Aggregated Residual Transformations for Deep Neural Networks

Saining Xie[1] Ross Girshick[2] Piotr Dollár[2] Zhuowen Tu[1] Kaiming He[2]
[1]UC San Diego [2]Facebook AI Research
{s9xie,ztu}@ucsd.edu {rbg,pdollar,kaiminghe}@fb.com

Abstract

提示：该*PDF*由*SpeedPaper*生成，版权归原文作者所有。翻译内容仅供参考，请仔细鉴别并以原文为准。

查看更多论文翻译与复现代码：*https://github.com/hanknewbird/SpeedPaper*

我们提出了一个用于图像分类的简单、高度模块化的网络架构。我们的网络是通过重复一个聚合了一组具有相同拓扑结构的变换的构建块来构建的。我们的简单设计导致了一个均匀的、多分支的架构，只需设置少量超参数。这种策略展现出了一个新的维度，我们称之为“基数”（变换集的大小），作为深度和宽度之外的一个关键因素。在*ImageNet-1K*数据集上，我们经验证明，在保持复杂性的受限条件下，增加基数能够提高分类准确性。此外，增加基数在增加容量时比加深或加宽更有效。我们的模型名为*ResNeXt*，是我们参加*ILSVRC 2016*分类任务并获得第二名的基础。我们在*ImageNet-5K*数据集和*COCO*检测数据集上进一步研究了*ResNeXt*，同样显示出比其*ResNet*对应物更好的结果。代码和模型可在网上公开获取[1]。

1. Introduction

视觉识别的研究正经历从“特征工程”转向“网络工程”的转变[25, 24, 44, 34, 36, 38, 14]。与传统的手工设计特征（例如，SIFT[29]和HOG[5]）不同，由神经网络从大规模数据中学习到的特征[33]在训练过程中需要较少的人类参与，并可以迁移到各种识别任务中[7, 10, 28]。然而，人类的努力已经转移到设计更好的网络架构来学习表示。

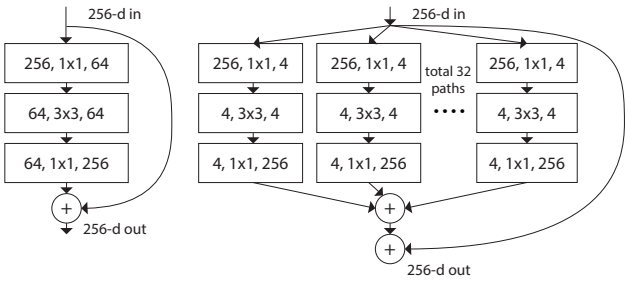

Figure 1. **左侧：**一个ResNet块[14]。 **右侧：**一个具有基数 = 32 的ResNeXt块，复杂度大致相同。一个图层显示为（输入通道数，滤波器尺寸，输出通道数）。

随着超参数数量的增加（宽度[2]、滤波器大小、步长等），特别是在有许多层的情况下，设计架构变得越来越困难。VGG-nets[36]展示了一种简单而有效的策略，构建非常深的网络：堆叠相同形状的构件块。这一策略被ResNets[14]所继承，其堆叠相同拓扑结构的模块。这一简单规则减少了超参数的自由选择，并且深度暴露为神经网络中的一个关键维度。此外，我们认为这一规则的简洁性可能降低了过分适应特定数据集的风险。VGG-nets和ResNets的鲁棒性已通过各种视觉识别任务（如[7, 10, 9, 28, 31, 14]）以及涉及语音[42, 30]和语言[4, 41, 20]的非视觉任务得到验证。

与VGG网络不同，Inception模型家族[38, 17, 39, 37]已经证明，精心设计的拓扑结构能够在低理论复杂性的情况下实现引人注目的准确性。Inception模型随着时间的推移发生了演变[38, 39]，但一个重要的共同特性是分裂-变换-融合策略。在Inception模块中，输入被拆分为几个低维度嵌入（通过1×1卷积），通过一组专门的滤波器（3×3，5×5，等）进行变换，然后通过串联进行合并。

[1]https://github.com/facebookresearch/ResNeXt

[2]宽度指的是层中的通道数。

可以证明，这种架构的解决空间是单个大层（如5×5）在高维嵌入上操作的解决空间的严格子空间。Inception模块的分裂-变换-融合行为预期能够接近大而密集层的表示能力，但其计算复杂度要低得多。

尽管准确性很高，Inception模型的实现伴随着一系列复杂因素 — 每个转换的滤波器数量和大小都是为每个单独转换定制的，模块是逐个阶段定制的。尽管对这些组件的仔细组合能产生出色的神经网络模型，但一般来说，如何将Inception架构调整到新的数据集/任务尚不清楚，特别是在需要设计许多因素和超参数的情况下。

