附表 2-1（通用）

南京邮电大学

毕业设计(论文)外文资料翻译

学　　院	计算机学院、软件学院、网络空间安全学院
专　　业	数据科学与大数据技术
学生姓名	李郅
班级学号	B21032121
外文出处	Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

附件：1. 外文资料翻译译文；2. 外文原文

指导教师评价：

1．翻译内容与课题的结合度：□ 优　□ 良　□ 中　□ 差

2．翻译内容的准确、流畅：□ 优　□ 良　□ 中　□ 差

3．专业词汇翻译的准确性：□ 优　□ 良　□ 中　□ 差

4．翻译字符数是否符合规定要求：□ 符合　□ 不符合

指导教师签名：

2022 年 5 月 13 日

将深度残差学习用于图像识别

何恺明、张翔宇、任少青、孙剑
微软研究院
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

摘要——更深的神经网络更难训练。我们提出了一个残差学习框架，以简化比以往更深网络的训练。我们明确地将层重新定义为学习残差函数，而不是学习无参考函数。我们提供了全面的实证证据，表明这些残差网络更容易优化，并且可以从显著增加的深度中获得更高的精度。在 ImageNet 数据集上，我们评估了深度达 152 层的残差网络——比 VGG 网络[40]深 8 倍，但复杂度仍然较低。这些残差网络的集成在 ImageNet 测试集上达到了 3.57%的错误率。这一结果在 ILSVRC 2015 分类任务中获得了第一名。我们还在 CIFAR-10 上进行了分析，深度为 100 层和 1000 层。表示的深度对于许多视觉识别任务至关重要。仅仅由于我们极其深的表示，我们在 COCO 对象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给 ILSVRC & COCO 2015 竞赛的基础，在这些竞赛中，我们还在 ImageNet 检测、ImageNet 定位、COCO 检测和 COCO 分割任务中获得了第一名。这强有力的证据表明残差学习原理是通用的，我们期望它适用于其他视觉和非视觉问题。

1. 介绍

深度卷积神经网络[22, 21]在图像分类[21, 49, 39]方面取得了一系列突破。深度网络自然地将低/中/高级特征[49]和分类器集成在一个端到端的多层结构中，并且通过堆叠层数（深度）可以丰富特征的“层次”。最近的证据[40, 43]表明，网络深度至关重要，在具有挑战性的 ImageNet 数据集[35]上的领先结果[40, 43, 12, 16]都利用了“非常深”的[40]模型，深度为 16[40]到 30[16]。许多其他非平凡的视觉识别任务[7, 11, 6, 32, 27]也从非常深的模型中受益匪浅。

由于深度的重要性，一个问题出现了：学习更好的网络是否像堆叠更多层一样容易？回答这个问题的一个障碍是著名的梯度消失/爆炸问题[14, 1, 8]，它从一开始就阻碍了收敛。然而，这个问题在很大程度上通过归一化初始化[23, 8, 36, 12]和中间归一化层[16]得到了解决，这使得具有数十层的网络能够通过反向传播[22]的随机梯度下降（SGD）开始收敛。

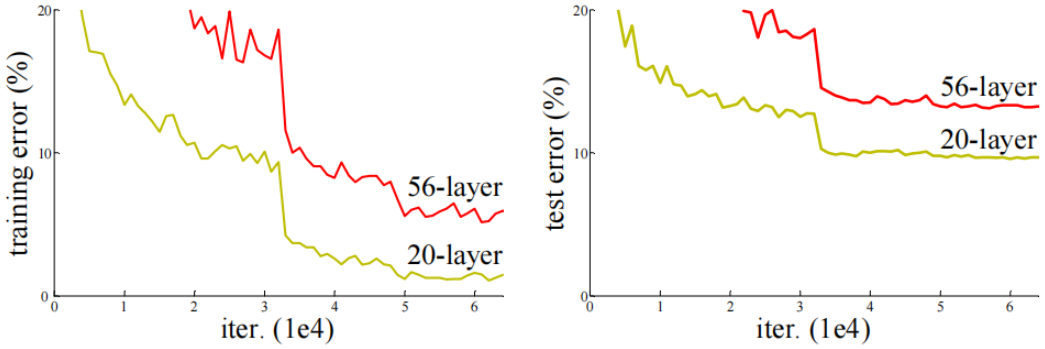

图 1. 在具有 20 层和 56 层“普通”网络的 CIFAR-10 上训练错误（左）和测试错误（右）。更深的网络具有更高的训练误差，因此测试误差也更大。ImageNet 上的类似现象如图 4 所示。

当更深的网络能够开始收敛时，暴露出一个退化问题：随着网络深度的增加，准确性会饱和（这可能并不令人惊讶），然后迅速下降。出乎意料的是，这种退化不是由过拟合引起的，向一个适当深度的模型添加更多层会导致更高的训练误差，如[10, 41]所报告，并通过我们的实验彻底验证。图 1 显示了一个典型的例子。

训练准确性的退化表明，并非所有系统都同样容易优化。让我们考虑一个较浅的架构及其更深的对应架构，该架构在其上添加了更多层。通过构造存在一个更深模型的解决方案：添加的层是恒等映射，其他层是从学习到的较浅模型中复制的。这个构造的解决方案的存在表明，更深的模型不应该比其较浅的对应模型产生更高的训练误差。但实验表明，我们当前手头的求解器无法找到与构造的解决方案同样好的解决方案（或无法在可行的时间内找到）。

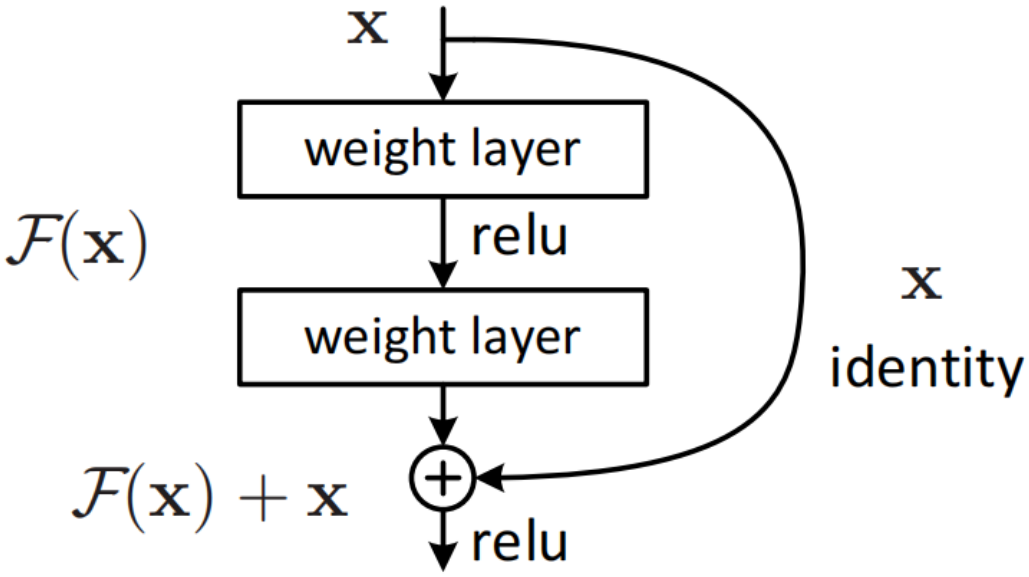

图 2:残差学习：一个建设中的基石。

在本文中，我们通过引入一个深度残差学习框架来解决退化问题。我们明确地让这些层拟合一个残差映射，而不是希望每几个堆叠层直接拟合一个期望的基础映射。形式上，将期望的基础映射表示为 H(x)，我们让堆叠的非线性层拟合另一个映射 F(x) := H(x) - x。原始映射重新表述为 F(x) + x。我们假设优化残差映射比优化原始的无参考映射更容易。极端情况下，如果恒等映射是最优的，那么推动残差为零比通过堆叠的非线性层拟合恒等映射更容易。

F(x) + x 的表述可以通过具有“捷径连接”的前馈神经网络实现（图 2）。捷径连接[2, 33, 48]是那些跳过一个或多个层的连接。在我们的情况下，捷径连接仅执行恒等映射，并且它们的输出被添加到堆叠层的输出中（图 2）。恒等捷径连接既不增加额外的参数也不增加计算复杂度。整个网络仍然可以通过 SGD 和反向传播进行端到端训练，并且可以使用常见的库（例如 Caffe[19]）轻松实现，而无需修改求解器。

我们在 ImageNet[35]上进行了全面的实验，展示了退化问题并评估了我们的方法。我们展示了：1）我们的极深残差网络易于优化，但简单堆叠层的对应“普通”网络在深度增加时表现出更高的训练误差；2）我们的深度残差网络可以轻松地从显著增加的深度中获得准确性收益，产生的结果明显优于以前的网络。

类似的现象也在 CIFAR-10 集[20]上显示，表明优化困难和我们方法的效果不仅仅类似于特定的数据集。我们在这个数据集上展示了成功训练的超过 100 层的模型，并探索了超过 1000 层的模型。

