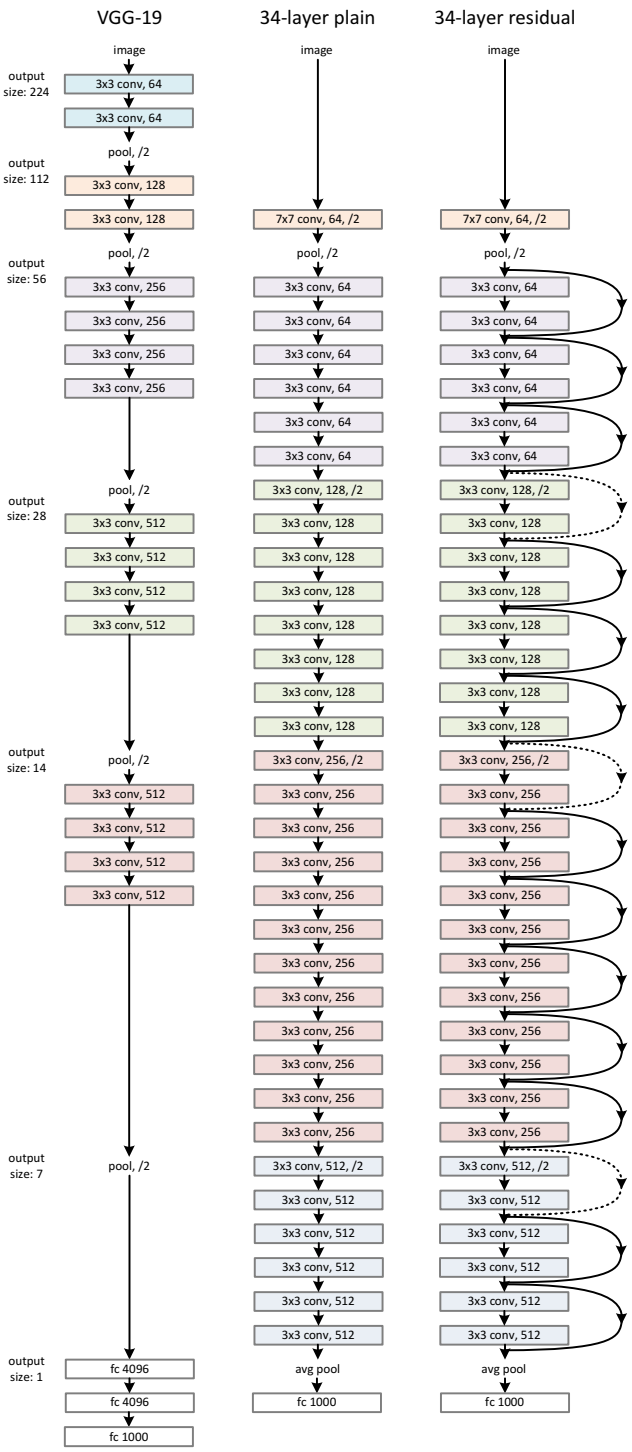

Figure 3. ImageNet网络架构示例。左:VGG-19模型[41](196亿FLOPs)作为参考。中间:具有34个参数层(36亿FLOPs)的普通网络。右:具有34个参数层(36亿FLOPs)的残差网络。虚线的快捷方式增加了尺寸。表**1**显示了更多的细节和其他变体。

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. ImageNet的架构。构建块显示在括号中(参见图5)，块的数量堆叠在一起。conv3_1, conv4_1和conv5_1进行下采样，步长为2。

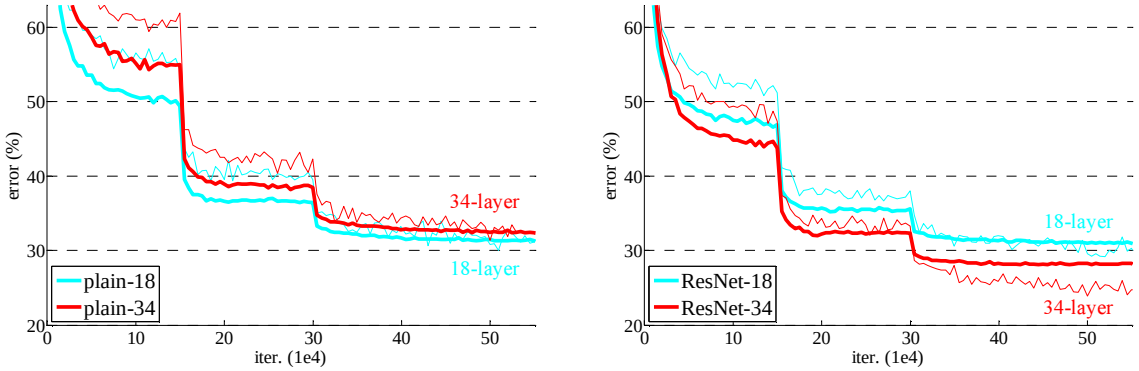

Figure 4. 在**ImageNet**上进行训练。细曲线表示训练误差，粗曲线表示中心作物的验证误差。左:18层和34层的普通网络。右:18层和34层的ResNets。在这张图中，与普通网络相比，残差网络没有额外的参数。