在本文中，我们提出了一种简单的架构，该架构采用了VGG/ResNet重复层的策略，同时以一种简单、可扩展的方式利用了分裂-变换-融合策略。我们网络中的一个模块执行一组转换，每个转换都在低维度嵌入上进行，其输出通过求和进行聚合。我们追求这种思想的简单实现 — 要聚合的转换都具有相同的拓扑结构（如图1(右)所示）。该设计允许我们扩展到任意数量的转换，而无需专门设计。

有趣的是，在这种简化情况下，我们展示了我们的模型还有其他两种等效形式（如图3所示）。图3(b)中的改组形式似乎类似于Inception-ResNet模块[37]，因为它串联了多个路径；但我们的模块与所有现有的Inception模块不同之处在于，我们所有的路径都共享相同的拓扑结构，因此路径的数量可以很容易地被单独列为要研究的因素。在更简明的改组中，我们的模块可以通过Krizhevsky等人的分组卷积[24]（如图3(c)所示）进行重塑，然而，这种方法被开发为一种工程妥协。

我们通过实验证明，我们聚合的转换在保持计算复杂度和模型大小不变的受限条件下，性能优于原始的ResNet模块 — 例如，图1(右)旨在保持与图1(左)相同的FLOPs复杂度和参数数量。我们强调，尽管通过增加容量（加深或加宽）相对容易提高准确性，但文献中很少有方法能够在保持（或减少）复杂度的情况下提高准确性。

我们的方法表明基数（转换集的大小）是一个具体的、可测量的维度，除了宽度和深度的维度之外，还具有重要意义。实验证明，增加基数比加深或加宽更有效地提高准确性，特别是当深度和宽度开始对现有模型产生递减收益时。我们的神经网络被命名为*ResNeXt*（暗示着*next*维度），在ImageNet分类数据集上的表现优于ResNet-101/152 [14]、ResNet-200 [15]、Inception-v3 [39]和Inception-ResNet-v2 [37]。特别地，一种包含101层的ResNeXt能够比ResNet-200 [15]取得更佳的准确性，但复杂度仅为50%。此外，ResNeXt比所有Inception模型都具有明显更简单的设计。 ResNeXt是我们在ILSVRC 2016分类任务中获得第二名的基础。本文进一步对ResNeXt在更大的ImageNet-5K数据集和COCO目标检测数据集 [27]上进行评估，结果表明其准确性一贯优于ResNet同类模型。我们预计ResNeXt也将对其他视觉（和非视觉）识别任务具有良好的泛化性。

2. Related Work

多分支卷积网络。Inception模型[38, 17, 39, 37]是成功的多分支架构，其中每个分支都经过精心定制。ResNets [14]可以被看作是两分支网络，其中一个分支是恒等映射。深度神经决策森林[22]是具有学习分裂函数的树状多分支网络。

分组卷积. 使用分组卷积可以追溯至AlexNet论文 [24]，甚至更早。Krizhevsky等人 [24] 给出的动机是为了将模型分布在两个GPU上。分组卷积得到了Caffe [19]、Torch [3]和其他主要用于AlexNet兼容性的库的支持。据我们所知，目前很少有证据表明利用分组卷积可以提高准确性。分组卷积的一个特殊情况是通道级卷积，其中分组数等于通道数。通道级卷积是 [35] 中可分卷积的一部分。

压缩卷积网络。在空间[6, 18]和/或通道[6, 21, 16]层面上的分解是一种广泛采用的技术，用于减少深度卷积网络的冗余并加速/压缩它们。Ioannou等人[16]提出了一种用于减少计算量的“根”模式网络，根部的分支通过分组卷积实现。这些方法[6, 18, 21, 16]已经表现出在较低复杂度和更小模型尺寸下与更高准确性的优雅折衷。与压缩不同，我们的方法是一种在实证上展示更强表示能力的架构。

集成。将一组独立训练的网络进行平均是提高准确性的有效解决方案 [24]，被广泛采用于识别竞赛中 [33]。Veit等人 [40] 将单个ResNet解释为一组更浅的网络的集成，这是由ResNet的加法行为所导致的 [15]。我们的方法利用加法来聚合一组转换。但我们认为将我们的方