在 ImageNet 分类数据集[35]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 上展示的最深的网络，同时复杂度仍然低于 VGG 网络[40]。我们的集成在 ImageNet 测试集上达到了 3.57%的 top-5 错误率，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具有出色的泛化性能，使我们在 ILSVRC 和 COCO 2015 竞赛中进一步赢得了 ImageNet 检测、ImageNet 定位、COCO 检测和 COCO 分割的第一名。这些强有力的证据表明，残差学习原理是通用的，我们期望它在其他视觉和非视觉问题中也适用。

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 Vector[30]可以被表述为 VLAD 的概率版本[18]。它们都是图像检索和分类[4, 47]的强大浅层表示。对于向量量化，编码残差向量[17]被证明比编码原始向量更有效。

在低级视觉和计算机图形学中，为了解决偏微分方程（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上的子问题，每个子问题负责较粗

和较细尺度之间的残差解。多重网格的替代方法是分层基预条件[44, 45]，它依赖于表示两个尺度之间残差向量的变量。已经表明[3, 44, 45]，这些求解器比不考虑残差性质的标准求解器收敛得更快。这些方法表明，良好的重新表述或预条件可以简化优化。

捷径连接

导致捷径连接[2, 33, 48]的实践和理论已经研究了很长时间。训练多层感知器（MLPs）的早期实践是添加一个从网络输入到输出的线性层[33, 48]。在[43, 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。[38, 37, 31, 46]的论文提出了通过捷径连接实现层响应、梯度和传播误差中心化的方法。在[43]中，“inception”层由一个捷径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”[41, 42]提出了带有门控函数[15]的捷径连接。这些门是数据依赖的，并且具有参数，与我们的恒等捷径不同，后者是无参数的。当门控捷径“关闭”（接近零）时，高速公路网络中的层表示非残差函数。相反，我们的表述总是学习残差函数；我们的恒等捷径永远不会关闭，并且所有信息总是通过，同时还需要学习额外的残差函数。此外，高速公路网络还没有证明在极深（例如，超过 100 层）时具有准确性收益。

3.深度残差学习

3.1 残差学习

让我们考虑 H(x)作为一个由几个堆叠层（不一定是整个网络）拟合的基础映射，x 表示这些层中第一层的输入。如果假设多个非线性层可以渐近地近似复杂函数 2，那么等价于假设它们可以渐近地近似残差函数，即 H(x) − x（假设输入和输出的维度相同）。因此，我们不是期望堆叠层近似 H(x)，而是明确地让这些层近似一个残差函数 F(x) := H(x) − x。原始函数因此变为 F(x) + x。尽管两种形式都应该能够渐近地近似期望的函数（如假设的那样），但学习的难易程度可能不同。

这种重新表述是受到关于退化问题的反直觉现象的启发（图 1，左）。正如我们在引言中讨论的，如果添加的层可以构造为恒等映射，那么更深的模型的训练误差不应该大于其较浅的对应模型。退化问题表明，求解器可能难以通过多个非线性层近似恒等映射。有了残差学习的重新表述，如果恒等映射是最优的，求解器可能只需将多个非线性层的权重推向零以接近恒等映射。

在实际情况中，恒等映射不太可能是最优的，但我们的重新表述可能有助于预条件问题。如果最优函数更接近恒等映射而不是零映射，那么求解器找到相对于恒等映射的扰动应该比将函数作为新的函数学习更容易。我们通过实验（图 7）表明，学习到的残差函数通常具有较小的响应，表明恒等映射提供了合理的预条件。

3.2 通过捷径实现恒等映射

我们将残差学习应用于每几个堆叠层。一个构建块如图 2 所示。形式上，在本文中我们考虑一个定义为：

$$y = F(x, \{W_i\}) + x. \quad (1)$$

这里 x 和 y 是所考虑层的输入和输出向量。函数 F(x, {Wi})表示要学习的残差映射。对于图 2 中的例子，它有两层，$F = W_2\sigma(W_1x)$，其中 σ 表示 ReLU[29]，为了简化符号，省略了偏置。操作 F + x 通过捷径连接和逐元素加法执行。我们在加法后采用第二个非线性（即，σ(y)，见图 2）。

捷径连接在方程（1）中既不引入额外的参数也不增加计算复杂度。这不仅在实践中很有吸引力，而且在我们比较普通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（除了可以忽略的逐元素加法）的普通/残差网络。

在方程（1）中 x 和 F 的维度必须相等。如果情况并非如此（例如，当改变输入/输出通道时），我们可以通过捷径连接执行线性投影 Ws 来匹配维度：

$$y = F(x, \{W_i\}) + W_sx. \quad (2)$$

我们也可以在方程（1）中使用方阵 Ws。但我们将通过实验表明，恒等映射足以解决退化问题，并且是经济的，因此 Ws 仅在匹配维度时使用。

残差函数 F 的形式是灵活的。本文中的实验涉及具有两层或三层的函数 F（图 5），但也可能有更多层。但如果 F 只有一层，方程（1）类似于线性层：$y = W_1x + x$，我们没有观察到这种优势。

我们还注意到，尽管上述符号是关于全连接层的简化，但它们适用于卷积层。函数 F(x, {Wi})可以表示多个卷积层。逐元素加法是在两个特征图上按通道执行的。

3.3 网络架构

我们测试了各种普通/残差网络，并观察到了一致的现象。为了提供讨论的实例，我们描述了两个 ImageNet 模型。

普通网络

我们的普通基线（图 3，中间）主要受到 VGG 网络[40]（图 3，左）的启发。卷积层大多有 3×3 滤波器，并遵循两个简单的设计规则：（i）对于相同输出特征图大小的层，具有相同数量的滤波器；（ii）如果特征图大小减半，则滤波器数量加倍，以保持每层的时间复杂度。我们通过具有 2 步长的卷积层直接执行下采样。网络以全局平均池化层和具有 softmax 的 1000 路全连接层结束。图 3（中间）中的总加权层数为 34。

值得注意的是，我们的模型比 VGG 网络[40]（图 3，左）具有更少的滤波器和更低的复杂度。我们的 34 层基线具有 36 亿次 FLOPs（乘加），仅为 VGG-19（196 亿次 FLOPs）的 18%。

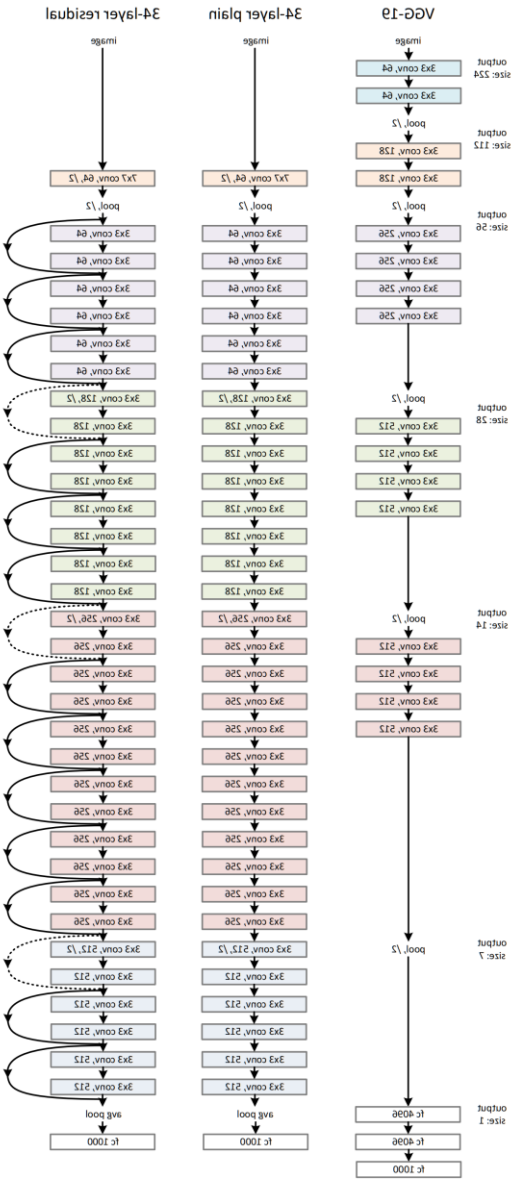

图 3。ImageNet 的网络架构示例。**左：**VGG-19 模型[40]（196 亿 FLOP）作为参考。Mid-dle：一个具有 34 个参数层（36 亿 FLOP）的普通网络。**右：**一个具有 34 个参数层（36 亿 FLOP）的残差网络。虚线快捷键增加了尺寸。表 1 显示了更多细节和其他变体。

残差网络

基于上述普通网络，我们插入捷径连接（图 3，右），将网络转变为对应的残差版本。当输入和输出维度相同时，可以直接使用恒等捷径（方程（1））（图 3 中的实线捷径）。当维度增加时（图 3 中的虚线捷径），我们考虑两种选项：（A）捷径仍然执行恒等映射，对于增加的维度添加额外的零项。此选项不引入