行缩放[41]。224 × 224裁剪是从图像或其水平翻转中随机采样，并减去每个像素的平均值[21]。使用[21]中的标准颜色增强。我们在每次卷积之后和激活之前采用批量归一化(BN) [16]，随后[16]。我们像[13]一样初始化权重，并从头开始训练所有普通/残差网络。我们使用小批量大小为256的SGD。学习率从0.1开始，当误差停滞时除以10，并对模型进行最多60×10^4次迭代训练。我们使用0.0001的权重衰减和0.9的动量。我们不使用dropout [14]，遵循[16]的做法。

在测试中，为了比较研究，我们采用了标准的10种作物测试[21]。为了获得最佳结果，我们采用了[41, 13]中的全卷积形式，并在多个尺度上平均分数(图像被调整大小，以便较短的边在{224, 256, 384, 480, 640}中)。

4. 实验

4.1. ImageNet分类

在ImageNet 2012分类数据集[36]上评估了该方法，该数据集由1000个类组成。模型在128万张训练图像上进行训练，并在50万张验证图像上进行评估。我们还获得了测试服务器报告的100k测试图像的最终结果。我们评估了top-1和top-5错误率。

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

Table 2. ImageNet验证的Top-1误差(%，10作物测试)。与普通的ResNets相比，这里的ResNets没有额外的参数。图4为培训流程。

简单的网络。首先评估了18层和34层的普通网络。34层的普通网络如图3(中间)所示。18层的普通网络也是类似的形式。详细架构参见表1。

表2中的结果表明，较深的34层普通网络比较浅的18层普通网络具有更高的验证误差。为了揭示原因，在图4(左)中，我们比较了他们在训练过程中的训练/验证错误。我们已经观察到退化问题——34层的普通网络在整个训练过程中具有更高的训练误差，即使18层的普通网络的解空间是34层解空间的子空间。

本文认为，这种优化困难不太可能是由梯度消失造成的。这些普通网络使用BN [16]进行训练，确保前向传播的信号具有非零方差。还验证了反向传播梯度在BN中表现出健康的规范。所以前进和后退的信号都不会消失。事实上，34层的普通网络仍然能够达到有竞争力的精度(表3)，这表明求解器在某种程度上是有

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 3. ImageNet验证的错误率(%，**10**作物测试)。VGG-16是基于我们的测试。ResNet-50/101/152属于选项B，它只使用增加维度的投影。

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. ImageNet验证集上单一模型结果的错误率(%)(测试集上报告的†除外)。

method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Table 5. 集成的错误率(%)。top-5错误发生在ImageNet的测试集上，由测试服务器报告。

效的。我们推测，深度普通网络可能具有指数级低收敛率，这影响了训练误差的减少[3]。出现这种优化困难的原因将在以后进行研究。

残差网络。接下来，我们评估了18层和34层残差网络*(ResNets)*。基线架构与上面的普通网络相同，除了为每对3 × 3过滤器添加一个快捷连接外，如图3(右)所示。在第一个比较中(表2和图4右)，我们对所有快捷方式使用标识映射，对增加维度使用零填充(选项A)。因此，与普通的对应方法相比，它们没有额外的参数。

从表2和图4中我们有三个主要的观察结果。首先，残差学习扭转了这种情况——34层的ResNet比18层的ResNet要好(提高2.8%)。更重要的是，34层的ResNet表现出相当低的训练误差，并且可以泛化到验证数据。这表明在这种情况下退化问题得到了很好的解决，并设法从增加深度中获得了精度增益。

其次，与普通的对应模型相比，34层的ResNet将top-1误差降低了3.5%(表2)，这是成功减少训练误差的结果(图4右*vs.*左)。这种比较验证了残差学习在极深系统上的有效性。

最后，我们还注意到，18层普通/残差网络同样准确(表2)，但18层ResNet收敛更快(图4右*vs.*左)。当网络"不是过深"(这里有18层)时，当前的SGD求解器仍然能够找到普通网络的良好解决方案。在这种情况下，ResNet通过在早期阶段提供更快的收敛速度来简化优化。

身份***vs.***投影快捷键。我们已经表明，无参数的身份快捷方式有助于训练。接下来，我们研究投影捷径(Eqn。(2))。在表3中，我们比较了三个选项:(A)零填充快捷键用于增加维度，并且所有快捷键都是无参数的(与表2和图4相同);(B)投影快捷方式用于增加维度，其他快捷方式为同一性;(C)所有的捷径都是投影。

