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 [41] 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, 50, 40]. Deep networks naturally integrate low/mid/high-level features [50] 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 [41, 44] reveals that network depth is of crucial importance, and the leading results [41, 44, 13, 16] on the challenging ImageNet dataset [36] all exploit "very deep" [41] models, with a depth of sixteen [41] to thirty [16]. Many other non-trivial visual recognition tasks [8, 12, 7, 32, 27] have also greatly benefited from very deep models.

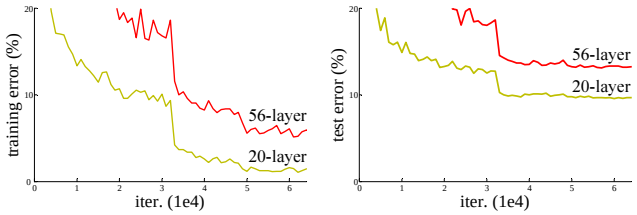

Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

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 [1, 9], which hamper convergence from the beginning. This problem, however, has been largely addressed by normalized initialization [23, 9, 37, 13] 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 [11, 42] 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.

深度残差学习在图像识别中的应用

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

摘要

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 [41] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet 测试
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. 引言

深度卷积神经网络[22, 21]引领了图像分类领域的一系列突破[21, 50, 40]。深度网络以端到端的多层方式，自然整合了低/中/高层特征[50]与分类器，且通过叠加层数（深度）可丰富特征的"层级"。最新研究[41, 44]表明网络深度至关重要，在极具挑战性的ImageNet数据集[36]上取得领先成果的模型[41, 44, 13, 16]均采用了"极深度"[41]架构，其深度从十六层[41]至三十层[16]不等。许多其他重要的视觉识别任务[8, 12, 7, 32, 27]同样...

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

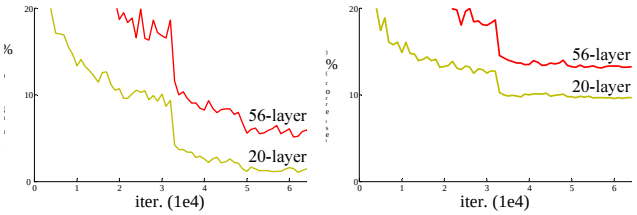

图1. 在CIFAR-10数据集上，20层与56层“普通”网络的训练误差（左）及测试误差（右）。更深的网络表现出更高的训练误差，因此测试误差也随之增加。图4展示了ImageNet数据集上的类似现象。

极大地受益于非常深的模型。

受深度重要性的驱动，一个问题随之产生：*Is learning better networks as easy as stacking more layers?*。回答这一问题的障碍在于著名的梯度消失/爆炸问题[1, 9]，它从一开始就阻碍了收敛。然而，通过归一化初始化[23, 9, 37, 13]和中间归一化层[16]，这一问题已得到很大程度的解决，这些技术使得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。

当更深的网络能够开始收敛时，一个*degradation*问题暴露出来：随着网络深度的增加，准确率会先达到饱和（这可能并不意外），然后迅速下降。出乎意料的是，这种退化是*not caused by overfitting*的，并且如[11,42]所述及我们的实验充分验证的那样，向一个已经足够深的模型继续添加层数会导致*higher train-ing error*。图1展示了一个典型例子。

训练准确度的下降表明，并非所有系统都同样易于优化。让我们考虑一个较浅的架构及其更深层的对应版本，后者在前者基础上增加了更多层。对于深层模型存在一个解*by construction*：新增层执行*identity*映射，其余层则复制自已学习的浅层模型。这一构造解的存在意味着，深层模型的训练误差不应高于其浅层对应版本。但实验表明，我们现有的求解器无法找到这样的解，以至于

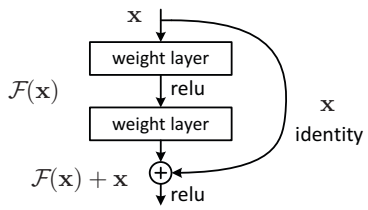

Figure 2. Residual learning: a building block.

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, 34, 49] 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 [36] 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 [36], 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 [41]. 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, 48]. 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 [45, 46], which relies on variables that represent residual vectors between two scales. It has been shown [3, 45, 46] 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, 34, 49] 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 [34, 49]. In [44, 24], a few intermediate layers are directly connected to auxiliary classifiers for addressing vanishing/exploding gradients. The papers of [39, 38, 31, 47] propose methods for centering layer responses, gradients, and propagated errors, implemented by shortcut connections. In [44], an "inception" layer is composed of a shortcut branch and a few deeper branches.

Concurrent with our work, "highway networks" [42, 43] 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-

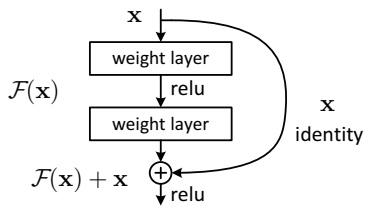

图2. 残差学习：一个构建模块。

与构建的解决方案相当或更好（或在可行时间内无法做到）。

在本文中，我们通过引入*deep residual learning*框架来解决退化问题。不同于期望每个堆叠层直接拟合期望的底层映射，我们明确让这些层拟合残差映射。形式上，将期望的底层映射表示为$\mathcal{H}(\mathbf{x})$，我们让堆叠的非线性层拟合另一个映射$\mathcal{F}(\mathbf{x}):=\mathcal{H}(\mathbf{x})-\mathbf{x}$。原始映射被重写为$\mathcal{F}(\mathbf{x})+\mathbf{x}$。我们假设优化残差映射比优化原始的、无参考的映射更为容易。极端情况下，若恒等映射是最优解，将残差推至零比通过堆叠非线性层拟合恒等映射更为简单。

$\mathcal{F}(\mathbf{x})+\mathbf{x}$的公式化可以通过带有“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。在我们的案例中，这些快捷连接仅执行*identity*映射，其输出会与堆叠层的输出相加（图2）。恒等快捷连接既不引入额外参数，也不增加计算复杂度。整个网络仍可通过反向传播的SGD进行端到端训练，并且能够利用现有库（*e.g*，如Caffe[19]）轻松实现，无需修改求解器。

我们在ImageNet[36]上进行了全面的实验，以展示退化问题并评估我们的方法。实验表明：1）我们极深的残差网络易于优化，而与之对应的“普通”网络（仅简单堆叠层）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能够轻松从大幅增加的深度中获得准确率提升，产生的效果显著优于以往的网络。

类似现象在CIFAR-10数据集[20]上同样显现，这表明优化难题及我们方法的效果并非仅局限于特定数据集。我们成功训练了该数据集上超过100层的模型，并探索了具有超过1000层的模型架构。

在ImageNet分类数据集[36]上，我们通过极深的残差网络取得了卓越成果。我们提出的152层残差网络是ImageNet上迄今最深的网络架构，同时其复杂度仍低于VGG网络[41]。我们的集成模型在测试集上实现了3.57%的top-5错误率

ImageNet *test* 数据集，以及
won the 1st place in the ILSVRC 2015 classification competition。极深的表示在其他识别任务上也展现出卓越的泛化性能，并促使我们在ILSVRC & COCO 2015竞赛中进一步*win the 1st places on: ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation*。这一有力证据表明，残差学习原理具有普适性，我们预期它同样适用于其他视觉及非视觉问题。

2. 相关工作

残差表示。在图像识别领域，VLAD[18]是一种基于字典残差向量编码的表示方法，而Fisher Vector[30]可视为VLAD的概率化版本[18]。这两种方法均为图像检索与分类任务中强大的浅层表示[4,48]。在向量量化方面，研究表明对残差向量进行编码[17]比直接编码原始向量更具效力。

在低级视觉与计算机图形学中，为求解偏微分方程（PDEs），广泛采用的多重网格法[3]将系统重构为多尺度下的子问题，每个子问题负责处理相邻粗细尺度间的残差解。另一种替代方案是分层基预处理[45,46]，该方法依赖于表示两尺度间残差向量的变量。已有研究[3,45,46]表明，这些求解器的收敛速度远超那些未考虑解残差特性的标准求解器。这些方法揭示出：良好的重构或预处理能有效简化优化过程。

捷径连接。导致捷径连接的实践与理论[2, 34, 49]已被研究多年。训练多层感知机(MLPs)的早期实践中，就有从网络输入直接连接到输出的线性层添加方式[34, 49]。在[44, 24]中，若干中间层被直接连至辅助分类器，以解决梯度消失/爆炸问题。[39, 38, 31, 47]等论文提出了通过捷径连接实现层响应、梯度及传播误差中心化的方法。[44]中描述的"初始"层，便是由一条捷径分支和若干更深分支构成。

与我们的工作同时，“高速公路网络”[42, 43]提出了带有门控功能的快捷连接[15]。这些门控是数据依赖且含参数的，而我们的恒等快捷连接则无参数。当门控快捷连接“关闭”（趋近于零）时，高速公路网络中的层表示*non-residual*函数。相反，我们的公式始终学习残差函数；恒等快捷连接永不关闭，所有信息始终通过，并额外学习残差函数。此外，高-

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 [41] (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 [41] (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].