额外参数；（B）使用方程（2）中的投影捷径来匹配维度（通过 1×1 卷积完成）。对于这两种选项，当捷径跨越两个大小的特征图时，它们以 2 的步长执行。

3.4 实现

我们的 ImageNet 实现遵循[21, 40]中的实践。图像的较短边在[256, 480]范围内随机采样进行尺度增强[40]。从图像或其水平翻转中随机采样一个 224×224 的裁剪，并减去每个像素的平均值[21]。使用[21]中的标准颜色增强。我们在每个卷积后和激活前采用批量归一化（BN）[16]，遵循[16]。我们按照[12]中的方式初始化权重，并从头开始训练所有普通/残差网络。我们使用 256 的小批量 SGD。学习率从 0.1 开始，并在错误率平稳时除以 10，并且模型训练最多 60×10^4 次迭代。我们使用 0.0001 的权重衰减和 0.9 的动量。我们不使用 dropout[13]，遵循[16]中的实践。

在测试中，为了比较研究，我们采用标准的 10 裁剪测试[21]。为了获得最佳结果，我们采用[40, 12]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，使得较短边在{224, 256, 384, 480, 640}中）。

4.实验

4.1 ImageNet 分类

我们在 ImageNet 2012 分类数据集[35]上评估我们的方法，该数据集包含 1000 个类别。模型在 128 万张训练图像上进行训练，并在 50k 验证图像上进行评估。我们还在 100k 测试图像上获得了最终结果，由测试服务器报告。我们评估 top-1 和 top-5 错误率。

普通网络。

我们首先评估 18 层和 34 层的普通网络。图 3（中间）显示了 34 层普通网络。18 层普通网络形式类似。表 1 提供了详细的架构。

7 表 2 中的结果表明，更深的 34 层普通网络比更浅的 18 层普通网络具有更高的验证错误率。为了揭示原因，在图 4（左）中，我们比较了它们在训练过程中的训练/验证错误率。我们观察到了退化问题——34 层普通网络在整个训练过程中始终具有更高的训练错误率，尽管 18 层普通网络的解空间是 34 层普通网络解空间的子空间。

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

表 1。ImageNet 的架构。构建块显示在括号中（另见图 5），其中堆叠了块的数量。下采样由 conv3 1、conv4 1 和 conv5 1 执行，步幅为 2。

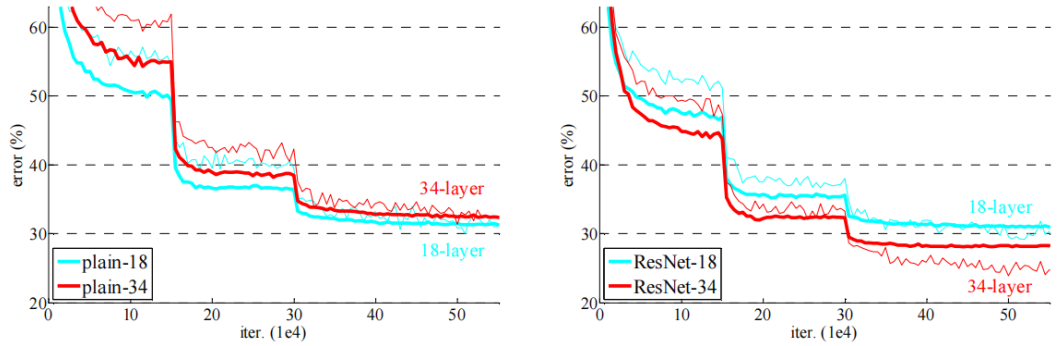

图 4。ImageNet 上的培训。细曲线表示训练误差，粗曲线表示中心作物的验证误差。左：18 层和 34 层的平面网络。右图：18 层和 34 层的 ResNets。在这个图中，与普通网络相比，残差网络没有额外的参数。

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

表 2。ImageNet 验证中的前 1 个错误（%，10 个作物测试）。与普通的对应物相比，这里的 ResNets 没有额外的参数。图 4 显示了培训程序。

我们论证这种优化困难不太可能是由梯度消失引起的。这些普通网络是用 BN[16]训练的，这确保了前向传播信号具有非零方差。我们还验证了反向传播的梯度在 BN 下表现出健康的范数。因此，前向和反向信号都不会消失。事实上，34 层普通网络仍然能够实现有竞争力的准确性（表 3），表明求解器在某种程度上是有效的。我们推测深度普通网络可能具有指数级低的收敛率，这影响了训练错误率的降低。这种优化困难的原因将在未来研究。

残差网络。

接下来我们评估 18 层和 34 层的残差网络（ResNets）。基线架构与上述普通网络相同，只是在每对 3×3 滤波器之间添加了一个捷径连接，如图 3（右）所示。在第一次比较（表 2 和图 4 右）中，我们对所有捷径使用恒等映射，并对增加维度使用零填充（选项 A）。因此，它们与普通对应网络相比没有额外参数。

从表 2 和图 4 中我们可以得出三个主要观察结果。首先，残差学习的情况发生了逆转——34 层 ResNet 优于 18 层 ResNet（提高了 2.8%）。更重要的是，34 层 ResNet 表现出显著较低的训练错误率，并且可以推广到验证数据。这表明退化问题在这种设置中得到了很好的解决，并且我们成功地从增加的深度中获得了准确性收益。

其次，与普通对应网络相比，34 层模型将 top-1 错误率降低了 3.5%（表 2），这是由于成功降低了训练错误率（图 4 右与左）。这一比较验证了残差学习在极深系统中的有效性。

model	top-1 err.	top-5 err.
VGG-16 [40]	28.07	9.33
GoogLeNet [43]	-	9.15
PReLU-net [12]	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**

表 3。ImageNet 验证的错误率（%，10 次作物测试）。VGG-16 基于我们的测试。ResNet-50/101/152 属于选项 B，仅使用投影来增加维度。

method	top-1 err.	top-5 err.
VGG [40] (ILSVRC'14)	-	8.43†
GoogLeNet [43] (ILSVRC'14)	-	7.89
VGG [40] (v5)	24.4	7.1
PReLU-net [12]	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**

表 4。ImageNet 验证集上单个模型结果的错误率（%）（测试集上报告的†除外）。

method	top-5 err. (**test**)
VGG [40] (ILSVRC'14)	7.32
GoogLeNet [43] (ILSVRC'14)	6.66
VGG [40] (v5)	6.8
PReLU-net [12]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

表 5。集合的错误率（%）。前 5 个错误位于 ImageNet 的测试集上，由测试服务器报告。

最后，我们还注意到 18 层普通/残差网络的准确性相当（表 2），但 18 层 ResNet 收敛更快（图 4 右与左）。当网络“不是过深”（这里为 18 层）时，当前的 SGD 求解器仍然能够为普通网络找到好的解决方案。在这种情况下，ResNet 通过在早期阶段提供更快的收敛来简化优化。

恒等与投影捷径。

我们已经表明，无参数的恒等捷径有助于训练。接下来我们研究投影捷径（方程（2））。在表 3 中，我们比较了三种选项：（A）对于增加维度使用零填充捷径，并且所有捷径都是无参数的（与表 2 和图 4 右相同）；（B）对于增加维度使用投影捷径，其他捷径是恒等的；（C）所有捷径都是投影。

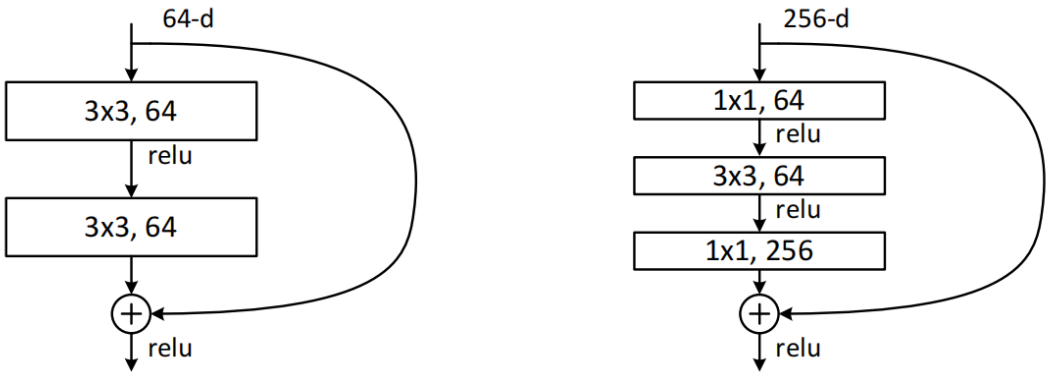

图 5。ImageNet 的更深的残差函数 F。左图：ResNet-34 的构建块（在 56×56 特征图上），如图 3 所示。右：ResNet-50/101/152 的“瓶颈”构建块。

表 3 表明，所有三个选项都比普通对应网络明显更好。B 略优于 A。我们认为这是因为 A 中填充零的维度实际上没有残差学习。C 略优于 B，我们将此归因于许多（十三个）投影捷径引入的额外参数。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非必不可少。因此，我们在本文的其余部分不使用选项 C，以减少内存/时间复杂度和模型大小。对于下面介绍的瓶颈架构，恒等捷径特别重要，因为它不会增加复杂度。

更深的瓶颈架构。

接下来我们描述我们为 ImageNet 设计的更深的网络。由于我们能够承受的训练时间的考虑，我们将构建块修改为瓶颈设计 4。对于每个残差函数 F，我们使用 3 层堆叠而不是 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 层成为一个具有较小输入/输出维度的瓶颈。图 5 显示了一个例子，其中两种设计具有相似的时间复杂度。