表3显示了这三个选项都比简单的对应选项要好得多。B略好于A。我们认为这是因为A中的零填充维度确实没有残差学习。C略好于B，我们将这归因于许多(13)投影快捷方式引入的额外参数。但A/B/C之间的微小差异表明，投影捷径对于解决退化问题并不是必要的。因此，在本文的其余部分中，我们不使用选项C，以降低内存/时间复杂度和模型大小。标识快捷方式特别重要，因为它不会增加下面介绍的瓶颈体系结构的复杂性。

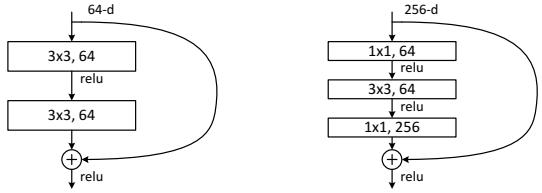

Figure 5. 更深层次的残差函数$\mathcal{F}$用于ImageNet。左:一个构建块(在56 × 56特征图上)，如图3中的ResNet-34。右:ResNet-50/101/152的"瓶颈"构建模块。

更深层次的瓶颈架构。接下来，我们描述ImageNet的深层网络。由于考虑到我们可以负担的训练时间，我们修改构建模块作为瓶颈设计[4]。对于每个残差函数$\mathcal{F}$，我们使用3层的堆栈而不是2层(图5)。这三层是1 × 1, 3 × 3和1 × 1卷积，其中1 × 1层负责减少然后增加(恢复)维度，使3 × 3层成为输入/输出维度更小的瓶颈。图5显示了一个例子，其中两种设计具有相似的时

[3]我们进行了更多的训练迭代(3 ×)，仍然观察到退化问题，这表明该问题不能通过简单地使用更多的迭代来解决。

[4]更深的非瓶颈ResNets (*e.g.*，图5左)也可以从增加深度中获得精度(如CIFAR-10所示)，但不如瓶颈ResNets经济。因此，使用瓶颈设计主要是出于实际考虑。我们进一步注意到，在瓶颈设计中，普通网络也存在退化问题。

间复杂度。

无参数的标识快捷方式对于瓶颈体系结构尤为重要。如果将图5(右)中的标识快捷方式替换为投影，可以表明时间复杂度和模型大小都增加了一倍，因为快捷方式连接到两个高维端点。因此，身份快捷方式为瓶颈设计带来了更有效的模型。

50层**ResNet:**我们将34层网络中的每个2层块替换为这个3层瓶颈块，得到一个50层的ResNet(表1)。我们使用选项B来增加维度。这个模型有38亿次失败。

101层和**152**层**ResNets:**我们通过使用更多的3层块来构建101层和152层ResNets(表1)。值得注意的是，尽管深度显著增加，152层的ResNet(113亿FLOPs)仍然比VGG-16/19 nets(153 / 196亿FLOPs)的复杂度低。

50/101/152层 的ResNets比34层 的ResNets更 准确(表3和4)。没有观察到退化问题，因此可以从大幅增加的深度中获得显著的精度增益。深度的好处体现在所有的评估指标上(表3和4)。

与最新方法的比较。 在表4中，我们将之前最好的单模型结果进行了比较。我们的基线34层ResNets已经达到了非常有竞争力的精度。所提出的152层ResNet的单模型top-5验证误差为4.49%。这个单模型结果优于所有之前的集成结果(表5)。我们将六个不同深度的模型组合起来形成一个集成(在提交时只有两个152层的模型)。这导致测试集上的top-5错误率为**3.57**%(表5)。该作品获得*2015*年*ILSVRC*竞赛第一名。

4.2. CIFAR-10和分析

我们在CIFAR-10数据集[20]上进行了更多的研究，该数据集由10个类别中的50k训练图像和10k测试图像组成。本文提出在训练集上训练的实验，并在测试集上进行评估。我们的重点是极端深度网络的行为，而不是推动最先进的结果，因此我们有意使用以下简单的架构。

普通/残差架构遵循图3(中/右)中的形式。网络输入是32 × 32个图像，减去每个像素的平均值。第一层是3 × 3个卷积。然后我们在大小分别为$\{32, 16, 8\}$的特征图上使用具有3 × 3个卷积的$6n$层堆栈，每个特征图大小有2 n层。过滤器的数量分别是$\{16, 32, 64\}$。下采样由步幅为2的卷积执行。该网络以全局平均池化、10路全连接层和softmax结束。总共有6 n +2个堆叠的加权层。下表总结了该架构。

output map size	32×32	16×16	8×8
# layers	1+2n	2n	2n
# filters	16	32	64

当使用快捷连接时，它们连接到3 × 3层(总共3n快捷方式)。在这个数据集上，我们在所有情况下都使用标识快捷方式(*i.e.*，选项A)，因此我们的残差模型与普通的对应模型具有完全相同的深度、宽度和参数数量。