深度网络在层数极大增加（如超过100层{v*}）时并未展现出准确性的提升。

3. 深度残差学习

3.1. 残差学习

让我们将$\mathcal{H}(\mathbf{x})$视为一个由若干堆叠层（不一定是整个网络）拟合的基础映射，其中$\mathbf{x}$表示这些层中第一层的输入。如果假设多个非线性层能够渐进逼近复杂函数[2]，那么这等同于假设它们能够渐进逼近残差函数*i.e*，$\mathcal{H}(\mathbf{x})-\mathbf{x}$ (前提是输入与输出的维度相同)。因此，我们不期望堆叠层直接逼近$\mathcal{H}(\mathbf{x})$，而是明确让这些层逼近一个残差函数$\mathcal{F}(\mathbf{x}):=\mathcal{H}(\mathbf{x})-\mathbf{x}$。于是原函数变为$\mathcal{F}(\mathbf{x})+\mathbf{x}$。尽管两种形式都应能渐进逼近目标函数（如假设所述），但学习的难易程度可能有所不同。

这种重新表述的动机源于关于退化问题的反直觉现象（图1左）。正如我们在引言中讨论的，如果增加的层可以被构建为恒等映射，那么更深的模型的训练误差不应大于其较浅的对应模型。退化问题表明，求解器可能在用多个非线性层逼近恒等映射时遇到困难。通过残差学习的重新表述，如果恒等映射是最优解，求解器可以简单地通过将多个非线性层的权重推向零来逼近恒等映射。

在实际情况下，恒等映射不太可能是最优解，但我们的重构可能有助于对问题进行预处理。如果最优函数更接近恒等映射而非零映射，那么求解器相对于恒等映射寻找扰动应当比从头学习新函数更为容易。实验表明（图7），学习到的残差函数通常响应值较小，这说明恒等映射提供了合理的预处理条件。

3.2. 通过捷径实现恒等映射

我们对每几个堆叠层采用残差学习。图2展示了一个构建块。形式上，在本文中，我们将构建块定义为：

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

这里$\mathbf{x}$和$\mathbf{y}$分别是所考虑层的输入与输出向量。函数$\mathcal{F}(\mathbf{x},\{W_i\})$表示待学习的残差映射。以图2中具有两层的示例为例，$\mathcal{F}=W_2\sigma(W_1\mathbf{x})$中的$\sigma$表示

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

ReLU [29]和偏置项被省略以简化表示。操作$\mathcal{F}+\mathbf{x}$通过一个快捷连接和逐元素加法完成。我们在加法后采用第二种非线性（*i.e*，$\sigma(\mathbf{y})$，见图2）。

式(1)中的快捷连接既没有引入额外参数，也未增加计算复杂度。这不仅在实践中极具吸引力，而且对于我们比较普通网络与残差网络尤为重要。这使得我们能够公平地对比具有相同参数数量、深度、宽度及计算成本（可忽略不计的逐元素加法除外）的普通/残差网络。

在方程(1)中，$\mathbf{x}$和$\mathcal{F}$的维度必须相等。若不符合此条件（例如*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：

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

我们也可以在方程(1)中使用方阵W_s。但实验将证明，恒等映射足以解决退化问题且更为经济，因此W_s仅在匹配维度时使用。

残差函数$\mathcal{F}$的形式具有灵活性。本文实验涉及一个包含两到三层（图5）的函数$\mathcal{F}$，但更多层也是可能的。然而，若$\mathcal{F}$仅含单层，则方程(1)类似于线性层：$\mathbf{y}=W_1\mathbf{x}+\mathbf{x}$，对此我们尚未观察到优势。

我们也注意到，尽管上述符号为了简洁起见主要针对全连接层，但它们同样适用于卷积层。函数{v*}可以表示多个卷积层的组合。逐元素的加法操作是在两个特征图上逐通道进行的。

3.3. 网络架构

我们测试了多种普通/残差网络，并观察到了一致的现象。为了提供讨论实例，我们针对ImageNet描述如下两个模型。

普通网络。我们的普通基线模型（图3，中）主要受到VGG网络[41]（图3，左）设计理念的启发。卷积层大多采用3×3滤波器，并遵循两条简单设计原则：(i) 对于相同尺寸的输出特征图，各层滤波器数量保持一致；(ii) 当特征图尺寸减半时，滤波器数量加倍以确保单层时间复杂度不变。我们直接通过步长为2的卷积层实现下采样。网络末端由全局平均池化层和带softmax的1000维全连接层构成。图3（中）所示结构的加权层总数达34层。

值得注意的是，我们的模型拥有*fewer*个滤波器和*lower*复杂度，相较于VGG网络[41]（图3左）。我们的34层基线模型具有36亿次浮点运算（乘加操作），仅为VGG-19（196亿次浮点运算）的18%。

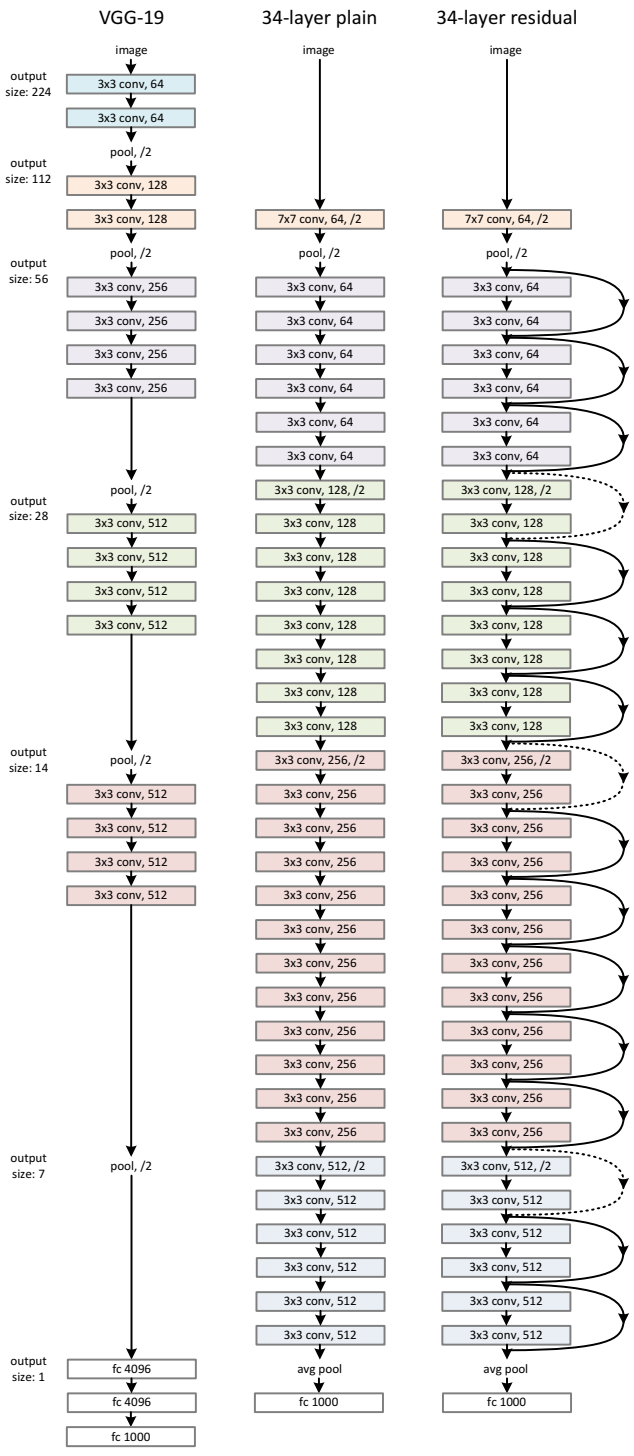

Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase 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, 41]. The image is resized with its shorter side randomly sampled in $[256, 480]$ for scale augmentation [41]. 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 [13] 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 [14], 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 [41, 13], 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 [36] 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