对于瓶颈架构，无参数的恒等捷径特别重要。如果图 5（右）中的恒等捷径被投影替换，可以证明时间复杂度和模型大小将翻倍，因为捷径连接到两个高维端。因此，恒等捷径导致瓶颈设计的更高效模型。

50 层 ResNet：

我们将 34 层网络中的每个 2 层块替换为这种 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。这个模型具有 38 亿次 FLOPs。

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 层的 ResNets（表 1）。值得注意的是，尽管深度显著增加，152 层的 ResNet（113 亿次 FLOPs）仍然比 VGG-16/19 网络（153/196 亿次 FLOPs）具有更低的复杂度。

50/101/152 层的 ResNets 比 34 层的网络在相当大的程度上更准确（表 3 和 4）。我们没有观察到退化问题，因此从显著增加的深度中获得了显著的准确性收益。深度的好处在所有评估指标中都有体现（表 3 和 4）。

与最新方法的比较。

在表 4 中，我们与以前最好的单模型结果进行了比较。我们的基线 34 层 ResNets 已经实现了非常有竞争力的准确性。我们的 152 层 ResNet 具有 4.49%的单模型 top-5 验证错误率。这一单模型结果超过了所有以前的集成结果(表 5)。我们将六个不同深度的模型组合成一个集成（提交时只有两个 152 层的）。这导致在测试集上 3.57%的 top-5 错误率（表 5）。这个条目在 ILSVRC 2015 中获得了第一名。

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，该数据集包含 10 个类别的 50k 训练图像和 10k 测试图像。我们在训练集上进行训练并在测试集上进行评估。我们的关注点是极深网络的行为，而不是推动最新结果，因此我们故意使用简单的架构，如下所示。

普通/残差架构遵循图 3（中间/右）的形式。网络输入是 32×32 图像，减去了每个像素的平均值。第一层是 3×3 卷积。然后我们在{32, 16, 8}大小的特征图上使用 6n 层 3×3 卷积的堆叠，每个特征图大小有 2n 层。滤波器的数量分别是{16, 32, 64}。通过步长为 2 的卷积进行下采样。网络以全局平均池化、10 路全连接层和 softmax 结束。总共有 6n+2 层加权层。下表总结了架构：

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

当使用捷径连接时，它们连接到 3×3 层的对（总共 3n 个捷径）。在这个数据集上，我们总是使用恒等捷径（即，选项 A），因此我们的残差模型与普通对应模型具有完全相同的深度、宽度和参数数量。

method			error (%)
Maxout [9]			9.38
NIN [25]			8.81
DSN [24]			8.22
	# layers	# params	
FitNet [34]	19	2.5M	8.39
Highway [41, 42]	19	2.3M	7.54 (7.72±0.16)
Highway [41, 42]	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

表 6。CIFAR-10 测试集上的分类错误。所有的方法都有数据增强。对于 ResNet-110，我们运行 5 次并显示“最佳（均值±标准差）”，如[42]所示。

我们使用 0.0001 的权重衰减和 0.9 的动量，并采用[12]中的权重初始化和 BN[16]，但不使用 dropout。这些模型在两个 GPU 上以 128 的小批量进行训练。我们从 0.1 的学习率开始，在 32k 和 48k 次迭代时将其除以 10，并在 64k 次迭代时终止训练，这是在 45k/5k 训练/验证分割上确定的。我们遵循[24]中的简单数据增强进行训练：在每侧填充 4 个像素，并从填充的图像或其水平翻转中随机采样一个 32×32 的裁剪。对于测试，我们只评估原始 32×32 图像的单视图。

我们比较 n = {3, 5, 7, 9}，导致 20、32、44 和 56 层网络。图 6（左）显示了普通网络的行为。深度普通网络随着深度的增加而遭受困扰，并且在更深时表现出更高的训练错误率。这种现象与 ImageNet（图 4，左）和 MNIST（见[41]）上的现象相似，表明这种优化困难是一个基本问题。

图 6（中）显示了 ResNets 的行为。也类似于 ImageNet 案例（图 4，右），我们的 ResNets 成功地克服了优化困难，并在深度增加时表现出准确性收益。

我们进一步探索了 n = 18，导致一个 110 层的 ResNet。在这种情况下，我们发现初始学习率 0.1 略大，无法开始收敛 5。因此，我们使用 0.01 来预热训练，直到训练错误率低于 80%（大约 400 次迭代），然后回到 0.1 并继续训练。其余的学习计划与之前相同。这个 110 层的网络收敛得很好（图 6，中）。它比

其他深度和薄的网络（如 FitNet[34]和 Highway[41]）具有更少的参数，但处于最新结果之列（6.43%，表 6）。

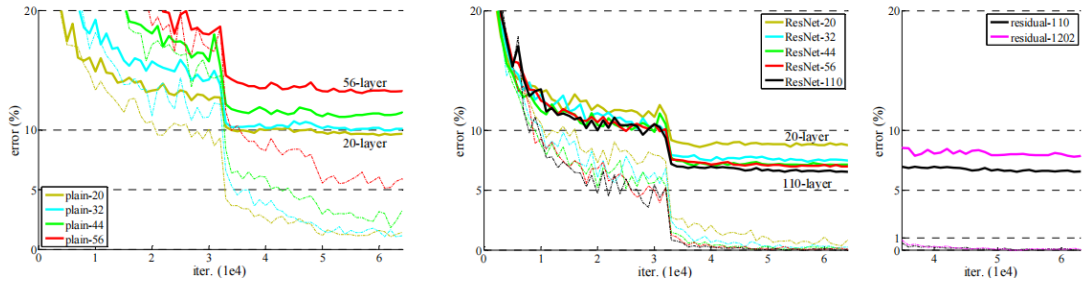

图 6。CIFAR-10 培训。虚线表示训练错误，粗线表示测试错误。左：普通网络。plain-110 的误差高于 60%，未显示。中间：ResNets。右：110 层和 1202 层的 ResNets。

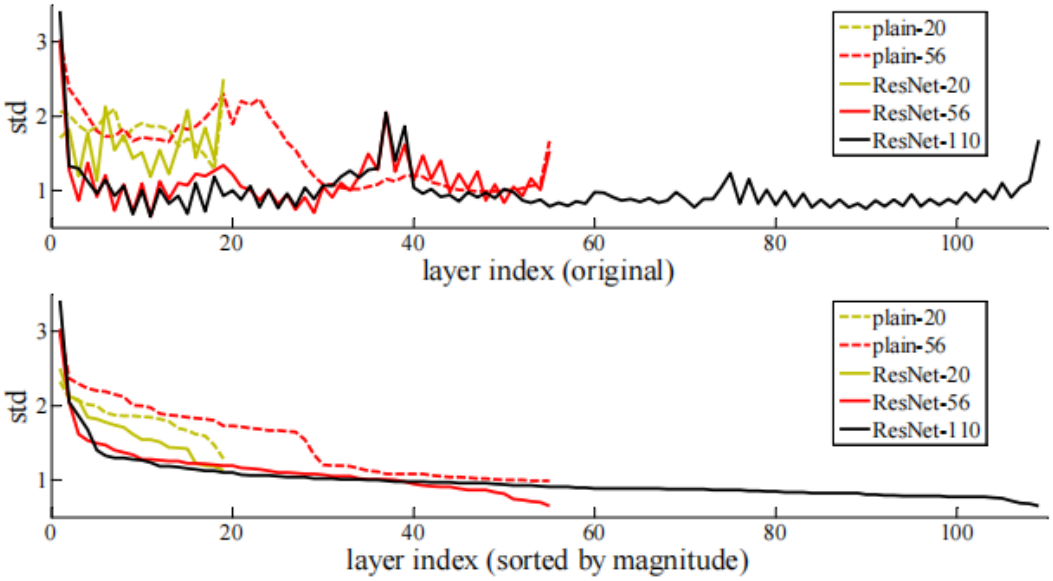

图 7。CIFAR-10 上层响应的标准偏差（std）。响应是 BN 之后和非线性之前每个 3×3 层的输出。顶部：图层按原始顺序显示。底部：答案按降序排列。

层响应分析。

图 7 显示了层响应的标准差（std）。响应是每个 3×3 层的输出，在 BN 之后和非线性（ReLU/addition）之前。对于 ResNets，这一分析揭示了残差函数的响应强度。图 7 表明，ResNets 通常比普通对应网络具有更小的响应。这些结果支持我们基本动机（第 3.1 节），即残差函数可能通常比非残差函数更接近零。我们还注意到，更深的 ResNet 具有更小的响应幅度，如 ResNet-20、56 和 110 之间的比较所示（图 7）。当有更多的层时，ResNet 的单个层倾向于较少地修改信号。