我们使用0.0001的权重衰减和0.9的动量，并采用[13]和BN [16]中的权重初始化，但没有dropout。这些模型在两个gpu上以128的小批量大小进行训练。

method			error (%)
Maxout [10]			9.38
NIN [25]			8.81
DSN [24]			8.22
	# layers	# params	
FitNet [35]	19	2.5M	8.39
Highway [42, 43]	19	2.3M	7.54 (7.72±0.16)
Highway [42, 43]	32	1.25M	8.80
ResNet	20	0.27M	8.75
ResNet	32	0.46M	7.51
ResNet	44	0.66M	7.17
ResNet	56	0.85M	6.97
ResNet	110	1.7M	**6.43** (6.61±0.16)
ResNet	1202	19.4M	7.93

Table 6. **CIFAR-10**测试集上的分类错误。所有方法都带有数据增强。对于ResNet-110，我们运行5次并显示"best (mean ± std)"，例如[43]。

我们从学习率0.1开始，在32k和48k次迭代时除以10，在64k次迭代时终止训练，这是根据45k/5k的train/val划分确定的。我们遵循[24]中的简单数据增强进行训练:每一侧填充4个像素，并从填充图像或其水平翻转中随机采样32 × 32裁剪。为了测试，我们只评估原始32 × 32图像的单视图。

我们比较了$n = \{3, 5, 7, 9\}$，得出20层、32层、44层和56层网络。图6(左)显示了普通网的行为。深度普通网络的深度增加，并在越深时表现出更高的训练误差。这种现象类似于ImageNet(图4，左)和MNIST(见[42])上的现象，表明这种优化难度是一个基本问题。

图6(中)显示了ResNets的行为。同样类似于ImageNet的情况(图4，右)，我们的ResNets成功克服了优化困难，并在深度增加时显示出精度的提高。

我们进一步探索$n = 18$，从而得到110层的ResNet。在这种情况下，我们发现初始学习率0.1有点大，无法开始收敛[5]。因此，我们使用0.01来热身训练，直到训练误差低于80%(大约400次迭代)，然后回到0.1并继续训练。其余的学习计划和之前一样。这个110层的网络很好地收敛(图6，中间)。它比其他深度和薄网络(如FitNet [35]和Highway [42])的参数更少(表6)，但它是最先进的结果之一(6.43%，表6)。

层响应分析。 图7表示层响应的标准差(std)。响应是每个3 × 3层的输出，在BN之后和其他非线性(ReLU/加法)之前。对于ResNets，该分析揭示了残差函数的响应强度。图. 7表明，ResNets的响应通常比普通网络的响应小。这些结果支持我们的基本动机(3.1节)，即残差函数可能比非残差函数通常更接近于零。我们还注意到，越深的ResNet，响应幅度越小，这一点可以从

[5]当初始学习率为0.1时，它在几个epoch后开始收敛(< 90%误差)，但仍然达到类似的精度。

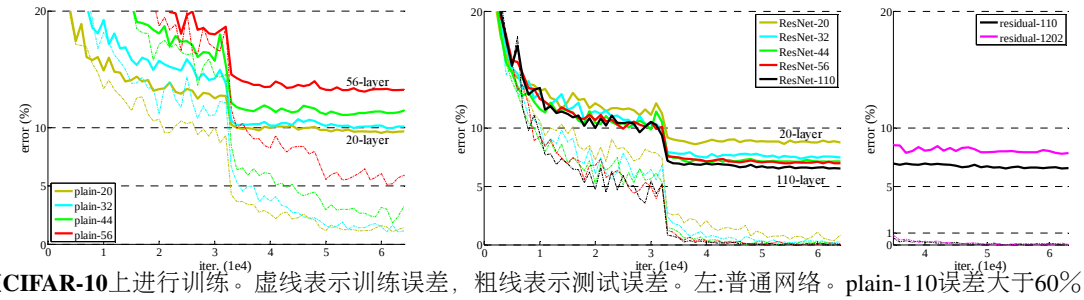

Figure 6. 在**CIFAR-10**上进行训练。虚线表示训练误差，粗线表示测试误差。左:普通网络。plain-110误差大于60%，不显示。中间:ResNets。右:具有110层和1202层的ResNets。

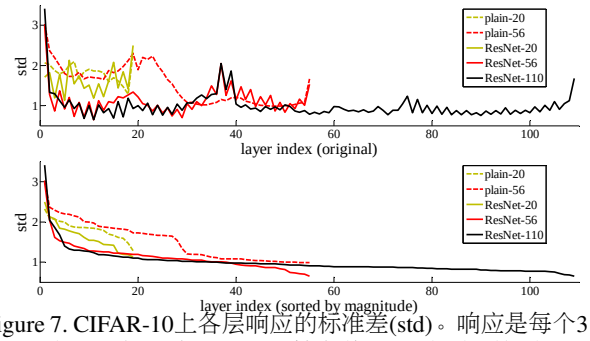

Figure 7. CIFAR-10上各层响应的标准差(std)。响应是每个3 × 3层的输出，在BN之后和非线性之前。上图:图层按原始顺序显示。下:响应按降序排列。

training data	07+12	07++12
test data	VOC 07 test	VOC 12 test
VGG-16	73.2	70.4
ResNet-101	**76.4**	**73.8**