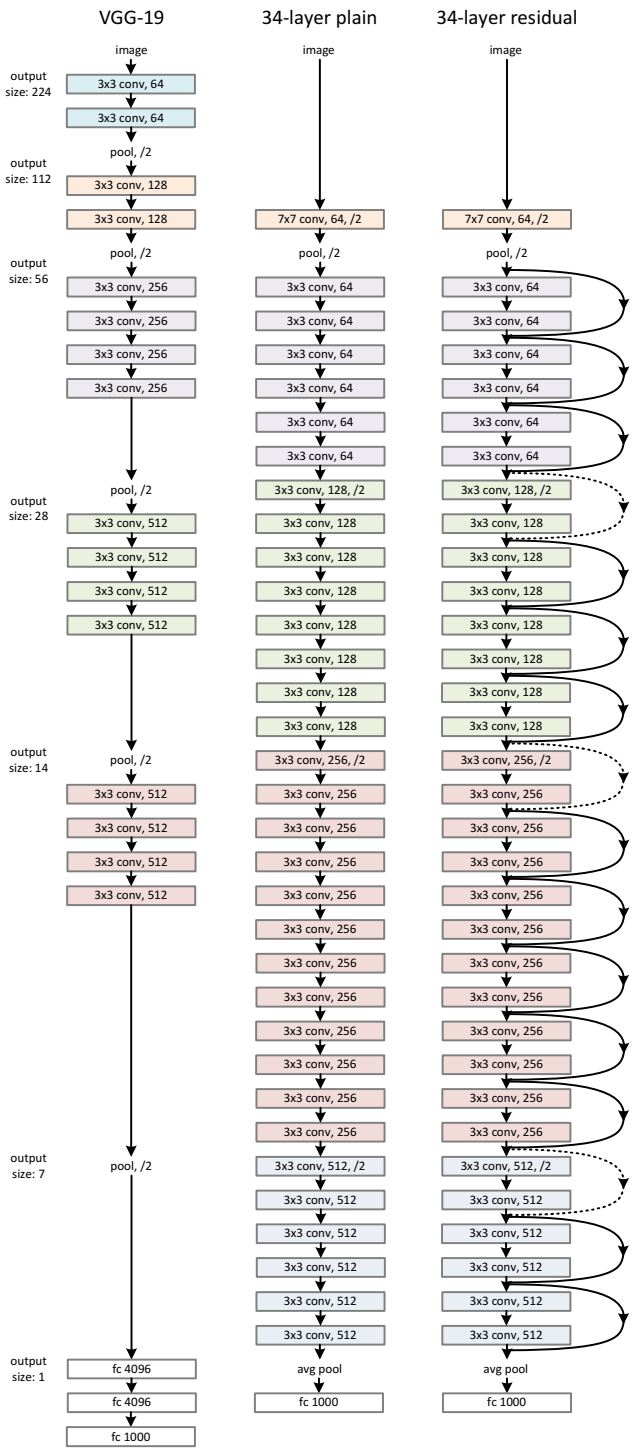

图3. ImageNet示例网络架构。左：作为参考的VGG-19模型[41]（196亿次浮点运算）。中：具有34个参数层的普通网络（36亿次浮点运算）。右：具有34个参数层的残差网络（36亿次浮点运算）。虚线捷径用于增加维度。表1展示了更多细节及其他变体。

残差网络。在上述普通网络的基础上，我们引入了快捷连接（图3右），将网络转变为对应的残差版本。当输入与输出维度相同时（图3中实线快捷路径），可直接使用恒等快捷连接（公式(1)）。当维度增加时（图3中虚线快捷路径），我们考虑两种方案：（A）快捷连接仍执行恒等映射，通过补零填充来增加维度，此方案不引入额外参数；（B）采用公式(2)中的投影快捷连接来匹配维度（通过1{v*}1卷积实现）。对于两种方案，当快捷连接跨越两种尺寸的特征图时，均采用步长为2的卷积操作。

3.4. 实现

我们在ImageNet上的实现遵循了[21, 41]中的做法。图像被调整大小，其短边在[256, 480]范围内随机采样以进行尺度增强[41]。从图像或其水平翻转中随机裁剪出224×224的区域，并减去像素均值[21]。采用了[21]中标准的色彩增强方法。按照[16]的做法，我们在每个卷积层后、激活函数前直接应用批量归一化（BN）[16]。权重初始化采用[13]的方法，所有普通/残差网络均从头开始训练。使用小批量大小为256的随机梯度下降（SGD）。学习率初始为0.1，在误差停滞时除以10，模型最多训练60××10^4次迭代。权重衰减设置为0.0001，动量为0.9。根据[16]的实践，我们未使用dropout[14]。

在测试阶段，为进行对比研究，我们采用了标准的10-crop测试方法[21]。为获得最佳效果，我们采用如[41, 13]所述的全卷积形式，并在多尺度上对分数进行平均（图像尺寸调整至短边为{224, 256, 384, 480, 640}）。

4. 实验

4.1. ImageNet分类

我们在ImageNet 2012分类数据集[36]上评估了我们的方法，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5万张验证图像上进行评估。我们还通过测试服务器获得了在10万张测试图像上的最终结果，并报告了top-1和top-5错误率。

普通网络。我们首先评估了18层和34层的普通网络。34层的普通网络如图3（中）所示。18层的普通网络结构与之类似。详细架构请参见表1。

表2中的结果显示，34层普通网络的验证误差高于较浅的18层普通网络。为了探究原因，我们在图4（左）中对比了它们在训练过程中的训练/验证误差。我们观察到了退化问题——

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.

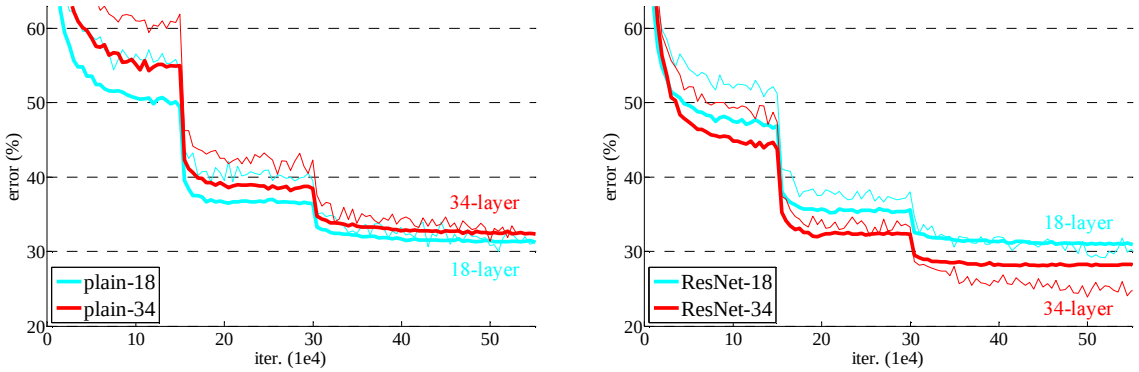

Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

	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.

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架构。构建块以括号形式展示（另见图5），并标注了堆叠的块数。下采样由conv3_1、conv4_1和conv5_1执行，步长为2。

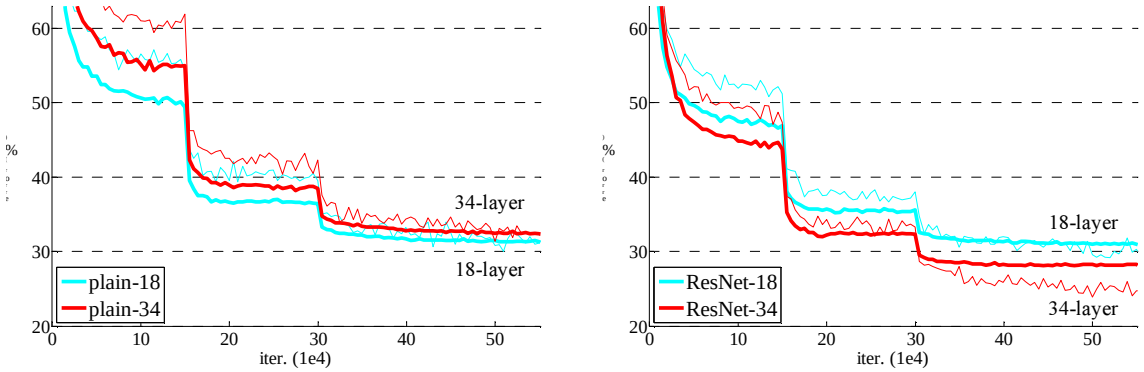

图4. ImageNet上的训练。细线表示训练误差，粗线表示中心裁剪的验证误差。左图：18层和34层的普通网络。右图：18层和34层的ResNet。在此图中，残差网络与其对应的普通网络相比没有额外参数。

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

表2. ImageNet验证集上的Top-1错误率（%，10-crop测试）。此处ResNet与其对应的普通网络相比未引入额外参数。训练过程如图4所示。

34层普通网络在整个训练过程中具有更高的*training*误差，尽管18层普通网络的解空间是34层网络解空间的一个子集。

我们认为，这种优化困难源于梯度消失问题。这些普通网络采用了批量归一化（BN）[16]进行训练，确保前向传播的信号具有非零方差。我们还验证了在BN作用下，反向传播的梯度保持健康范数。因此，无论是前向还是反向信号都不会消失。事实上，34层的普通网络仍能取得具有竞争力的准确率（表3），这表明求解器在一定程度上是有效的。我们推测，深层普通网络可能存在指数级降低的收敛速度，这影响了降低训练误差[3]。此类优化困难的原因将在未来进行研究。

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滤波器之间添加了快捷连接，如图3（右）所示。在第一次比较中（表2和图4右），我们对所有快捷方式使用恒等映射，并通过零填充来增加维度（选项A）。因此，与普通网络相比，它们具有*no extra parameter*。

从表2和图4中，我们得出三个主要观察结果。首先，残差学习使情况发生了逆转——34层ResNet的表现优于18层ResNet（高出2.8%）。更重要的是，34层ResNet展现出显著更低的训练误差，并且能够泛化至验证数据。这表明在此设置下退化问题得到了有效解决，我们成功通过增加网络深度获得了准确率的提升。

其次，与其普通版本相比，34层的

[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 [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. 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 [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. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

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. 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.