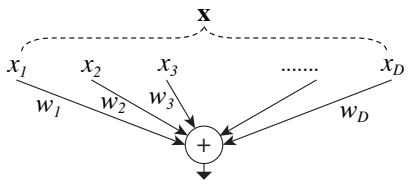

Figure 2. 一个执行内积的简单神经元。

法视为集成是不准确的，因为要聚合的成员是联合训练的，而不是独立训练的。

3. Method

3.1. Template

我们采用高度模块化设计，遵循VGG/ResNets。我们的网络由一系列残差块组成。这些块具有相同的拓扑结构，并受到两个受到VGG/ResNets启发的简单规则的约束：(i) 如果产生大小相同的空间映射，那么这些块共享相同的超参数（宽度和滤波器尺寸），并且(ii)每当空间映射被下采样2倍时，块的宽度就会乘以2倍。第二个规则确保在所有块中，计算复杂度（以FLOPs计算，即乘加操作的数量）大致相同。

有了这两个规则，我们只需要设计一个模板模块，然后就可以相应确定网络中的所有模块。因此，这两个规则极大地缩小了设计空间，使我们能够专注于少数关键因素。根据这些规则构建的网络在表 1.

3.2. Revisiting Simple Neurons

人工神经网络中最简单的神经元执行内积（加权和），这是由全连接层和卷积层完成的基本变换。内积可以看作是一种聚合变换形式：

$$\sum_{i=1}^{D} w_i x_i, \tag{1}$$

其中，$\mathbf{x} = [x_1, x_2, ..., x_D]$ 是传递给神经元的 D 通道输入向量，w_i 是第 i 通道的滤波器权重。这个操作（通常包括一些输出非线性），被称为“神经元”。请参见图 2。

上述操作可以重新表述为*拆分*、*转换*和*聚合*的组合。 (i) *拆分*：向量 $\mathbf{x}$ 被切片成低维嵌入，在上述示例中，它是一个单维子空间 x_i。 (ii) *转换*：低维表示被转换，上文中简单地经过缩放：$w_i x_i$。 (iii) *聚合*：所有嵌入的变换被 $\sum_{i=1}^{D}$ 聚合。

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**×10^6	**25.0**×10^6
FLOPs		**4.1**×10^9	**4.2**×10^9

Table 1. （**左侧**） ResNet-50。（**右侧**） 具有32×4d模板的ResNeXt-50（使用图 3(c)中的重新构造）。方括号内是残差块的形状，方括号外是在一个阶段上堆叠的块的数量。“C=32” 表示带有32组的分组卷积[24]。 *这两个模型之间的参数数量和FLOPs数相似。*

3.3. Aggregated Transformations

考虑到上述对简单神经元的分析，我们考虑用一个更通用的函数来替代基本变换（$w_i x_i$），这个函数本身也可以是一个网络。与“网络中的网络”[26]不同，后者会增加深度维度，我们展示了我们的“神经元中的网络”沿着一个新维度进行扩展。

形式上，我们将汇总的变换表示为：

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{C} \mathcal{T}_i(\mathbf{x}), \tag{2}$$

其中$\mathcal{T}_i(\mathbf{x})$可以是任意函数。类似于简单的神经元，$\mathcal{T}_i$应该将$\mathbf{x}$投影到一个（可选的低维）嵌入中，然后进行转换。

在Eqn.(2)中，C是要聚合的转换集的大小。我们将C称为*基数* [2]。在Eqn.(2)中，C的位置类似于Eqn.(1)中的D，但C不需要等于D，可以是任意数。虽然宽度维度与简单转换的数量（内积）相关，但我