Table 7. 使用基线Faster R-CNN在PASCAL VOC 2007/2012测试集上的目标检测mAP(%)。请参阅10和11以获得更好的结果。

metric	mAP@.5	mAP@[.5, .95]
VGG-16	41.5	21.2
ResNet-101	**48.4**	**27.2**

Table 8. 使用基线Faster R-CNN在COCO验证集上的目标检测图(%)。更好的结果参见9。

图7中ResNet-20、56和110之间的比较中得到证明。当层数更多时，ResNets的单个层倾向于对信号的修改更少。

探索超过**1000**层。我们探索了一个超过1000层的深度模型。我们将$n = 200$设置为1202层的网络，该网络按照上述方法进行训练。我们的方法没有优化难度，这个10^3层的网络能够达到训练误差$< 0.1\%$(图6，右)。它的测试误差仍然相当好(7.93%，表6)。

但在如此深度的模型上仍有一些开放性问题。这个1202层的网络的测试结果比我们的110层的网络差，尽管两者的训练误差相似。我们认为这是因为过度拟合。1202层的网络对于这个小数据集来说可能太大了(19.4M)。强正则化，如maxout [10]或dropout [14]，应用于此数据集获得最佳结果([10, 25, 24, 35])。本文没有使用maxout/dropout，只是简单地通过设计的深层和稀薄的体系结构施加正则化，而不会分散对优化困难的关注。但是结合更强的正则化可能会改善结果，我们将在未来研究。

4.3. PASCAL和MS COCO上的目标检测

该方法在其他识别任务上具有良好的泛化性能。表7和8显示了PASCAL VOC 2007和2012 [5]和COCO [26]上的目标检测基线结果。采用*Faster R-CNN*[32]作为检测方法。在这里，我们对用ResNet-101替换VGG-16 [41]的改进感兴趣。使用两个模型的检测实现(见附录)是相同的，因此收益只能归因于更好的网络。最值得注意的是，在具有挑战性的COCO数据集上，我们获得了6.0%的COCO的标准指标(mAP@[。5.95])，相对改善了28%。这种增益完全是由于学习到的表示。

基于深度残差网络，我们在ILSVRC &COCO 2015竞赛中获得了多个赛道的第一名:ImageNet检测，ImageNet定位，COCO检测和COCO分割。详情见附录。

References

[1] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[2] C. M. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[3] W. L. Briggs, S. F. McCormick, et al. *A Multigrid Tutorial*. Siam, 2000.

[4] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011.

[5] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, pages 303–338, 2010.

[6] S. Gidaris and N. Komodakis. Object detection via a multi-region & semantic segmentation-aware cnn model. In *ICCV*, 2015.

[7] R. Girshick. Fast R-CNN. In *ICCV*, 2015.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[10] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv:1302.4389*, 2013.

[11] K. He and J. Sun. Convolutional neural networks at constrained time cost. In *CVPR*, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[17] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *TPAMI*, 33, 2011.

[18] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *TPAMI*, 2012.

[19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014.

[20] A. Krizhevsky. Learning multiple layers of features from tiny images. *Tech Report*, 2009.

[21] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[22] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

[23] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, pages 9–50. Springer, 1998.

[24] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. *arXiv:1409.5185*, 2014.

[25] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv:1312.4400*, 2013.

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*. 2014.

[27] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[28] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *NIPS*, 2014.

[29] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[30] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

[31] T. Raiko, H. Valpola, and Y. LeCun. Deep learning made easier by linear transformations in perceptrons. In *AISTATS*, 2012.

[32] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[33] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. *arXiv:1504.06066*, 2015.

[34] B. D. Ripley. *Pattern recognition and neural networks*. Cambridge university press, 1996.

[35] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *arXiv:1409.0575*, 2014.

[37] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120*, 2013.

[38] N. N. Schraudolph. Accelerated gradient descent by factor-centering decomposition. Technical report, 1998.

[39] N. N. Schraudolph. Centering neural network gradient factors. In *Neural Networks: Tricks of the Trade*, pages 207–226. Springer, 1998.

[40] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

[41] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[42] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv:1505.00387*, 2015.

[43] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. *1507.06228*, 2015.

[44] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[45] R. Szeliski. Fast surface interpolation using hierarchical basis functions. *TPAMI*, 1990.

[46] R. Szeliski. Locally adapted hierarchical basis preconditioning. In *SIGGRAPH*, 2006.

[47] T. Vatanen, T. Raiko, H. Valpola, and Y. LeCun. Pushing stochastic gradient towards second-order methods–backpropagation learning with transformations in nonlinearities. In *Neural Information Processing*, 2013.

[48] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008.