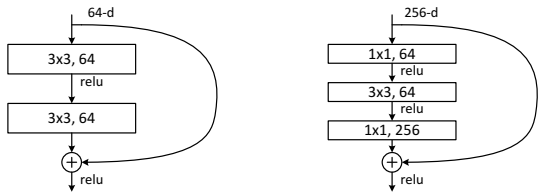

Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

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.

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**

表3. ImageNet验证集上的错误率（%，10-crop测试）。VGG-16基于我们的测试结果。ResNet-50/101/152采用仅通过投影增加维度的选项B配置。

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**

表4. ImageNet验证集上单模型结果的错误率(%)（除†报告于测试集外）。

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**

表5. 集成模型的错误率（%）。top-5错误率基于ImageNet测试集，并由测试服务器报告。

ResNet将top-1错误率降低了3.5%（表2），这源于训练误差的成功减小（图4右*vs.* 左）。这一对比验证了残差学习在极深度系统上的有效性。

最后，我们还注意到，18层的普通/残差网络在准确率上相当（表2），但18层的ResNet收敛速度更快（图4右*vs.*左）。当网络"不过于深"（此处为18层）时，当前的SGD求解器仍能为普通网络找到良好的解。在这种情况下，ResNet通过早期提供更快的收敛速度来简化优化过程。

标识 ***vs***。投影快捷方式。我们已经证明

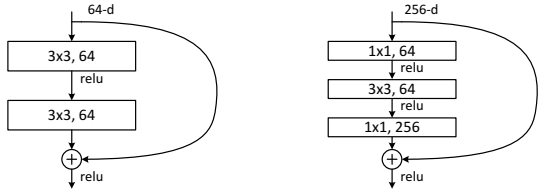

图5. 针对ImageNet的更深层残差函数$\mathcal{F}$。左图：一个构建块（在56×56特征图上），与图3中ResNet-34的结构相同。右图：ResNet-50/101/152的"瓶颈"构建块。

无参数的身份捷径有助于训练。接下来我们研究投影捷径（公式(2)）。在表3中我们比较了三种方案：(A) 使用零填充捷径进行维度增加，且所有捷径均为无参数（与表2及图4右侧相同）；(B) 使用投影捷径进行维度增加，其他捷径保持身份映射；(C) 所有捷径均为投影。

表3显示，所有三个选项均明显优于普通版本。B略优于A，我们认为这是因为A中零填充的维度确实未进行残差学习。C比B稍好一些，我们将此归因于众多（十三条）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关键。因此，在本文后续部分我们未采用选项C，以降低内存/时间复杂度和模型大小。恒等捷径对于不增加下文将介绍的瓶颈架构复杂度尤为重要。

更深的瓶颈架构。接下来我们描述为ImageNet设计的更深层网络。出于对可承受训练时间的考虑，我们将构建块修改为*bottleneck*设计[4]。对于每个残差函数$\mathcal{F}$，我们采用3层堆叠而非2层（图5所示）。这三层分别是1××1、3××3和1××1卷积，其中1××1层负责先降低后恢复（还原）维度，使3××3层成为具有较小输入/输出维度的瓶颈。图5展示了两种设计在时间复杂度上相近的实例。

无参数的恒等捷径对于瓶颈架构尤为重要。如果将图5（右）中的恒等捷径替换为投影，可以证明时间复杂度和模型大小会翻倍，因为捷径连接到了两个高维端。因此，恒等捷径为瓶颈设计带来了更高效的模型。

50层ResNet：我们将每个2层块替换为

[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 [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. 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 [43].

We use a weight decay of 0.0001 and momentum of 0.9, and adopt the weight initialization in [13] 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 [42]), 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.

采用这种3层瓶颈块的34层网络，最终形成一个50层的ResNet（表1）。我们选用选项B来增加维度。该模型具有38亿次浮点运算(FLOPs)。

101层和152层ResNet：我们通过使用更多的3层块构建了101层和152层的ResNet（表1）。值得注意的是，尽管深度显著增加，152层ResNet（113亿次浮点运算）的计算量*lower complexity*仍低于VGG-16/19网络（15 3/196亿次浮点运算）。

50/101/152层的ResNet在准确率上比34层的模型有显著提升（表3和表4）。我们并未观察到退化问题，因此通过大幅增加深度获得了可观的精度提升。深度的优势在所有评估指标中均得到验证（表3和表4）。

与最先进方法的比较。在表4中，我们与之前最佳单模型结果进行了对比。我们的基线34层ResNet已取得极具竞争力的准确率。152层ResNet模型的单模型top-5验证误差为4.49%，这一单模型表现超越了之前所有的集成方法结果（表5）。我们通过组合六个不同深度的模型形成集成（提交时仅包含两个152层模型），最终在测试集上实现了3.57%的top-5误差率（表5）。{v*}

4.2. CIFAR-10与分析

我们在CIFAR-10数据集[20]上进行了更多研究，该数据集包含10个类别的5万张训练图像和1万张测试图像。我们展示了在训练集上训练、测试集上评估的实验结果。我们的关注点在于极深度网络的行为特性，而非追求最先进的性能表现，因此我们有意采用如下简单架构。

普通/残差架构遵循图3（中/右）所示形式。网络输入为32×32像素的图像，并已减去逐像素均值。首层采用3×3卷积核。随后我们堆叠了6n层网络，分别在尺寸为{32,、16,、8}的特征图上进行3×3卷积操作，每种特征图尺寸对应2n个卷积层。滤波器数量分别为{16,、32,、64}。下采样通过步长为2的卷积实现。网络末端采用全局平均池化、10维全连接层及softmax分类器。整个网络共包含6n+2个堆叠的权重层。下表概括了该架构：

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，它们会连接到3×3层的各对（总共3n条快捷路径）。在此数据集上，我们在所有情况下均采用恒等快捷连接（*i.e*，即选项A），

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

表6. CIFAR-10测试集上的分类错误率。所有方法均采用数据增强。对于ResNet-110，我们运行5次并如[43]所示展示“最佳（均值±标准差）”。

因此我们的残差模型在深度、宽度和参数数量上与普通模型完全相同。

我们采用权重衰减为0.0001，动量为0.9，并采用[13]中的权重初始化方法及BN[16]技术，但未使用dropout。这些模型在两个GPU上以128的小批量规模进行训练。初始学习率设为0.1，在32k和48k次迭代时分别降至十分之一，训练在64k次迭代时终止，该设定基于45k/5k的训练/验证集划分确定。训练过程中遵循[24]中的简单数据增强策略：每边填充4像素后，从填充图像或其水平翻转中随机裁剪出32×32的区域。测试时，我们仅评估原始32×32图像的单视角表现。

我们比较了$n = \{3,5,7,9\}$，分别对应20层、32层、44层和56层网络。图6（左）展示了普通网络的表现。随着深度增加，深层普通网络的训练误差显著上升。这一现象与ImageNet（图4左）和MNIST数据集（见[42]）上的情况相似，表明此类优化难题是一个本质性问题。

图6（中）展示了ResNets的行为表现。与ImageNet案例（图4右）类似，我们的ResNets成功克服了优化难题，并在深度增加时展现出精度提升。

我们进一步探索了$n = 18$，这导向了一个110层的ResNet。在此情况下，我们发现初始学习率0.1略大，难以开始收敛[5]。因此，我们采用0.01的学习率进行训练预热，直至训练误差降至80%以下（约400次迭代），随后恢复至0.1并继续训练。其余学习率调度方式与先前一致。这个110层网络表现出良好的收敛性（图6中）。相比其他深而窄的网络，它具有*fewer*参数

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

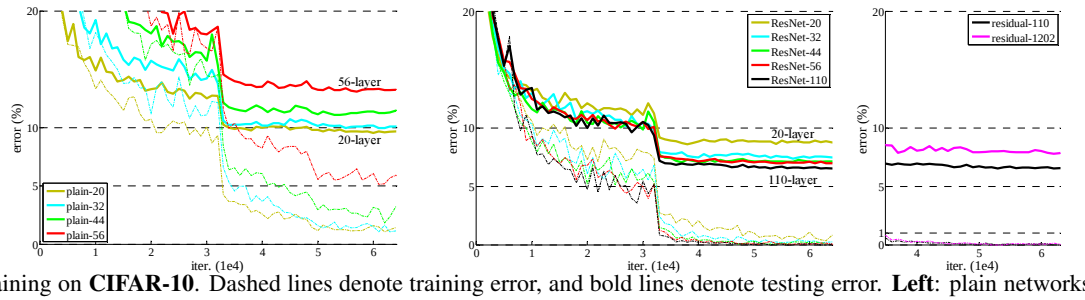

Figure 6. Training on **CIFAR-10**. Dashed lines denote training error, and bold lines denote testing error. **Left**: plain networks. The error of plain-110 is higher than 60% and not displayed. **Middle**: ResNets. **Right**: ResNets with 110 and 1202 layers.