探索超过 1000 层。

我们探索了一个超过 1000 层的极具攻击性的深模型。我们设置 n = 200，导致一个 1202 层的网络，如上所述进行训练。我们的方法没有优化困难，这个

103 层的网络能够实现<0.1%的训练错误率（图 6，右）。其测试错误率仍然相当不错（7.93%，表 6）。

但对于这种极具攻击性的深模型仍然存在开放性问题。这个 1202 层网络的测试结果比我们的 110 层网络差，尽管两者都有类似的训练错误率。我们认为这是由于过拟合。1202 层网络可能对于这个小数据集来说不必要地大（19.4M）。应用了强正则化，如 maxout[9]或 dropout[13]，以获得这个数据集上的最佳结果（[9, 25, 24, 34]）。在本文中，我们没有使用 maxout/dropout，只是简单地通过设计深度和薄架构来施加正则化，而不分散对优化困难的关注。但与更强的正则化结合可能会改善结果，我们将在未来研究。

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**

表 7。使用基线 Faster R-CNN 对 PASCAL VOC 2007/2012 测试集进行对象检测 mAP（%）。更好的结果请参见附录。

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

表 8。使用基线 Faster R-CNN 对 COCO 验证集进行对象检测 mAP（%）。更好的结果请参见附录。

4.3 PASCAL 和 MS COCO 上的目标检测

我们的方法在其他识别任务上具有良好的泛化性能。表 7 和 8 显示了在 PASCAL VOC 2007 和 2012[5]以及 COCO[26]上的目标检测基线结果。我们采用 Faster R-CNN[32]作为检测方法。这里我们感兴趣的是用 ResNet-101 替换 VGG-16[40]所带来的改进。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 标准指标（mAP@[.5, .95]）的增加，这是一个 28%的相对改进。这一收益仅仅是由于学习到的表示。

基于深度残差网络，我们在 ILSVRC 和 COCO 2015 竞赛的几个赛道中获得了第一名：ImageNet 检测、ImageNet 定位、COCO 检测和 COCO 分割。细节在附录中。

参考文献

[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. Zis serman. The Pascal Visual Object Classes (VOC) Challenge. IJCV, pages 303–338, 2010.

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

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

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

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

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

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

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

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

[14] S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. Diploma thesis, TU Munich, 1991.

[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. Muller. 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. Deeplysupervised 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. Dollar, 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. Montufar, 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] B. D. Ripley. Pattern recognition and neural networks. Cambridge university press, 1996.
[34] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. In ICLR, 2015.

[35] 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.
[36] 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.
[37] N. N. Schraudolph. Accelerated gradient descent by factor-centering decomposition. Technical report, 1998.
[38] N. N. Schraudolph. Centering neural network gradient factors. In Neural Networks: Tricks of the Trade, pages 207–226. Springer, 1998.
[39] 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.
[40] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.
[41] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. arXiv:1505.00387, 2015.
[42] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. 1507.06228, 2015.
[43] 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.
[44] R. Szeliski. Fast surface interpolation using hierarchical basis functions. TPAMI, 1990.
[45] R. Szeliski. Locally adapted hierarchical basis preconditioning. In SIGGRAPH, 2006.
[46] 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.
[47] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008.
[48] W. Venables and B. Ripley. Modern applied statistics wi1999.
[49] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In ECCV, 2014.

Deep Residual Learning for Image Recognition

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

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [40] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test *set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.*

The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions[1]*, where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.*

1. Introduction

Deep convolutional neural networks [22, 21] have led to a series of breakthroughs for image classification [21, 49, 39]. Deep networks naturally integrate low/mid/high-level features [49] and classifiers in an end-to-end multi-layer fashion, and the "levels" of features can be enriched by the number of stacked layers (depth). Recent evidence [40, 43] reveals that network depth is of crucial importance, and the leading results [40, 43, 12, 16] on the challenging ImageNet dataset [35] all exploit "very deep" [40] models, with a depth of sixteen [40] to thirty [16]. Many other non-trivial visual recognition tasks [7, 11, 6, 32, 27] have also greatly benefited from very deep models.

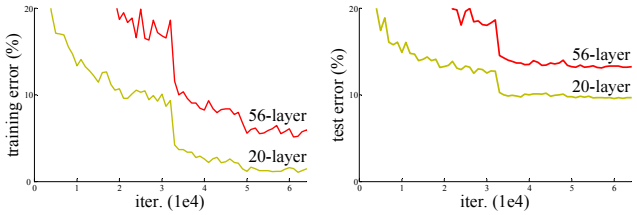

Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Driven by the significance of depth, a question arises: *Is learning better networks as easy as stacking more layers?* An obstacle to answering this question was the notorious problem of vanishing/exploding gradients [14, 1, 8], which hamper convergence from the beginning. This problem, however, has been largely addressed by normalized initialization [23, 8, 36, 12] and intermediate normalization layers [16], which enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with back-propagation [22].

When deeper networks are able to start converging, a *degradation* problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Unexpectedly, such degradation is *not caused by overfitting*, and adding more layers to a suitably deep model leads to *higher training error*, as reported in [10, 41] and thoroughly verified by our experiments. Fig. 1 shows a typical example.

The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize. Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it. There exists a solution *by construction* to the deeper model: the added layers are *identity* mapping, and the other layers are copied from the learned shallower model. The existence of this constructed solution indicates that a deeper model should produce no higher training error than its shallower counterpart. But experiments show that our current solvers on hand are unable to find solutions that

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

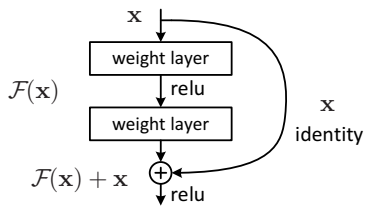

Figure 2. Residual learning: a building block.

are comparably good or better than the constructed solution (or unable to do so in feasible time).

In this paper, we address the degradation problem by introducing a *deep residual learning* framework. Instead of hoping each few stacked layers directly fit a desired underlying mapping, we explicitly let these layers fit a residual mapping. Formally, denoting the desired underlying mapping as $\mathcal{H}(\mathbf{x})$, we let the stacked nonlinear layers fit another mapping of $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$. The original mapping is recast into $\mathcal{F}(\mathbf{x})+\mathbf{x}$. We hypothesize that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. To the extreme, if an identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers.

The formulation of $\mathcal{F}(\mathbf{x})+\mathbf{x}$ can be realized by feedforward neural networks with "shortcut connections" (Fig. 2). Shortcut connections [2, 33, 48] are those skipping one or more layers. In our case, the shortcut connections simply perform *identity* mapping, and their outputs are added to the outputs of the stacked layers (Fig. 2). Identity shortcut connections add neither extra parameter nor computational complexity. The entire network can still be trained end-to-end by SGD with backpropagation, and can be easily implemented using common libraries (*e.g.*, Caffe [19]) without modifying the solvers.

We present comprehensive experiments on ImageNet [35] to show the degradation problem and evaluate our method. We show that: 1) Our extremely deep residual nets are easy to optimize, but the counterpart "plain" nets (that simply stack layers) exhibit higher training error when the depth increases; 2) Our deep residual nets can easily enjoy accuracy gains from greatly increased depth, producing results substantially better than previous networks.

Similar phenomena are also shown on the CIFAR-10 set [20], suggesting that the optimization difficulties and the effects of our method are not just akin to a particular dataset. We present successfully trained models on this dataset with over 100 layers, and explore models with over 1000 layers.

On the ImageNet classification dataset [35], we obtain excellent results by extremely deep residual nets. Our 152-layer residual net is the deepest network ever presented on ImageNet, while still having lower complexity than VGG nets [40]. Our ensemble has **3.57%** top-5 error on the ImageNet *test* set, and *won the 1st place in the ILSVRC 2015 classification competition*. The extremely deep representations also have excellent generalization performance on other recognition tasks, and lead us to further *win the 1st places on: ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation* in ILSVRC & COCO 2015 competitions. This strong evidence shows that the residual learning principle is generic, and we expect that it is applicable in other vision and non-vision problems.

2. Related Work

Residual Representations. In image recognition, VLAD [18] is a representation that encodes by the residual vectors with respect to a dictionary, and Fisher Vector [30] can be formulated as a probabilistic version [18] of VLAD. Both of them are powerful shallow representations for image retrieval and classification [4, 47]. For vector quantization, encoding residual vectors [17] is shown to be more effective than encoding original vectors.

In low-level vision and computer graphics, for solving Partial Differential Equations (PDEs), the widely used Multigrid method [3] reformulates the system as subproblems at multiple scales, where each subproblem is responsible for the residual solution between a coarser and a finer scale. An alternative to Multigrid is hierarchical basis preconditioning [44, 45], which relies on variables that represent residual vectors between two scales. It has been shown [3, 44, 45] that these solvers converge much faster than standard solvers that are unaware of the residual nature of the solutions. These methods suggest that a good reformulation or preconditioning can simplify the optimization.

Shortcut Connections. Practices and theories that lead to shortcut connections [2, 33, 48] have been studied for a long time. An early practice of training multi-layer perceptrons (MLPs) is to add a linear layer connected from the network input to the output [33, 48]. In [43, 24], a few intermediate layers are directly connected to auxiliary classifiers for addressing vanishing/exploding gradients. The papers of [38, 37, 31, 46] propose methods for centering layer responses, gradients, and propagated errors, implemented by shortcut connections. In [43], an "inception" layer is composed of a shortcut branch and a few deeper branches.