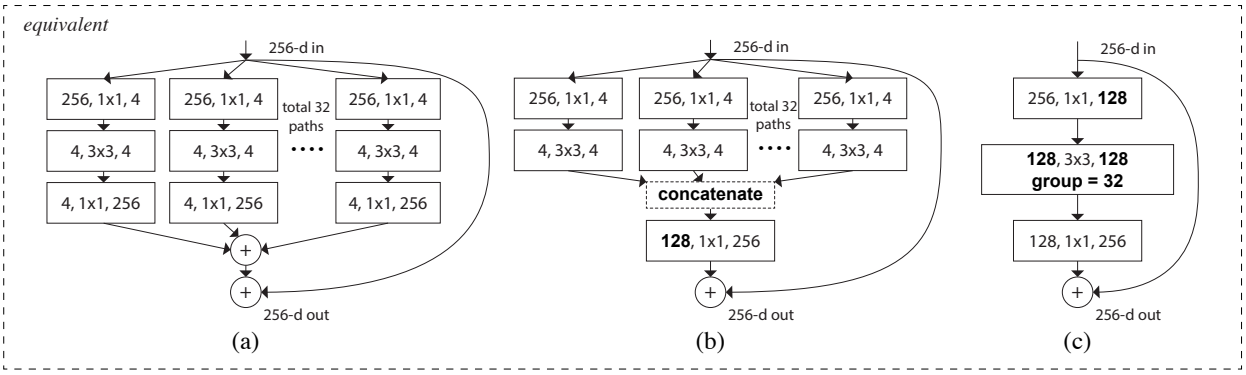

Figure 3. ResNeXt 的等价构建模块。 **(a)**: 聚合残差变换，与图 1 的右侧相同。 **(b)**: 一个相当于 (a) 的模块，以早期串联实现。 **(c)**: 一个相当于 (a, b) 的模块，以分组卷积 [24] 实现。在**加粗**文本中的符号突出显示了重构的改变。一个层被表示为（# 输入通道，滤波器大小，# 输出通道）。

们认为基数维度控制更复杂转换的数量。我们通过实验证明，基数是一个重要维度，可以比宽度和深度维度更有效。

在本文中，我们考虑一种简单的设计转换函数的方法：所有的$\mathcal{T}_i$具有相同的拓扑结构。这扩展了VGG风格重复相同形状层的策略，有助于隔离几个因素并扩展到任意数量的转换。我们将每个$\mathcal{T}_i$设置为瓶颈形状的架构 [14]，如图 1（右侧）所示。在这种情况下，每个$\mathcal{T}_i$中的第一个1×1层产生低维嵌入。

在Eqn.(2)中聚合的转换充当剩余函数 [14]（图 1 右侧）：

$$\mathbf{y} = \mathbf{x} + \sum_{i=1}^{C} \mathcal{T}_i(\mathbf{x}), \tag{3}$$

where $\mathbf{y}$ is the output.

*与Inception-ResNet的关系。*一些张量操作显示出图1(右侧)中的模块（也显示在图3(a)中）等效于图3(b)。[3] 图3(b)与Inception-ResNet[37]模块相似，因为涉及到分支和在残差函数中进行连接。但与所有的Inception或Inception-ResNet模块不同，我们的模块在多个路径之间共享相同的拓扑结构。我们的模块需要最少额外的设计工作来设计每个路径。

与分组卷积的关系. 利用分组卷积的符号表示 [24]，上述模块变得更加简洁。[4] 这种改写在图 3(c)中有所体现。所有低维嵌入（第一个1×1层）可以被一个更宽的层替代（如，图 3(c)中的1×1, 128维）。分割是通过将输入通道分组的分组卷积层实现的。图 3(c)中的分组卷积层执行了32组卷积，其输入和输出通道均为4维。分组卷积层将它们拼接为层的输出。图 3(c)中的模块看起来像图 1(左)中的原始瓶颈残差模块，不同之处在于图 3(c)是更宽但稀疏连接的模块。

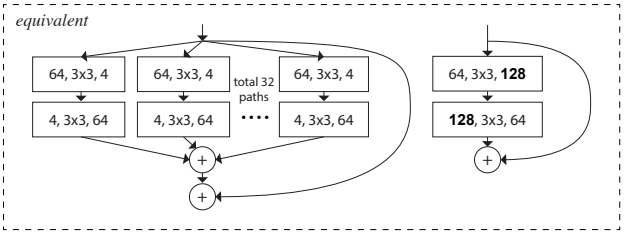

Figure 4.（**左**）：深度= 2的变换聚合。（**右**）：一个等效的块，显然更宽。

我们注意到，当块的深度 ≥ 3 时，重新表述才能产生非平凡的拓扑结构。如果块的深度 $= 2$（如 [14] 中的基础块），重新表述会使得一个宽泛的、密集的模块变得显然。请参见图 4 中的示例。

讨论

我们注意到，虽然我们提出的重构显示出串联（图3(b)）或分组卷积（图3(c)）的形式，但这些重构并不总是适用于方程（3）的一般形式，例如，如果变换$\mathcal{T}_i$采用任意形式且是异构的。我们选择在本文中使用同质形式，因为它们更简单且更易扩展。在这种简

[3]以下是一个非正式但描述性的证明。注意等式:$A_1B_1 + A_2B_2 = [A_1, A_2][B_1; B_2]$,其中[,]表示水平连接,[;]表示垂直连接。令A_i表示最后一层的权重,B_i表示块中倒数第二层的输出响应。对于$C = 2$的情况,在图3(a)中的逐元素相加为$A_1B_1 + A_2B_2$，在图3(b)中最后一层的权重为$[A_1, A_2]$，并且图3(b)中倒数第二层的输出连接为$[B_1; B_2]$。