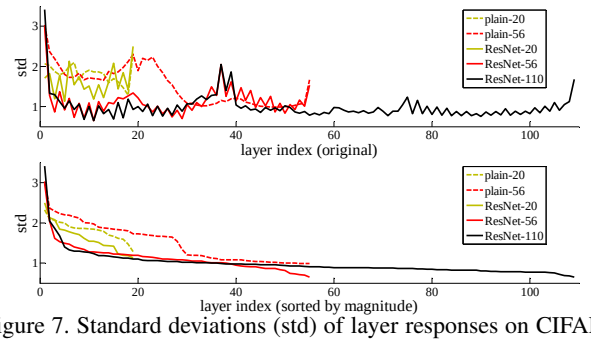

Figure 7. Standard deviations (std) of layer responses on CIFAR-10. The responses are the outputs of each 3×3 layer, after BN and before nonlinearity. **Top**: the layers are shown in their original order. **Bottom**: the responses are ranked in descending order.

networks such as FitNet [35] and Highway [42] (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 [10] or dropout [14] is applied to obtain the best results ([10, 25, 24, 35]) 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 Table 10 and 11 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 Table 9 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 [41] 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.

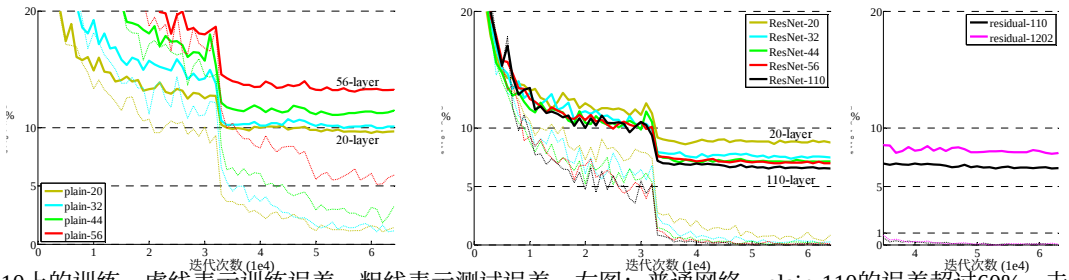

图6. CIFAR-10上的训练。虚线表示训练误差，粗线表示测试误差。左图：普通网络。plain-110的误差超过60%，未予显示。中图：ResNet网络。右图：110层和1202层的ResNet网络。

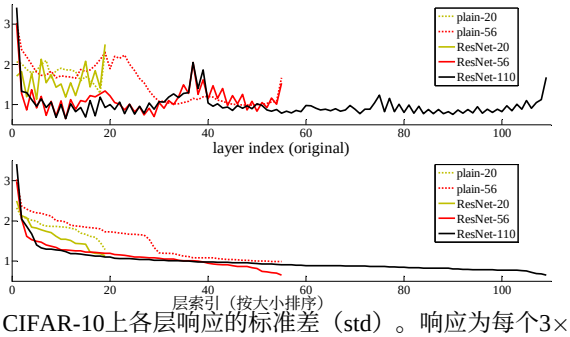

图7. CIFAR-10上各层响应的标准差（std）。响应为每个3×3层的输出，经过BN处理但未经过非线性激活。上图：各层按原始顺序排列。下图：响应按降序排列。

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**

表7. 使用基准Faster R-CNN在PASCAL VOC 2007/2012测试集上的目标检测mAP（%）。更优结果另见表10和11。

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

表8. 使用基准Faster R-CNN在COCO验证集上的目标检测mAP（%）。更佳结果另见表9。

诸如FitNet [35]和Highway [42]等网络（表6），但仍属于最先进的结果之列（6.43%，表6）。

层响应分析。图7展示了各层响应的标准差（std）。这些响应是每个3×3层的输出，经过批量归一化（BN）后但在其他非线性操作（ReLU/加法）之前。对于ResNet而言，该分析揭示了残差函数的响应强度。图7显示，ResNet的响应通常比其对应的普通网络更小。这些结果支持了我们的基本动机（第3.1节），即残差函数可能普遍比非残差函数更接近于零。我们还注意到，更深的ResNet具有更小的响应幅度，如图7中ResNet-20、56和110之间的比较所示。当层数更多时，ResNet的单个层倾向于对信号的修改更少。

探索超过1000层的深度模型。我们研究了一个极端深度的模型，层数超过1000层。设定n =200后，网络扩展至1202层，并采用上述方法进行训练。我们的方法展现出*no optimization difficulty*，这一10^3层的网络能够实现*training error*<0.1%的性能（图6右）。其测试误差仍保持较好水平（7.93%，表6）。

但对于如此激进的深度模型，仍存在一些未解决的问题。这个1202层网络的测试结果比我们110层网络的还要差，尽管两者训练误差相近。我们认为这是由于过拟合所致。对于这个小数据集而言，1202层网络可能过大（19.4M参数）。在该数据集上取得最佳结果（[10, 25, 24, 35]）需要应用强正则化方法，如maxout[10]或dropout[14]。本文中，我们未采用maxout/dropout，而是通过设计深层窄架构自然实现正则化，以集中研究优化过程中的难点。未来我们将探索结合更强正则化方法来提升效果。

4.3. 在PASCAL和MS COCO上的目标检测

我们的方法在其他识别任务上展现出良好的泛化性能。表7和表8展示了在PASCAL VOC 2007、2012[5]及COCO[26]数据集上的目标检测基线结果。我们采用*Faster R-CNN*[32]作为检测方法，重点研究了用ResNet-101替换VGG-16[41]带来的提升。由于两种模型的检测实现（详见附录）完全一致，因此性能增益只能归因于更优的网络架构。最显著的是，在极具挑战性的COCO数据集上，我们实现了标准评估指标(mAP@[.5, .95])6.0%的提升，相对改进幅度达28%。这一提升完全源自学习到的表征优势。

基于深度残差网络，我们在ILSVRC和COCO 2015竞赛的多个赛道上斩获第一名：ImageNet检测、ImageNet定位、COCO检测以及COCO分割。具体细节详见附录。

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.

参考文献

[1] Y. Bengio, P. Simard, 和 P. Frasconi. 用梯度下降学习长期依赖关系十分困难. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. [2] C. M. Bishop. *Neural networks for pattern recognition*. 牛津大学出版社, 1995. [3] W. L. Briggs, S. F. McCormick, 等. *A Multigrid Tutorial*. Siam, 2000. [4] K. Chatfield, V. Lempitsky, A. Vedaldi, 和 A. Zisserman. 细节决定成败：近期特征编码方法评估. 载于*BMVC*, 2011. [5] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, 和 A. Zisserman. Pascal视觉对象分类（VOC）挑战赛. *IJCV*, 页码303–338, 2010. [6] S. Gidaris 和 N. Komodakis. 通过多区域与语义分割感知CNN模型进行目标检测. 载于*ICCV*, 2015. [7] R. Girshick. Fast R-CNN. 载于*ICCV*, 2015. [8] R. Girshick, J. Donahue, T. Darrell, 和 J. Malik. 用于精确目标检测与语义分割的丰富特征层次结构. 载于*CVPR*, 2014. [9] X. Glorot 和 Y. Bengio. 理解训练深度前馈神经网络的困难. 载于*AISTATS*, 2010. [10] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, 和 Y. Bengio. Maxout网络. *arXiv:1302.4389*, 2013. [11] K. He 和 J. Sun. 约束时间成本下的卷积神经网络. 载于*CVPR*, 2015. [12] K. He, X. Zhang, S. Ren, 和 J. Sun. 深度卷积网络中用于视觉识别的空间金字塔池化. 载于*ECCV*, 2014. [13] K. He, X. Zhang, S. Ren, 和 J. Sun. 深入探索整流器：超越ImageNet分类的人类水平性能. 载于*ICCV*, 2015. [14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, 和 R. R. Salakhutdinov. 通过防止特征检测器共适应改进神经网络. *arXiv:1207.0580*, 2012. [15] S. Hochreiter 和 J. Schmidhuber. 长短期记忆. *Neural computation*, 9(8):1735–1780, 1997. [16] S. Ioffe 和 C. Szegedy. 批归一化：通过减少内部协变量偏移加速深度网络训练. 载于*ICML*, 2015. [17] H. Jegou, M. Douze, 和 C. Schmid. 最近邻搜索的产品量化. *TPAMI*, 33, 2011. [18] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, 和 C. Schmid. 将局部图像描述符聚合为紧凑编码. *TPAMI*, 2012. [19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, 和 T. Darrell. Caffe：快速特征嵌入的卷积架构. *arXiv:1408.5093*, 2014. [20] A. Krizhevsky. 从微小图像中学习多层特征. *Tech Report*, 2009. [21] A. Krizhevsky, I. Sutskever, 和 G. Hinton. 使用深度卷积神经网络进行ImageNet分类. 载于*NIPS*, 2012. [22] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, 和 L. D. Jackel. 反向传播应用于手写邮政编码识别. *Neural computation*, 1989. [23] Y. LeCun, L. Bottou, G. B. Orr, 和 K.-R. Müller. 高效反向传播. 载于*Neural Networks: Tricks of the Trade*, 页码9–50. Springer, 1998. [24] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, 和 Z. Tu. 深度监督网络. *arXiv:1409.5185*, 2014. [25] M. Lin, Q. Chen, 和 S. Yan. 网络中的网络. *arXiv:1312.4400*, 2013. [26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, 和 C. L. Zitnick. Microsoft COCO：上下文中的常见物体. 载于*ECCV*. 2014. [27] J. Long, E. Shelhamer, 和 T. Darrell. 全卷积网络用于语义分割. 载于*CVPR*, 2015.