Concurrent with our work, "highway networks" [41, 42] present shortcut connections with gating functions [15]. These gates are data-dependent and have parameters, in contrast to our identity shortcuts that are parameter-free. When a gated shortcut is "closed" (approaching zero), the layers in highway networks represent *non-residual* functions. On the contrary, our formulation always learns residual functions; our identity shortcuts are never closed, and all information is always passed through, with additional residual functions to be learned. In addition, high-

way networks have not demonstrated accuracy gains with extremely increased depth (*e.g.*, over 100 layers).

3. Deep Residual Learning

3.1. Residual Learning

Let us consider $\mathcal{H}(\mathbf{x})$ as an underlying mapping to be fit by a few stacked layers (not necessarily the entire net), with $\mathbf{x}$ denoting the inputs to the first of these layers. If one hypothesizes that multiple nonlinear layers can asymptotically approximate complicated functions[2], then it is equivalent to hypothesize that they can asymptotically approximate the residual functions, *i.e.*, $\mathcal{H}(\mathbf{x}) - \mathbf{x}$ (assuming that the input and output are of the same dimensions). So rather than expect stacked layers to approximate $\mathcal{H}(\mathbf{x})$, we explicitly let these layers approximate a residual function $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$. The original function thus becomes $\mathcal{F}(\mathbf{x})+\mathbf{x}$. Although both forms should be able to asymptotically approximate the desired functions (as hypothesized), the ease of learning might be different.

This reformulation is motivated by the counterintuitive phenomena about the degradation problem (Fig. 1, left). As we discussed in the introduction, if the added layers can be constructed as identity mappings, a deeper model should have training error no greater than its shallower counterpart. The degradation problem suggests that the solvers might have difficulties in approximating identity mappings by multiple nonlinear layers. With the residual learning reformulation, if identity mappings are optimal, the solvers may simply drive the weights of the multiple nonlinear layers toward zero to approach identity mappings.

In real cases, it is unlikely that identity mappings are optimal, but our reformulation may help to precondition the problem. If the optimal function is closer to an identity mapping than to a zero mapping, it should be easier for the solver to find the perturbations with reference to an identity mapping, than to learn the function as a new one. We show by experiments (Fig. 7) that the learned residual functions in general have small responses, suggesting that identity mappings provide reasonable preconditioning.

3.2. Identity Mapping by Shortcuts

We adopt residual learning to every few stacked layers. A building block is shown in Fig. 2. Formally, in this paper we consider a building block defined as:

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

Here $\mathbf{x}$ and $\mathbf{y}$ are the input and output vectors of the layers considered. The function $\mathcal{F}(\mathbf{x}, \{W_i\})$ represents the residual mapping to be learned. For the example in Fig. 2 that has two layers, $\mathcal{F} = W_2\sigma(W_1\mathbf{x})$ in which σ denotes ReLU [29] and the biases are omitted for simplifying notations. The operation $\mathcal{F} + \mathbf{x}$ is performed by a shortcut connection and element-wise addition. We adopt the second nonlinearity after the addition (*i.e.*, $\sigma(\mathbf{y})$, see Fig. 2).

The shortcut connections in Eqn.(1) introduce neither extra parameter nor computation complexity. This is not only attractive in practice but also important in our comparisons between plain and residual networks. We can fairly compare plain/residual networks that simultaneously have the same number of parameters, depth, width, and computational cost (except for the negligible element-wise addition).

The dimensions of $\mathbf{x}$ and $\mathcal{F}$ must be equal in Eqn.(1). If this is not the case (*e.g.*, when changing the input/output channels), we can perform a linear projection W_s by the shortcut connections to match the dimensions:

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

We can also use a square matrix W_s in Eqn.(1). But we will show by experiments that the identity mapping is sufficient for addressing the degradation problem and is economical, and thus W_s is only used when matching dimensions.

The form of the residual function $\mathcal{F}$ is flexible. Experiments in this paper involve a function $\mathcal{F}$ that has two or three layers (Fig. 5), while more layers are possible. But if $\mathcal{F}$ has only a single layer, Eqn.(1) is similar to a linear layer: $\mathbf{y} = W_1\mathbf{x} + \mathbf{x}$, for which we have not observed advantages.

We also note that although the above notations are about fully-connected layers for simplicity, they are applicable to convolutional layers. The function $\mathcal{F}(\mathbf{x}, \{W_i\})$ can represent multiple convolutional layers. The element-wise addition is performed on two feature maps, channel by channel.

3.3. Network Architectures

We have tested various plain/residual nets, and have observed consistent phenomena. To provide instances for discussion, we describe two models for ImageNet as follows.

Plain Network. Our plain baselines (Fig. 3, middle) are mainly inspired by the philosophy of VGG nets [40] (Fig. 3, left). The convolutional layers mostly have 3×3 filters and follow two simple design rules: (i) for the same output feature map size, the layers have the same number of filters; and (ii) if the feature map size is halved, the number of filters is doubled so as to preserve the time complexity per layer. We perform downsampling directly by convolutional layers that have a stride of 2. The network ends with a global average pooling layer and a 1000-way fully-connected layer with softmax. The total number of weighted layers is 34 in Fig. 3 (middle).

It is worth noticing that our model has *fewer* filters and *lower* complexity than VGG nets [40] (Fig. 3, left). Our 34-layer baseline has 3.6 billion FLOPs (multiply-adds), which is only 18% of VGG-19 (19.6 billion FLOPs).

[2]This hypothesis, however, is still an open question. See [28].

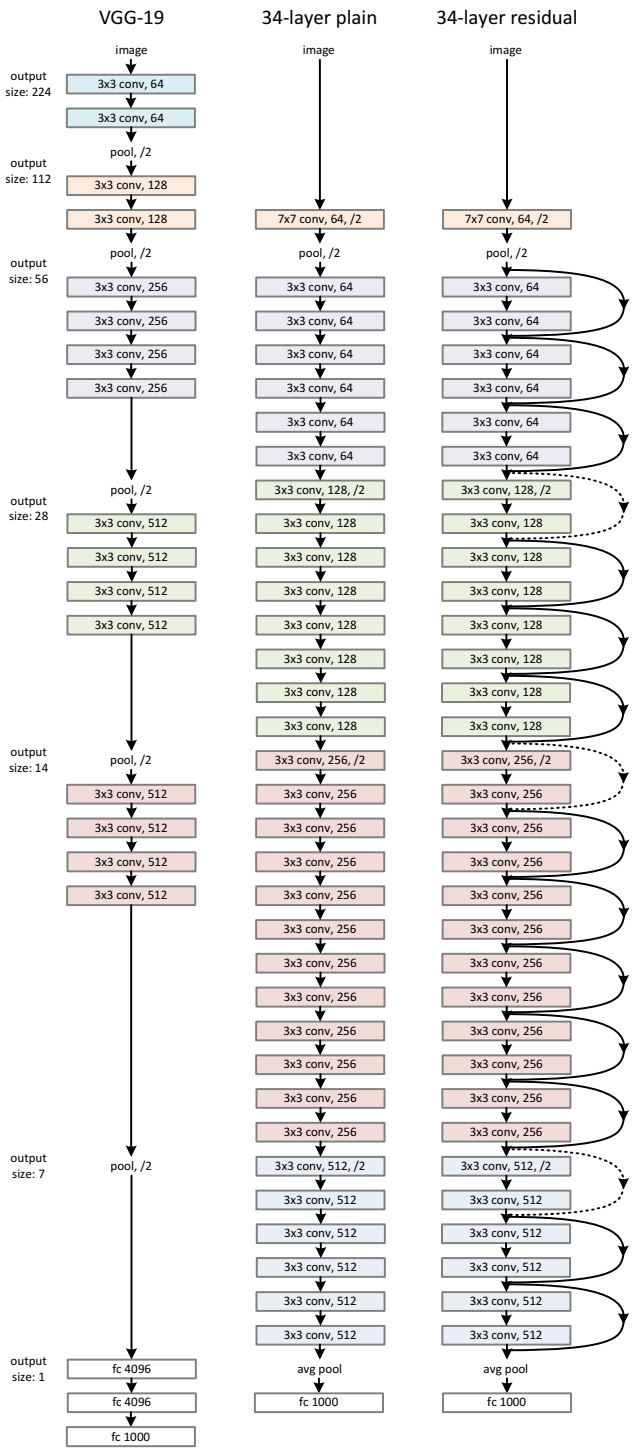

Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [40] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Residual Network. Based on the above plain network, we insert shortcut connections (Fig. 3, right) which turn the network into its counterpart residual version. The identity shortcuts (Eqn.(1)) can be directly used when the input and output are of the same dimensions (solid line shortcuts in Fig. 3). When the dimensions increase (dotted line shortcuts in Fig. 3), we consider two options: (A) The shortcut still performs identity mapping, with extra zero entries padded for increasing dimensions. This option introduces no extra parameter; (B) The projection shortcut in Eqn.(2) is used to match dimensions (done by 1×1 convolutions). For both options, when the shortcuts go across feature maps of two sizes, they are performed with a stride of 2.