[49] W. Venables and B. Ripley. Modern applied statistics with s-plus. 1999.

[50] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In *ECCV*, 2014.

A. 目标检测基线

在本节中，我们将介绍基于基线Faster R-CNN [32]系统的检测方法。模型由ImageNet分类模型初始化，然后在目标检测数据上进行微调。在ILSVRC &COCO 2015检测比赛时，我们使用ResNet-50/101进行了实验。

与[32]中使用的VGG-16不同，我们的ResNet没有隐藏的fc层。本文采用"Conv特征映射网络"(NoC) [33]的想法来解决这个问题。我们使用那些在图像上步幅不大于16像素的层来计算全图像共享的conv特征图(*i.e.*，conv1，conv2__x，conv3__x和conv4__x，在ResNet-101中共有91个conv层;表1)。我们认为这些层类似于VGG-16中的13个conv层，通过这样做，ResNet和VGG-16都具有相同的总步幅(16像素)的conv特征图。这些层由一个区域候选网络(RPN，生成300个候选框)[32]和一个快速R-CNN检测网络[7]共享。RoI池化[7]在conv5__1之前执行。在这个roi池化特性上，每个region采用conv5__x及以上所有层，起到VGG-16的fc层的作用。最后的分类层由两个兄弟层(classification和box regression [7])代替。

为了使用BN层，在预训练后，我们计算ImageNet训练集上每一层的BN统计量(均值和方差)。然后在微调过程中固定BN层以进行目标检测。因此，BN层成为具有恒定偏移量和尺度的线性激活，并且BN统计不会通过微调来更新。我们修复BN层主要是为了减少Faster R-CNN训练中的内存消耗。

PASCAL voc

在[7, 32]之后，对于PASCAL VOC 2007 测试集，我们使用VOC 2007中的5k 训练图像和VOC 2012中的16k训练图像进行训练(' 07+12 ')。对于PASCAL VOC 2012测试集，我们使用VOC 2007中的10k 训练 + 测试图像和VOC 2012中的16k 训练图像进行训练(" 07++12")。用于训练Faster R-CNN的超参数与[32]相同。表7显示了结果。ResNet-101比VGG-16提高了$> 3\%$。这种增益完全是由于ResNet学习到的改进特征。

可可女士

MS COCO数据集[26]涉及80个对象类别。我们评估了PASCAL VOC指标(mAP @ IoU = 0.5)和标准COCO指标(mAP @ IoU = 0.5: 0.05: 0.95)。我们使用训练集上的80k张图像进行训练，val集上的40k张图像进行评估。我们对COCO的检测系统与PASCAL VOC的检测系统类似。我们使用8-GPU实现来训练COCO模型，因此RPN步骤具有8张图像的小批量大小(*i.e.*，每个GPU 1张)，Fast R-CNN步骤具有16张图像的小批量大小。RPN步骤和Fast R-CNN步骤都以学习率0.001训练240k次迭代，然后以0.0001训练80k次迭代。

表8显示了MS COCO验证集上的结果。ResNet-101的mAP@[增长了6%。与VGG-16相比，相对提升了28%，这完全是由更好的网络学习到的特征所贡献。值得注意的是，地图@[。5.95]的绝对增长率(6.0%)几乎与mAP@.一样大5分(6.9%)。这表明，更深的网络可以提高识别和定位。

B. 目标检测的改进

为了完整起见，我们报告为比赛所做的改进。这些改进是基于深度特征的，因此应该受益于残差学习。

可可女士

盒子细化。我们的盒子改进部分遵循[6]中的迭代定位。在Faster R-CNN中，最终输出是一个与建议框不同的回归框。因此，对于推理，我们从回归框中汇集一个新特征，并获得一个新的分类分数和一个新的回归框。我们将这300个新预测与原始的300个预测相结合。使用IoU阈值0.3 [8]对预测框的并集应用非极大值抑制(NMS)，然后是框投票[6]。Box的细化使mAP提高了2个点(表9)。

全局上下文。我们在Fast R-CNN步骤中结合全局上下文。给定全图像转换特征图，我们通过全局空间金字塔池化[12](具有"单层"金字塔)来池化一个特征，该特征可以实现为"RoI"池化，使用整个图像的边界框作为RoI。这种池化特征被输入到后roi层以获得全局上下文特征。这个全局特征与原始的逐区域特征连接在一起，然后是兄弟分类层和框回归层。这个新结构是端到端的训练。全局上下文改进mAP@.约1点(表9)。

多尺度测试。在上面，所有结果都是通过单一尺度的训练/测试得到的[32]，其中图像的较短的边是$s = 600$像素。多尺度训练/测试已在[12, 7]中通过从特征金字塔中选择尺度进行开发，在[33]中通过使用maxout层进行开发。在我们当前的实现中，我们执行了以下多尺度测试[33];由于时间有限，我们没有进行多尺度训练。此外，我们只对Fast R-CNN步骤进行了多尺度测试(但还没有对RPN步骤进行测试)。使用训练好的模型，我们在图像金字塔上计算conv特征映射，其中图像的较短边为$s \in \{200, 400, 600, 800, 1000\}$。我们从金字塔中选择了两个相邻的尺度[33]。RoI池化和后续层在这两个尺度的特征图上执行[33]，它们由maxout合并，如[33]。多尺度测试将mAP提高了2个点以上(表9)。