[28] G. Montúfar, R. Pascanu, K. Cho, 和 Y. Bengio。论深度神经网络的线性区域数量。载于*NIPS*，2014年。 [29] V. Nair 和 G. E. Hinton。修正线性单元改进受限玻尔兹曼机。载于*ICML*，2010年。 [30] F. Perronnin 和 C. Dance。视觉词汇上的Fisher核用于图像分类。载于*CVPR*，2007年。 [31] T. Raiko, H. Valpola, 和 Y. LeCun。通过感知机中的线性变换简化深度学习。载于*AISTATS*，2012年。 [32] S. Ren, K. He, R. Girshick, 和 J. Sun。Faster R-CNN：利用区域提议网络实现实时目标检测。载于*NIPS*，2015年。 [33] S. Ren, K. He, R. Girshick, X. Zhang, 和 J. Sun。卷积特征图上的目标检测网络。*arXiv:1504.06066*，2015年。 [34] B. D. Ripley。*Pattern recognition and neural networks*。剑桥大学出版社，1996年。 [35] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, 和 Y. Bengio。Fitnets：瘦深度网络的提示。载于*ICLR*，2015年。 [36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein等。ImageNet大规模视觉识别挑战赛。*arXiv:1409.0575*，2014年。 [37] A. M. Saxe, J. L. McClelland, 和 S. Ganguli。深度线性神经网络学习非线性动力学的精确解。*arXiv:1312.6120*，2013年。 [38] N. N. Schraudolph。基于因子中心分解的加速梯度下降。技术报告，1998年。 [39] N. N. Schraudolph。神经网络梯度因子的中心化。载于*Neural Networks: Tricks of the Trade*，第207–226页。Springer，1998年。 [40] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, 和 Y. LeCun。Overfeat：使用卷积网络集成识别、定位与检测。载于*ICLR*，2014年。 [41] K. Simonyan 和 A. Zisserman。超深卷积网络用于大规模图像识别。载于*ICLR*，2015年。 [42] R. K. Srivastava, K. Greff, 和 J. Schmidhuber。高速公路网络。*arXiv:1505.00387*，2015年。 [43] R. K. Srivastava, K. Greff, 和 J. Schmidhuber。训练极深度网络。*1507.06228*，2015年。 [44] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, 和 A. Rabinovich。深入卷积。载于*CVPR*，2015年。 [45] R. Szeliski。使用分层基函数的快速表面插值。*TPAMI*，1990年。 [46] R. Szeliski。局部适应的分层基预条件。载于*SIGGRAPH*，2006年。 [47] T. Vatanen, T. Raiko, H. Valpola, 和 Y. LeCun。将随机梯度推向二阶方法——非线性变换的反向传播学习。载于*Neural InformationProcessing*，2013年。 [48] A. Vedaldi 和 B. Fulkerson。VLFeat：计算机视觉算法的开放便携库，2008年。 [49] W. Venables 和 B. Ripley。现代应用统计与S-PLUS。1999年。 [50] M. D. Zeiler 和 R. Fergus。可视化与理解卷积神经网络。载于*ECCV*，2014年。

A. Object Detection Baselines

In this section we introduce our detection method based on the baseline Faster R-CNN [32] system. The models are initialized by the ImageNet classification models, and then fine-tuned on the object detection data. We have experimented with ResNet-50/101 at the time of the ILSVRC & COCO 2015 detection competitions.

Unlike VGG-16 used in [32], our ResNet has no hidden fc layers. We adopt the idea of "Networks on Conv feature maps" (NoC) [33] to address this issue. We compute the full-image shared conv feature maps using those layers whose strides on the image are no greater than 16 pixels (*i.e.*, conv1, conv2_x, conv3_x, and conv4_x, totally 91 conv layers in ResNet-101; Table 1). We consider these layers as analogous to the 13 conv layers in VGG-16, and by doing so, both ResNet and VGG-16 have conv feature maps of the same total stride (16 pixels). These layers are shared by a region proposal network (RPN, generating 300 proposals) [32] and a Fast R-CNN detection network [7]. RoI pooling [7] is performed before conv5_1. On this RoI-pooled feature, all layers of conv5_x and up are adopted for each region, playing the roles of VGG-16's fc layers. The final classification layer is replaced by two sibling layers (classification and box regression [7]).

For the usage of BN layers, after pre-training, we compute the BN statistics (means and variances) for each layer on the ImageNet training set. Then the BN layers are fixed during fine-tuning for object detection. As such, the BN layers become linear activations with constant offsets and scales, and BN statistics are not updated by fine-tuning. We fix the BN layers mainly for reducing memory consumption in Faster R-CNN training.

PASCAL VOC

Following [7, 32], for the PASCAL VOC 2007 *test* set, we use the 5k *trainval* images in VOC 2007 and 16k *trainval* images in VOC 2012 for training ("07+12"). For the PASCAL VOC 2012 *test* set, we use the 10k *trainval+test* images in VOC 2007 and 16k *trainval* images in VOC 2012 for training ("07++12"). The hyper-parameters for training Faster R-CNN are the same as in [32]. Table 7 shows the results. ResNet-101 improves the mAP by >3% over VGG-16. This gain is solely because of the improved features learned by ResNet.

MS COCO

The MS COCO dataset [26] involves 80 object categories. We evaluate the PASCAL VOC metric (mAP @ IoU = 0.5) and the standard COCO metric (mAP @ IoU = .5:.05:.95). We use the 80k images on the train set for training and the 40k images on the val set for evaluation. Our detection system for COCO is similar to that for PASCAL VOC. We train the COCO models with an 8-GPU implementation, and thus the RPN step has a mini-batch size of 8 images (*i.e.*, 1 per GPU) and the Fast R-CNN step has a mini-batch size of 16 images. The RPN step and Fast R-CNN step are both trained for 240k iterations with a learning rate of 0.001 and then for 80k iterations with 0.0001.

Table 8 shows the results on the MS COCO validation set. ResNet-101 has a 6% increase of mAP@[.5, .95] over VGG-16, which is a 28% relative improvement, solely contributed by the features learned by the better network. Remarkably, the mAP@[.5, .95]'s absolute increase (6.0%) is nearly as big as mAP@.5's (6.9%). This suggests that a deeper network can improve both recognition and localization.

B. Object Detection Improvements

For completeness, we report the improvements made for the competitions. These improvements are based on deep features and thus should benefit from residual learning.

MS COCO

Box refinement. Our box refinement partially follows the iterative localization in [6]. In Faster R-CNN, the final output is a regressed box that is different from its proposal box. So for inference, we pool a new feature from the regressed box and obtain a new classification score and a new regressed box. We combine these 300 new predictions with the original 300 predictions. Non-maximum suppression (NMS) is applied on the union set of predicted boxes using an IoU threshold of 0.3 [8], followed by box voting [6]. Box refinement improves mAP by about 2 points (Table 9).

Global context. We combine global context in the Fast R-CNN step. Given the full-image conv feature map, we pool a feature by global Spatial Pyramid Pooling [12] (with a "single-level" pyramid) which can be implemented as "RoI" pooling using the entire image's bounding box as the RoI. This pooled feature is fed into the post-RoI layers to obtain a global context feature. This global feature is concatenated with the original per-region feature, followed by the sibling classification and box regression layers. This new structure is trained end-to-end. Global context improves mAP@.5 by about 1 point (Table 9).

Multi-scale testing. In the above, all results are obtained by single-scale training/testing as in [32], where the image's shorter side is $s = 600$ pixels. Multi-scale training/testing has been developed in [12, 7] by selecting a scale from a feature pyramid, and in [33] by using maxout layers. In our current implementation, we have performed multi-scale *testing* following [33]; we have not performed multi-scale training because of limited time. In addition, we have performed multi-scale testing only for the Fast R-CNN step (but not yet for the RPN step). With a trained model, we compute conv feature maps on an image pyramid, where the image's shorter sides are $s \in \{200, 400, 600, 800, 1000\}$.