3.4. Implementation

Our implementation for ImageNet follows the practice in [21, 40]. The image is resized with its shorter side randomly sampled in $[256, 480]$ for scale augmentation [40]. A 224×224 crop is randomly sampled from an image or its horizontal flip, with the per-pixel mean subtracted [21]. The standard color augmentation in [21] is used. We adopt batch normalization (BN) [16] right after each convolution and before activation, following [16]. We initialize the weights as in [12] and train all plain/residual nets from scratch. We use SGD with a mini-batch size of 256. The learning rate starts from 0.1 and is divided by 10 when the error plateaus, and the models are trained for up to 60×10^4 iterations. We use a weight decay of 0.0001 and a momentum of 0.9. We do not use dropout [13], following the practice in [16].

In testing, for comparison studies we adopt the standard 10-crop testing [21]. For best results, we adopt the fully-convolutional form as in [40, 12], and average the scores at multiple scales (images are resized such that the shorter side is in $\{224, 256, 384, 480, 640\}$).

4. Experiments

4.1. ImageNet Classification

We evaluate our method on the ImageNet 2012 classification dataset [35] that consists of 1000 classes. The models are trained on the 1.28 million training images, and evaluated on the 50k validation images. We also obtain a final result on the 100k test images, reported by the test server. We evaluate both top-1 and top-5 error rates.

Plain Networks. We first evaluate 18-layer and 34-layer plain nets. The 34-layer plain net is in Fig. 3 (middle). The 18-layer plain net is of a similar form. See Table 1 for detailed architectures.

The results in Table 2 show that the deeper 34-layer plain net has higher validation error than the shallower 18-layer plain net. To reveal the reasons, in Fig. 4 (left) we compare their training/validation errors during the training procedure. We have observed the degradation problem - the

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. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Downsampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

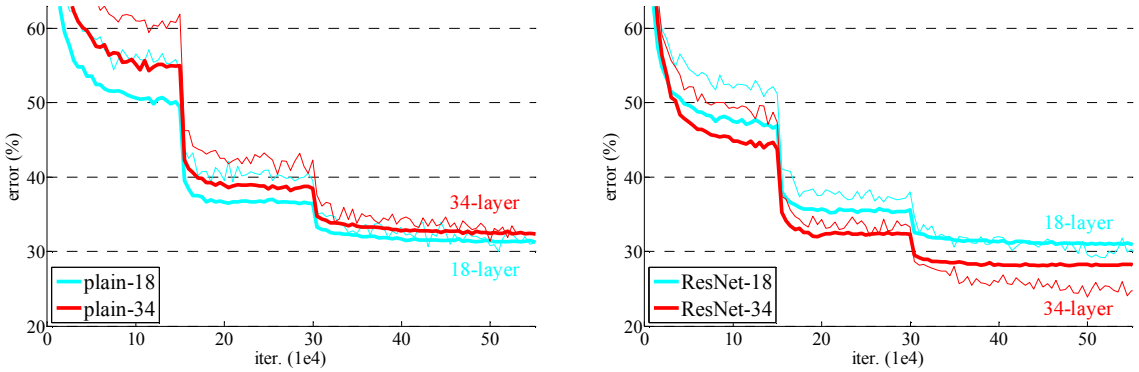

Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

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

Table 2. Top-1 error (%, 10-crop testing) on ImageNet validation. Here the ResNets have no extra parameter compared to their plain counterparts. Fig. 4 shows the training procedures.

34-layer plain net has higher *training* error throughout the whole training procedure, even though the solution space of the 18-layer plain network is a subspace of that of the 34-layer one.

We argue that this optimization difficulty is *unlikely* to be caused by vanishing gradients. These plain networks are trained with BN [16], which ensures forward propagated signals to have non-zero variances. We also verify that the backward propagated gradients exhibit healthy norms with BN. So neither forward nor backward signals vanish. In fact, the 34-layer plain net is still able to achieve competitive accuracy (Table 3), suggesting that the solver works to some extent. We conjecture that the deep plain nets may have exponentially low convergence rates, which impact the reducing of the training error[3]. The reason for such optimization difficulties will be studied in the future.

Residual Networks. Next we evaluate 18-layer and 34-layer residual nets (*ResNets*). The baseline architectures are the same as the above plain nets, expect that a shortcut connection is added to each pair of 3×3 filters as in Fig. 3 (right). In the first comparison (Table 2 and Fig. 4 right), we use identity mapping for all shortcuts and zero-padding for increasing dimensions (option A). So they have *no extra parameter* compared to the plain counterparts.

We have three major observations from Table 2 and Fig. 4. First, the situation is reversed with residual learning – the 34-layer ResNet is better than the 18-layer ResNet (by 2.8%). More importantly, the 34-layer ResNet exhibits considerably lower training error and is generalizable to the validation data. This indicates that the degradation problem is well addressed in this setting and we manage to obtain accuracy gains from increased depth.

Second, compared to its plain counterpart, the 34-layer

[3]We have experimented with more training iterations (3×) and still observed the degradation problem, suggesting that this problem cannot be feasibly addressed by simply using more iterations.

model	top-1 err.	top-5 err.
VGG-16 [40]	28.07	9.33
GoogLeNet [43]	-	9.15
PReLU-net [12]	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. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

method	top-1 err.	top-5 err.
VGG [40] (ILSVRC'14)	-	8.43†
GoogLeNet [43] (ILSVRC'14)	-	7.89
VGG [40] (v5)	24.4	7.1
PReLU-net [12]	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. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

method	top-5 err. (**test**)
VGG [40] (ILSVRC'14)	7.32
GoogLeNet [43] (ILSVRC'14)	6.66
VGG [40] (v5)	6.8
PReLU-net [12]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

ResNet reduces the top-1 error by 3.5% (Table 2), resulting from the successfully reduced training error (Fig. 4 right *vs*. left). This comparison verifies the effectiveness of residual learning on extremely deep systems.

Last, we also note that the 18-layer plain/residual nets are comparably accurate (Table 2), but the 18-layer ResNet converges faster (Fig. 4 right *vs*. left). When the net is "not overly deep" (18 layers here), the current SGD solver is still able to find good solutions to the plain net. In this case, the ResNet eases the optimization by providing faster convergence at the early stage.

Identity *vs*. Projection Shortcuts. We have shown that parameter-free, identity shortcuts help with training. Next we investigate projection shortcuts (Eqn.(2)). In Table 3 we compare three options: (A) zero-padding shortcuts are used for increasing dimensions, and all shortcuts are parameter-free (the same as Table 2 and Fig. 4 right); (B) projection shortcuts are used for increasing dimensions, and other shortcuts are identity; and (C) all shortcuts are projections.

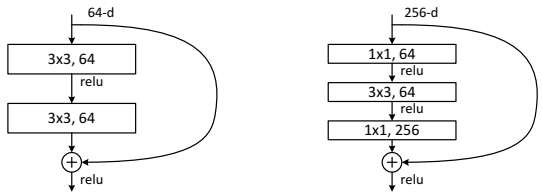

Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

Table 3 shows that all three options are considerably better than the plain counterpart. B is slightly better than A. We argue that this is because the zero-padded dimensions in A indeed have no residual learning. C is marginally better than B, and we attribute this to the extra parameters introduced by many (thirteen) projection shortcuts. But the small differences among A/B/C indicate that projection shortcuts are not essential for addressing the degradation problem. So we do not use option C in the rest of this paper, to reduce memory/time complexity and model sizes. Identity shortcuts are particularly important for not increasing the complexity of the bottleneck architectures that are introduced below.

Deeper Bottleneck Architectures. Next we describe our deeper nets for ImageNet. Because of concerns on the training time that we can afford, we modify the building block as a *bottleneck* design[4]. For each residual function $\mathcal{F}$, we use a stack of 3 layers instead of 2 (Fig. 5). The three layers are 1×1, 3×3, and 1×1 convolutions, where the 1×1 layers are responsible for reducing and then increasing (restoring) dimensions, leaving the 3×3 layer a bottleneck with smaller input/output dimensions. Fig. 5 shows an example, where both designs have similar time complexity.

The parameter-free identity shortcuts are particularly important for the bottleneck architectures. If the identity shortcut in Fig. 5 (right) is replaced with projection, one can show that the time complexity and model size are doubled, as the shortcut is connected to the two high-dimensional ends. So identity shortcuts lead to more efficient models for the bottleneck designs.

50-layer ResNet: We replace each 2-layer block in the

[4]Deeper *non*-bottleneck ResNets (*e.g*., Fig. 5 left) also gain accuracy from increased depth (as shown on CIFAR-10), but are not as economical as the bottleneck ResNets. So the usage of bottleneck designs is mainly due to practical considerations. We further note that the degradation problem of plain nets is also witnessed for the bottleneck designs.

34-layer net with this 3-layer bottleneck block, resulting in a 50-layer ResNet (Table 1). We use option B for increasing dimensions. This model has 3.8 billion FLOPs.

101-layer and 152-layer ResNets: We construct 101-layer and 152-layer ResNets by using more 3-layer blocks (Table 1). Remarkably, although the depth is significantly increased, the 152-layer ResNet (11.3 billion FLOPs) still has *lower complexity* than VGG-16/19 nets (15.3/19.6 billion FLOPs).

The 50/101/152-layer ResNets are more accurate than the 34-layer ones by considerable margins (Table 3 and 4). We do not observe the degradation problem and thus enjoy significant accuracy gains from considerably increased depth. The benefits of depth are witnessed for all evaluation metrics (Table 3 and 4).