使用验证数据。接下来，我们使用80k+40k训练集进行训练，20k测试-开发集进行评估。测试开发集没有公开可用的真实值，结果由评估服务器报告。在这种设置下，结果是mAP@.55.7%的5票和一张地图@[。5，.95] 34.9%(表9)。这是我们的单模型结果。

合奏。在Faster R-CNN中，系统旨在学习建议区域和目标分类器，因此可以使用集成来促进这两项任务。我们使用一个集成来建议区域，建议的联合集由每个区域的分类器集成来处理。表9显示了我们基于3个网络集成的结果。测试开发集上的mAP分别为59.0%和37.4%。该结果在*COCO 2015*检测任务中获得第一名。

PASCAL voc

我们基于上述模型重新访问PASCAL VOC数据集。使用COCO数据集上的单个模型(55.7%mAP@.5在表9中)，我们在PASCAL VOC集上对该模型进行微

training data	COCO train		COCO trainval	
test data	COCO val		COCO test-dev	
mAP	@.5	@[.5, .95]	@.5	@[.5, .95]
baseline Faster R-CNN (VGG-16)	41.5	21.2		
baseline Faster R-CNN (ResNet-101)	48.4	27.2		
+box refinement	49.9	29.9		
+context	51.1	30.0	53.3	32.2
+multi-scale testing	53.8	32.5	**55.7**	**34.9**
ensemble			**59.0**	**37.4**

Table 9. 使用Faster R-CNN和ResNet-101对MS COCO的目标检测进行改进。

system	net	data	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
baseline	VGG-16	07+12	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
baseline	ResNet-101	07+12	76.4	79.8	80.7	76.2	68.3	55.9	85.1	85.3	**89.8**	56.7	87.8	69.4	88.3	88.9	80.9	78.4	41.7	78.6	79.8	85.3	72.0
baseline+++	ResNet-101	COCO+07+12	**85.6**	**90.0**	**89.6**	**87.8**	**80.8**	**76.1**	**89.9**	**89.9**	89.6	**75.5**	**90.0**	**80.7**	**89.6**	**90.3**	**89.1**	**88.7**	**65.4**	**88.1**	**85.6**	**89.0**	**86.8**

Table 10. 在PASCAL VOC 2007测试集上的检测结果。基线是更快的R-CNN系统。系统"基线+++"包括框细化、上下文和多尺度测试，如表9。

system	net	data	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
baseline	VGG-16	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
baseline	ResNet-101	07++12	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
baseline+++	ResNet-101	COCO+07++12	**83.8**	**92.1**	**88.4**	**84.8**	**75.9**	**71.4**	**86.3**	**87.8**	**94.2**	**66.8**	**89.4**	**69.2**	**93.9**	**91.9**	**90.9**	**89.6**	**67.9**	**88.2**	**76.8**	**90.3**	**80.0**

Table 11. PASCAL VOC 2012测试集(http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=4)上的检测结果。基线是更快的R-CNN系统。系统"基线+++"包括框细化、上下文和多尺度测试，如表9。

	val2	test
GoogLeNet [44] (ILSVRC'14)	-	43.9
our single model (ILSVRC'15)	60.5	58.8
our ensemble (ILSVRC'15)	**63.6**	**62.1**

Table 12. 我们在ImageNet检测数据集上的结果(mAP，%)。我们的检测系统是Faster R-CNN [32]使用ResNet-101对表9进行了改进。

调。采用了盒细化、上下文和多尺度测试等改进方法。通过这样做，我们在PASCAL VOC 2007上实现了85.6%的mAP(表10)，在PASCAL VOC 2012上实现了83.8%(表11) [6]。PASCAL VOC 2012的结果比之前的最新结果[6]高了10分。

ImageNet检测

ImageNet检测(DET)任务涉及200个对象类别。通过mAP@.5对准确率进行了评估。我们对ImageNet DET的目标检测算法与表9中的MS COCO相同。这些网络在1000类的ImageNet分类集上进行预训练，并在DET数据上进行微调。我们在[8]之后将验证集分成两部分(val1/val2)。我们使用DET训练集和val1集对检测模型进行微调。val2集用于验证。我们不使用其他ILSVRC 2015数据。我们使用ResNet-101的单个模型在DET测试集上有58.8%的mAP，我们的3个模型