A. 目标检测基线

在本节中，我们介绍基于基准Faster R-CNN[32]系统的检测方法。模型以ImageNet分类模型进行初始化，随后在目标检测数据上进行微调。在ILSVRC & COCO 2015检测竞赛期间，我们尝试了ResNet-50/101架构。

与[32]中使用的VGG-16不同，我们的ResNet没有隐藏的全连接层。我们采用“卷积特征图上的网络”（NoC）[33]的思想来解决这一问题。我们利用那些在图像上步幅不超过16像素的层（*i.e*，如conv1、conv2_x、conv3_x和conv4_x，在ResNet-101中共计91个卷积层；见表1）来计算全图共享的卷积特征图。我们将这些层视为与VGG-16中的13个卷积层类似，这样ResNet和VGG-16的卷积特征图具有相同的总步幅（16像素）。这些层由区域提议网络（RPN，生成300个提议）[32]和Fast R-CNN检测网络[7]共享。在conv5_1之前执行RoI池化[7]。在这个RoI池化后的特征上，conv5_x及更高层的所有层被用于每个区域，扮演了VGG-16中全连接层的角色。最终的分类层被替换为两个并列的层（分类和边界框回归[7]）。

关于BN层的使用，在预训练完成后，我们会在ImageNet训练集上计算每一层的BN统计量（均值与方差）。随后，在目标检测的微调阶段，这些BN层将被固定。如此一来，BN层便转化为带有恒定偏移和缩放系数的线性激活函数，其统计量也不会通过微调进行更新。我们固定BN层主要是为了减少Faster R-CNN训练过程中的内存消耗。

PASCAL VOC

遵循[7, 32]的做法，对于PASCAL VOC 2007 *test*数据集，我们采用VOC 2007中的5k *trainval*图像和VOC 2012中的16k *train-val*图像进行训练（“07+12”）。针对PASCAL VOC 2012 *test*数据集，则使用VOC 2007的10k *trainval*+*test*图像与VOC 2012的16k *trainval*图像进行训练（“07++12”）。训练Faster R-CNN的超参数设置与[32]保持一致。表7展示了实验结果。ResNet-101相较VGG-16将mAP提升了>3%，这一提升完全得益于ResNet学习到的更优特征。

MS COCO

MS COCO数据集[26]包含80个物体类别。我们采用PASCAL VOC评估指标（mAP @ IoU = 0.5）和标准COCO评估指标（mAP @ IoU = .5:.05:.95）。训练阶段使用训练集中的80k图像，评估阶段则使用验证集中的40k图像。针对COCO的检测系统与PASCAL VOC方案类似。我们采用8-GPU配置训练COCO模型，因此RPN阶段的小批量大小为8张图像（*i.e*，每GPU 1张），Fast R-CNN步骤的小批量大小为16张图像。RPN步骤和Fast R-CNN步骤均训练240k次迭代，学习率为0.001，随后再以0.0001的学习率训练80k次迭代。

表8展示了在MS COCO验证集上的结果。ResNet-101相较于VGG-16在mAP@[.5, .95]上提升了6%，即相对提升了28%，这完全归功于更优网络所学习到的特征。值得注意的是，mAP@[.5, .95]的绝对提升（6.0%）几乎与mAP@.5的提升（6.9%）相当。这表明更深的网络能够同时改进识别和定位性能。

B. 目标检测改进

为了完整性，我们报告了为竞赛所做的改进。这些改进基于深度特征，因此应能从残差学习中获益。

MS COCO

Box refinement. 我们的框优化部分遵循了[6]中的迭代定位方法。在Faster R-CNN中，最终输出是一个与提议框不同的回归框。因此，在推理阶段，我们会从回归框中提取新的特征，获得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。对预测框的并集采用0.3的交并比(IoU)阈值进行非极大值抑制(NMS)[8]，随后进行框投票[6]。框优化使mAP提升了约2个百分点（表9）。

Global context. 我们在Fast R-CNN步骤中融入了全局上下文信息。给定全图卷积特征图后，我们通过全局空间金字塔池化[12]（采用“单层”金字塔结构）提取特征，该操作可视为将整幅图像的边界框作为RoI进行“RoI池化”。池化后的特征被送入RoI后置层，以获取全局上下文特征。这一全局特征随后与原始的各区域特征拼接，再接续兄弟分类层和边界框回归层。整个新结构采用端到端训练方式。引入全局上下文使mAP@.5提升了约1个百分点（表9）。

Multi-scale testing. 在上述过程中，所有结果均采用如[32]中所述的单尺度训练/测试方法获得，其中图像的短边设定为$s=600$像素。多尺度训练/测试技术已在[12,7]中通过从特征金字塔选择尺度，以及在[33]中通过使用maxout层得到发展。在当前实现中，我们依据[33]进行了多尺度*testing*处理；由于时间有限，尚未实施多尺度训练。此外，我们仅在Fast R-CNN步骤（尚未在RPN步骤中）执行了多尺度测试。针对训练好的模型，我们在图像金字塔上计算卷积特征图，其中图像的短边分别为$s\in\{200,、400,、600,、800,和1000\}$像素。

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. Object detection improvements on MS COCO using Faster R-CNN and ResNet-101.

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. Detection results on the PASCAL VOC 2007 test set. The baseline is the Faster R-CNN system. The system "baseline+++" include box refinement, context, and multi-scale testing in Table 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. Detection results on the PASCAL VOC 2012 test set (`http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=4`). The baseline is the Faster R-CNN system. The system "baseline+++" include box refinement, context, and multi-scale testing in Table 9.

We select two adjacent scales from the pyramid following [33]. RoI pooling and subsequent layers are performed on the feature maps of these two scales [33], which are merged by maxout as in [33]. Multi-scale testing improves the mAP by over 2 points (Table 9).

Using validation data. Next we use the 80k+40k trainval set for training and the 20k test-dev set for evaluation. The test-dev set has no publicly available ground truth and the result is reported by the evaluation server. Under this setting, the results are an mAP@.5 of 55.7% and an mAP@[.5, .95] of 34.9% (Table 9). This is our single-model result.

Ensemble. In Faster R-CNN, the system is designed to learn region proposals and also object classifiers, so an ensemble can be used to boost both tasks. We use an ensemble for proposing regions, and the union set of proposals are processed by an ensemble of per-region classifiers. Table 9 shows our result based on an ensemble of 3 networks. The mAP is 59.0% and 37.4% on the test-dev set. *This result won the 1st place in the detection task in COCO 2015.*

PASCAL VOC

We revisit the PASCAL VOC dataset based on the above model. With the single model on the COCO dataset (55.7% mAP@.5 in Table 9), we fine-tune this model on the PASCAL VOC sets. The improvements of box refinement, context, and multi-scale testing are also adopted. By doing so we achieve 85.6% mAP on PASCAL VOC 2007 (Table 10) and 83.8% on PASCAL VOC 2012 (Table 11)[6]. The result on PASCAL VOC 2012 is 10 points higher than the previous state-of-the-art result [6].

	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. Our results (mAP, %) on the ImageNet detection dataset. Our detection system is Faster R-CNN [32] with the improvements in Table 9, using ResNet-101.

ImageNet Detection

The ImageNet Detection (DET) task involves 200 object categories. The accuracy is evaluated by mAP@.5. Our object detection algorithm for ImageNet DET is the same as that for MS COCO in Table 9. The networks are pre-trained on the 1000-class ImageNet classification set, and are fine-tuned on the DET data. We split the validation set into two parts (val1/val2) following [8]. We fine-tune the detection models using the DET training set and the val1 set. The val2 set is used for validation. We do not use other ILSVRC 2015 data. Our single model with ResNet-101 has

[6] `http://host.robots.ox.ac.uk:8080/anonymous/3OJ4OJ.html`, submitted on 2015-11-26.

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**

表9. 使用Faster R-CNN和ResNet-101在MS COCO数据集上的目标检测改进。

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**

表10. PASCAL VOC 2007测试集上的检测结果。基准系统为Faster R-CNN。"baseline+++"系统包含了表9中的边界框优化、上下文信息以及多尺度测试。

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**

表11. PASCAL VOC 2012测试集上的检测结果（http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=4）。基准系统为Faster R-CNN。表9中“baseline+++”系统包含边界框优化、上下文信息及多尺度测试。

我们按照[33]的方法从金字塔中选取两个相邻的尺度。在这两个尺度的特征图上进行RoI池化及后续层操作，并如[33]所述采用maxout方式进行融合。多尺度测试使mAP提升了超过2个百分点（表9）。

Using validation data. 接下来，我们使用80k+40k的训练验证集进行训练，并以20k的测试开发集进行评估。测试开发集没有公开可用的基准真值，结果由评估服务器上报。在此设置下，我们得到的mAP@.5为55.7%，mAP@[.5, .95]为34.9%（表9）。这是我们的单模型结果。

Ensemble. 在Faster R-CNN中，系统被设计为同时学习区域提议和物体分类器，因此可以采用集成方法来提升这两项任务的性能。我们使用一个集成模型来生成区域提议，然后通过一组针对每个区域的分类器集成处理这些提议的并集。表9展示了基于3个网络集成后的结果，在测试开发集上的mAP分别达到59.0%和37.4%。*This result won the 1st place in the detection task in COCO 2015.*

PASCAL VOC

我们基于上述模型重新审视了PASCAL VOC数据集。利用在COCO数据集上的单一模型（表9中55.7%的mAP@.5指标），我们在PASCAL VOC数据集上对该模型进行了微调。同时采用了边界框优化、上下文信息以及多尺度测试等改进措施。通过这一系列操作

	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**

表12. 我们在ImageNet检测数据集上的结果（mAP，%）。我们的检测系统采用Faster R-CNN[32]并结合表9中的改进，使用ResNet-101作为基础网络。

我们在PASCAL VOC 2007上达到了85.6%的mAP（表10），在PASCAL VOC 2012上达到了83.8%（表11）[6]。这一结果比之前的最先进水平[6]高出10个百分点。

ImageNet 检测

ImageNet检测（DET）任务涵盖200个物体类别，其精度通过mAP@.5进行评估。我们在ImageNet DET上采用的物体检测算法与表9中MS COCO所使用的相同。网络首先在包含1000个类别的ImageNet分类集上进行预训练，随后在DET数据上进行微调。按照文献[8]的方法，我们将验证集划分为两部分（val1/val2）。检测模型使用DET训练集和val1集进行微调，val2集则用于验证。未使用其他ILSVRC 2015数据。我们采用ResNet-101的单模型达到了

[6] http://host.robots.ox.ac.uk:8080/anonymous/3OJ4OJ.html, submitted on 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. Localization error (%) on the ImageNet validation. In the column of "LOC error on GT class" ([41]), the ground truth class is used. In the "testing" column, "1-crop" denotes testing on a center crop of 224×224 pixels, "dense" denotes dense (fully convolutional) and multi-scale testing.