Comparisons with State-of-the-art Methods. In Table 4 we compare with the previous best single-model results. Our baseline 34-layer ResNets have achieved very competitive accuracy. Our 152-layer ResNet has a single-model top-5 validation error of 4.49%. This single-model result outperforms all previous ensemble results (Table 5). We combine six models of different depth to form an ensemble (only with two 152-layer ones at the time of submitting). This leads to **3.57%** top-5 error on the test set (Table 5). *This entry won the 1st place in ILSVRC 2015.*

4.2. CIFAR-10 and Analysis

We conducted more studies on the CIFAR-10 dataset [20], which consists of 50k training images and 10k testing images in 10 classes. We present experiments trained on the training set and evaluated on the test set. Our focus is on the behaviors of extremely deep networks, but not on pushing the state-of-the-art results, so we intentionally use simple architectures as follows.

The plain/residual architectures follow the form in Fig. 3 (middle/right). The network inputs are 32×32 images, with the per-pixel mean subtracted. The first layer is 3×3 convolutions. Then we use a stack of $6n$ layers with 3×3 convolutions on the feature maps of sizes $\{32, 16, 8\}$ respectively, with $2n$ layers for each feature map size. The numbers of filters are $\{16, 32, 64\}$ respectively. The subsampling is performed by convolutions with a stride of 2. The network ends with a global average pooling, a 10-way fully-connected layer, and softmax. There are totally $6n$+2 stacked weighted layers. The following table summarizes the architecture:

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

When shortcut connections are used, they are connected to the pairs of 3×3 layers (totally $3n$ shortcuts). On this dataset we use identity shortcuts in all cases (*i.e.*, option A), so our residual models have exactly the same depth, width, and number of parameters as the plain counterparts.

method			error (%)
Maxout [9]			9.38
NIN [25]			8.81
DSN [24]			8.22
	# layers	# params	
FitNet [34]	19	2.5M	8.39
Highway [41, 42]	19	2.3M	7.54 (7.72±0.16)
Highway [41, 42]	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. Classification error on the **CIFAR-10** test set. All methods are with data augmentation. For ResNet-110, we run it 5 times and show "best (mean±std)" as in [42].

We use a weight decay of 0.0001 and momentum of 0.9, and adopt the weight initialization in [12] and BN [16] but with no dropout. These models are trained with a mini-batch size of 128 on two GPUs. We start with a learning rate of 0.1, divide it by 10 at 32k and 48k iterations, and terminate training at 64k iterations, which is determined on a 45k/5k train/val split. We follow the simple data augmentation in [24] for training: 4 pixels are padded on each side, and a 32×32 crop is randomly sampled from the padded image or its horizontal flip. For testing, we only evaluate the single view of the original 32×32 image.

We compare $n = \{3, 5, 7, 9\}$, leading to 20, 32, 44, and 56-layer networks. Fig. 6 (left) shows the behaviors of the plain nets. The deep plain nets suffer from increased depth, and exhibit higher training error when going deeper. This phenomenon is similar to that on ImageNet (Fig. 4, left) and on MNIST (see [41]), suggesting that such an optimization difficulty is a fundamental problem.

Fig. 6 (middle) shows the behaviors of ResNets. Also similar to the ImageNet cases (Fig. 4, right), our ResNets manage to overcome the optimization difficulty and demonstrate accuracy gains when the depth increases.

We further explore $n = 18$ that leads to a 110-layer ResNet. In this case, we find that the initial learning rate of 0.1 is slightly too large to start converging[5]. So we use 0.01 to warm up the training until the training error is below 80% (about 400 iterations), and then go back to 0.1 and continue training. The rest of the learning schedule is as done previously. This 110-layer network converges well (Fig. 6, middle). It has *fewer* parameters than other deep and thin

[5]With an initial learning rate of 0.1, it starts converging (<90% error) after several epochs, but still reaches similar accuracy.

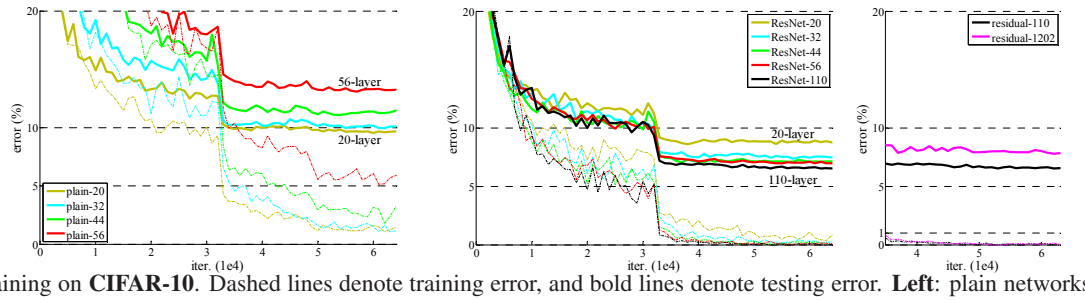

Figure 6. Training on **CIFAR-10**. Dashed lines denote training error, and bold lines denote testing error. **Left**: plain networks. The error of plain-110 is higher than 60% and not displayed. **Middle**: ResNets. **Right**: ResNets with 110 and 1202 layers.

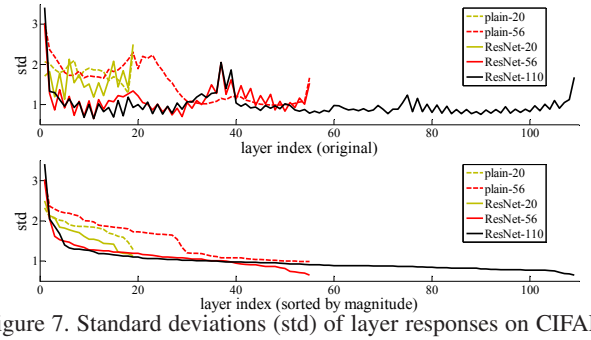

Figure 7. Standard deviations (std) of layer responses on CIFAR-10. The responses are the outputs of each 3×3 layer, after BN and before nonlinearity. **Top**: the layers are shown in their original order. **Bottom**: the responses are ranked in descending order.

networks such as FitNet [34] and Highway [41] (Table 6), yet is among the state-of-the-art results (6.43%, Table 6).

Analysis of Layer Responses. Fig. 7 shows the standard deviations (std) of the layer responses. The responses are the outputs of each 3×3 layer, after BN and before other nonlinearity (ReLU/addition). For ResNets, this analysis reveals the response strength of the residual functions. Fig. 7 shows that ResNets have generally smaller responses than their plain counterparts. These results support our basic motivation (Sec.3.1) that the residual functions might be generally closer to zero than the non-residual functions. We also notice that the deeper ResNet has smaller magnitudes of responses, as evidenced by the comparisons among ResNet-20, 56, and 110 in Fig. 7. When there are more layers, an individual layer of ResNets tends to modify the signal less.

Exploring Over 1000 layers. We explore an aggressively deep model of over 1000 layers. We set $n = 200$ that leads to a 1202-layer network, which is trained as described above. Our method shows *no optimization difficulty*, and this 10^3-layer network is able to achieve *training error* <0.1% (Fig. 6, right). Its test error is still fairly good (7.93%, Table 6).

But there are still open problems on such aggressively deep models. The testing result of this 1202-layer network is worse than that of our 110-layer network, although both have similar training error. We argue that this is because of overfitting. The 1202-layer network may be unnecessarily large (19.4M) for this small dataset. Strong regularization such as maxout [9] or dropout [13] is applied to obtain the best results ([9, 25, 24, 34]) on this dataset. In this paper, we use no maxout/dropout and just simply impose regularization via deep and thin architectures by design, without distracting from the focus on the difficulties of optimization. But combining with stronger regularization may improve results, which we will study in the future.

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. Object detection mAP (%) on the PASCAL VOC 2007/2012 test sets using **baseline** Faster R-CNN. See also appendix for better results.

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

Table 8. Object detection mAP (%) on the COCO validation set using **baseline** Faster R-CNN. See also appendix for better results.

4.3. Object Detection on PASCAL and MS COCO

Our method has good generalization performance on other recognition tasks. Table 7 and 8 show the object detection baseline results on PASCAL VOC 2007 and 2012 [5] and COCO [26]. We adopt *Faster R-CNN* [32] as the detection method. Here we are interested in the improvements of replacing VGG-16 [40] with ResNet-101. The detection implementation (see appendix) of using both models is the same, so the gains can only be attributed to better networks. Most remarkably, on the challenging COCO dataset we obtain a 6.0% increase in COCO's standard metric (mAP@[.5, .95]), which is a 28% relative improvement. This gain is solely due to the learned representations.

Based on deep residual nets, we won the 1st places in several tracks in ILSVRC & COCO 2015 competitions: ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation. The details are in the appendix.

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

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

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

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

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

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

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

[13] 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.

[14] S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma thesis, TU Munich*, 1991.

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

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

[35] 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.

[36] 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.

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

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

[39] 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.

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

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

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

[43] 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.

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

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

[46] 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.

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

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

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