[6] http://host.robots.ox.ac.uk:8080/anonymous/3OJ4OJ.html，提交日期:2015-11-26。

LOC method	LOC network	testing	LOC error on GT CLS	classification network	top-5 LOC error on predicted CLS
VGG's [41]	VGG-16	1-crop	33.1 [41]		
RPN	ResNet-101	1-crop	13.3		
RPN	ResNet-101	dense	11.7		
RPN	ResNet-101	dense		ResNet-101	14.4
RPN+RCNN	ResNet-101	dense		ResNet-101	**10.6**
RPN+RCNN	ensemble	dense		ensemble	**8.9**

Table 13. ImageNet验证上的定位误差(%)。在" GT类上的LOC error ' ([41])这一列中，使用了ground truth类。在"testing"列中，"1-crop '表示对224 × 224像素的中心裁剪进行测试，"dense '表示密集(全卷积)和多尺度测试。

的集成在DET测试集上有62.1%的mAP(表12)。该结果在*ILSVRC 2015*的*ImageNet*检测任务中获得第一名，超过第二名**8.5**分(绝对)。

C. ImageNet定位

ImageNet定位(LOC)任务[36]需要对对象进行分类和定位。在[40, 41]之后，我们假设首先采用图像级分类器来预测图像的类标签，并且定位算法只考虑基于预测的类来预测边界框。我们采用"逐类回归"(PCR)策略[40, 41]，为每个类学习一个边界框回归器。为ImageNet分类预训练网络，然后对其进行微调以进行定位。在提供的1000类ImageNet训练集上训练网络。

method	top-5 localization err	
	val	test
OverFeat [40] (ILSVRC'13)	30.0	29.9
GoogLeNet [44] (ILSVRC'14)	-	26.7
VGG [41] (ILSVRC'14)	26.9	25.3
ours (ILSVRC'15)	**8.9**	**9.0**

Table 14. 在ImageNet数据集上与最先进方法的定位误差(%)比较。

我们的定位算法基于[32]的RPN框架，并进行了少量修改。与[32]中与类别无关的方式不同，我们用于本地化的RPN是按类设计的。这个RPN以两个兄弟1 × 1卷积层结束，用于二分类*(cls)*和框回归*(reg)*，如[32]。与[32]相反，*cls*和*reg*层都在*per-class* from中。具体来说，*cls*层有1000-d输出，每个维度是二元逻辑回归，用于预测是否为目标类;*reg*层有1000 × 4-d输出，由1000类的框回归器组成。正如在[32]中，我们的边界框回归是参考每个位置的多个平移不变的"锚"框。

在我们的ImageNet分类训练(3.4部分)中，我们随机采样224 × 224种作物用于数据增强。我们使用256张图像的小批量大小进行微调。为了避免负样本占主导地位，对每个图像随机采样8个锚点，其中采样的正锚点和负锚点的比例为1:1 [32]。为了测试，该网络完全卷积地应用在图像上。

表13比较了本地化结果。在[41]之后，我们首先使用ground truth类作为分类预测执行'oracle '测试。VGG的论文[41]报告了使用真实类的中心裁剪误差为33.1%(表13)。在相同的设置下，我们使用ResNet-101网络的RPN方法将中心裁剪误差显著降低到13.3%。通过比较，证明了该框架的卓越性能。通过密集(全卷积)和多尺度测试，我们的ResNet-101在使用真实类别时的误差为11.7%。使用ResNet-101进行分类预测(4.6%的top-5分类误差，表4)，top-5定位误差为14.4%。

以上结果仅基于Faster R-CNN [32]中的建议网络(RPN)。人们可以使用Faster R-CNN中的检测网络(Fast R-CNN [7])来改善结果。但我们注意到，在这个数据集上，一幅图像通常包含一个主目标，并且建议区域彼此高度重叠，因此具有非常相似的roi池化特征。因此，Fast R-CNN [7]的以图像为中心的训练生成了小变化的样本，这可能不是随机训练所希望的。受此启发，在我们当前的实验中，我们使用以roi为中心的原始R-CNN [8]代替Fast R-CNN。

我们的R-CNN实现如下。我们将上述训练的每个类的RPN应用于训练图像，以预测地面真值类的边界框。这些预测框发挥了类依赖建议的作用。对于每个训练图像，提取得分最高的200个候选框作为训练样本来训练R-CNN分类器。图像区域从建议框中裁剪，变形为224 × 224像素，并输入到分类网络中，如R-CNN [8]。该网络的输出由*cls*和*reg*的两个兄弟fc层组成，也是按类形式。这个R-CNN网络在训练集上以roi为中心，使用256的小批量大小进行微调。为了测试，RPN为每个预测类别生成得分最高的200个候选框，R-CNN网络用于更新这些候选框的得分和框位置。

这种方法将top-5定位误差降低到10.6%(表13)。这是我们在验证集上的单模型结果。使用分类和定位的网络集合，在测试集上实现了9.0%的top-5定位误差。这个数字显著优于ILSVRC 14结果(表14)，显示了64%的相对误差减少。该结果在*ILSVRC 2015*的*ImageNet*定位任务中获得第一名。