58.8% mAP and our ensemble of 3 models has 62.1% mAP on the DET test set (Table 12). *This result won the 1st place in the ImageNet detection task in ILSVRC 2015*, surpassing the second place by **8.5 points** (absolute).

C. ImageNet Localization

The ImageNet Localization (LOC) task [36] requires to classify and localize the objects. Following [40, 41], we assume that the image-level classifiers are first adopted for predicting the class labels of an image, and the localization algorithm only accounts for predicting bounding boxes based on the predicted classes. We adopt the "per-class regression" (PCR) strategy [40, 41], learning a bounding box regressor for each class. We pre-train the networks for ImageNet classification and then fine-tune them for localization. We train networks on the provided 1000-class ImageNet training set.

Our localization algorithm is based on the RPN framework of [32] with a few modifications. Unlike the way in [32] that is category-agnostic, our RPN for localization is designed in a *per-class* form. This RPN ends with two sibling 1×1 convolutional layers for binary classification (*cls*) and box regression (*reg*), as in [32]. The *cls* and *reg* layers are both in a *per-class* from, in contrast to [32]. Specifically, the *cls* layer has a 1000-d output, and each dimension is *binary logistic regression* for predicting being or not being an object class; the *reg* layer has a 1000×4-d output consisting of box regressors for 1000 classes. As in [32], our bounding box regression is with reference to multiple translation-invariant "anchor" boxes at each position.

As in our ImageNet classification training (Sec. 3.4), we randomly sample 224×224 crops for data augmentation. We use a mini-batch size of 256 images for fine-tuning. To avoid negative samples being dominate, 8 anchors are randomly sampled for each image, where the sampled positive and negative anchors have a ratio of 1:1 [32]. For testing, the network is applied on the image fully-convolutionally.

Table 13 compares the localization results. Following [41], we first perform "oracle" testing using the ground truth class as the classification prediction. VGG's paper [41] reports a center-crop error of 33.1% (Table 13) using ground truth classes. Under the same setting, our RPN method using ResNet-101 net significantly reduces the center-crop error to 13.3%. This comparison demonstrates the excellent performance of our framework. With dense (fully convolutional) and multi-scale testing, our ResNet-101 has an error of 11.7% using ground truth classes. Using ResNet-101 for predicting classes (4.6% top-5 classification error, Table 4), the top-5 localization error is 14.4%.

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. Comparisons of localization error (%) on the ImageNet dataset with state-of-the-art methods.

The above results are only based on the *proposal network* (RPN) in Faster R-CNN [32]. One may use the *detection network* (Fast R-CNN [7]) in Faster R-CNN to improve the results. But we notice that on this dataset, one image usually contains a single dominate object, and the proposal regions highly overlap with each other and thus have very similar RoI-pooled features. As a result, the image-centric training of Fast R-CNN [7] generates samples of small variations, which may not be desired for stochastic training. Motivated by this, in our current experiment we use the original R-CNN [8] that is RoI-centric, in place of Fast R-CNN.

Our R-CNN implementation is as follows. We apply the per-class RPN trained as above on the training images to predict bounding boxes for the ground truth class. These predicted boxes play a role of class-dependent proposals. For each training image, the highest scored 200 proposals are extracted as training samples to train an R-CNN classifier. The image region is cropped from a proposal, warped to 224×224 pixels, and fed into the classification network as in R-CNN [8]. The outputs of this network consist of two sibling fc layers for *cls* and *reg*, also in a per-class form. This R-CNN network is fine-tuned on the training set using a mini-batch size of 256 in the RoI-centric fashion. For testing, the RPN generates the highest scored 200 proposals for each predicted class, and the R-CNN network is used to update these proposals' scores and box positions.

This method reduces the top-5 localization error to 10.6% (Table 13). This is our single-model result on the validation set. Using an ensemble of networks for both classification and localization, we achieve a top-5 localization error of 9.0% on the test set. This number significantly outperforms the ILSVRC 14 results (Table 14), showing a 64% relative reduction of error. *This result won the 1st place in the ImageNet localization task in ILSVRC 2015.*

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**

表13. ImageNet验证集上的定位误差（%）。在“GT类别上的定位误差”（[41]）一列中，使用了真实类别。在“测试”列中，“1-crop”表示在224×224像素的中心裁剪上进行测试，“dense”表示密集（全卷积）及多尺度测试。

58.8%的mAP，而我们的3个模型集成在DET测试集上达到了62.1%的mAP（表12）。
This result won the 1st place in the ImageNet detection task in ILSVRC 2015，以8.5个百分点的绝对优势超越第二名。

C. ImageNet 定位

ImageNet定位（LOC）任务[36]要求对物体进行分类与定位。遵循[40, 41]的方法，我们假设首先采用图像级分类器预测图像的类别标签，而定位算法仅负责基于预测类别生成边界框。我们采用“每类回归”（PCR）策略[40, 41]，为每个类别学习一个边界框回归器。网络先在ImageNet分类任务上进行预训练，随后针对定位任务进行微调。训练过程使用提供的1000类ImageNet训练集完成。

我们的定位算法基于[32]中的RPN框架，并进行了若干改进。与[32]中类别无关的设计不同，我们的定位RPN采用*per-class*形式构建。如[32]所示，该RPN末端设有两个并行的1×1卷积层，分别用于二元分类（*cls*）和边界框回归（*reg*）。与[32]形成对比的是，*cls*层和*reg*层均采用*per-class*形式。具体而言，*cls*层输出1000维向量，每个维度*binary logistic regression*用于预测是否属于某物体类别；*reg*层则输出1000×4维向量，包含针对1000个类别的边界框回归器。与[32]一致，我们的边界框回归参照了每个位置上多个具有平移不变性的"锚框"。

与我们的ImageNet分类训练（见第3.4节）类似，我们随机采样224×224的裁剪区域进行数据增强。微调时采用256张图像的小批量大小。为防止负样本占据主导，每张图像随机采样8个锚点，其中正负锚点的采样比例为1:1[32]。测试时，网络以全卷积方式应用于整张图像。

表13对比了定位结果。遵循[41]的方法，我们首先使用真实类别作为分类预测进行“oracle”测试。VGG的论文[41]中

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**

表14. 在ImageNet数据集上与最先进方法的定位误差(%)比较。

使用真实类别时，中心裁剪误差报告为33.1%（表13）。在相同设置下，我们采用ResNet-101网络的RPN方法将中心裁剪误差显著降低至13.3%。这一对比证明了我们框架的卓越性能。通过密集（全卷积）和多尺度测试，我们的ResNet-101在使用真实类别时误差降至11.7%。当采用ResNet-101进行类别预测（top-5分类误差4.6%，见表4）时，top-5定位误差为14.4%。

上述结果仅基于Faster R-CNN[32]中的*proposal network*(RPN)。有人可能会尝试在Faster R-CNN中使用*detectionnetwork* (Fast R-CNN[7])来提升效果。但我们注意到，在该数据集中，单张图像通常仅包含一个主导物体，且候选区域间高度重叠，导致RoI池化后的特征极为相似。因此，Fast R-CNN[7]以图像为中心的训练方式生成的样本变异较小，这可能不利于随机训练。基于此，当前实验中我们采用以RoI为中心的原始R-CNN[8]替代Fast R-CNN。

我们的R-CNN实现步骤如下。首先，在训练图像上应用上述训练得到的类别相关RPN（区域提议网络），以预测真实类别对应的边界框。这些预测框充当了类别相关的候选区域。对于每张训练图像，我们提取得分最高的200个候选区域作为训练样本，用于训练R-CNN分类器。图像区域从候选框中裁剪后，被调整为224×224像素大小，并如同原始R-CNN[8]那样输入分类网络。该网络输出包含两个并列的全连接层，分别对应*cls*和*reg*，同样以逐类别形式组织。此R-CNN网络采用RoI-centric方式在训练集上进行微调，mini-batch大小为256。测试阶段，RPN为每个预测类别生成得分最高的200个候选框，R-CNN网络则用于更新这些候选框的得分和位置信息。

该方法将前五定位误差降至10.6%（表13）。这是我们在验证集上的单一模型结果。通过使用网络集成同时进行分类与定位，我们在测试集上实现了9.0%的前五定位误差。这一数值显著超越了ILSVRC 14的结果（表14），显示出误差相对减少了64%。
This result won the 1st place in the ImageNet localization task in ILSVRC 2015.