[4]在分组卷积层 [24]，输入和输出通道被分成 C 组，每组内部单独执行卷积操作。

cardinality C	1	2	4	8	32
width of bottleneck d	64	40	24	14	4
width of group conv.	64	80	96	112	128

Table 2. 关于基数和宽度之间的关系（用于conv2模板），在残差块上大致保持复杂度。对于conv2模板，参数数量约为70k。FLOPs数量约为0.22十亿（conv2的参数数量×56×56）。

化情况下，图3(c)的分组卷积有助于简化实现。

3.4. Model Capacity

我们在接下来的部分实验中将展示，我们的模型在保持模型复杂度和参数数量不变的情况下提高了准确性。这不仅在实践中具有趣味性，更重要的是，复杂度和参数数量代表了模型的固有容量，因此通常被视为深度网络的基本属性[8]。

当我们评估不同的基数C时，同时保持复杂度不变，我们希望最小化对其他超参数的修改。我们选择调整瓶颈的宽度（例如，在图1（右侧）中为4维），因为它可以与块的输入和输出隔离开来。这种策略对其他超参数（块的深度或输入/输出宽度）不会产生更改，因此有助于我们专注于基数的影响。

在图1（左侧）中，原始的ResNet瓶颈块[14]具有约70k参数及与之成比例的FLOPs（在相同特征图大小上）。使用瓶颈宽度d，我们在图1（右侧）的模板中有：

$$C \cdot (256 \cdot d + 3 \cdot 3 \cdot d \cdot d + d \cdot 256) \tag{4}$$

当$C = 32$且$d = 4$时，方程(4)约为70k。表 2显示了基数C和瓶颈宽度d之间的关系。

由于我们在第 3.1节采用了两条规则，所以上述近似相等在ResNet瓶颈块与我们的ResNeXt在所有阶段上是成立的（除了特征图尺寸发生变化的子采样层）。表1比较了原始ResNet-50和我们类似容量的ResNeXt-50。[5] 我们注意到复杂性只能近似保持，但复杂性的差异很小，不会影响我们的结果。

4. Implementation details

我们的实现遵循[14]和`fb.resnet.torch`的公开代码[11]。在ImageNet数据集上，输入图像是通过[38]的比例和纵横比增强，由[11]实现的调整后从调整大小的图像中随机裁剪出的224×224像素图像。捷径连接是身份连接，除了那些增加维度的，它们是投影连接（[14]中的B类型）。 conv3、4和5的下采样是在每个阶段的第一个块的3×3层中通过步长为2的卷积完成的，如[11]所建议的。我们在8个GPU（每个GPU 32个）上使用小批量大小为256的SGD优化器。权重衰减率为0.0001，动量为0.9。我们从学习速率0.1开始，根据[11]中的计划将其分别除以10三次。我们采用[13]中的权重初始化。在所有割除比较中，我们评估从更短边为256的图像中取出的单个224×224中心裁剪的误差。

我们的模型由图 3(c)的形式实现。我们在图 3(c)中的卷积操作后立即执行批量归一化（BN）[17]。[6]在每个BN后立即进行ReLU操作，除了在块的输出处执行ReLU操作，紧随[14]上述步骤进行。

我们注意到图 3中的三种形式在适当处理BN和ReLU的情况下是严格等价的。我们训练了所有三种形式并获得了相同的结果。我们选择通过图3(c)实现，因为它比其他两种形式更简明和速度更快。

5. Experiments

5.1. Experiments on ImageNet-1K

我们在1000类 ImageNet 分类任务[33]上进行消融实验。我们遵循[14]构建了50层和101层的残差网络。我们简单地用我们的块替换了ResNet-50/101中的所有块。

符号说明。因为我们采用了第3.1节中的两条规则，我们可以仅通过模板来引用一个架构。例如，表 1展示了一个由基数 = 32 和瓶颈宽度 = 4d 的模板构建的ResNeXt-50（见图 3）。为简便起见，这个网络被表示为 ResNeXt-50 (**32**×**4d**)。我们注意到模板的输入/输出宽度固定为256-d（图 3），特征图被下采样时，宽度每次加倍（参见表 1）。

基数 *vs.*宽度。我们首先评估了在保持复杂度不变的情况下，基数 C 与瓶颈宽度之间的权衡，如表 2 中所列。表 3 显示了结果，图 5 显示了误差 *vs.* epoch 的曲线。与 ResNet-50（表 3 顶部和图 5 左侧）相比，32×4d

[5]参数数量稍微较小和FLOPs稍微较高主要是由于地图尺寸发生变化的块引起的。

[6]使用BN时,在图 3(a)的等效形式中，在汇总变换之后并添加到快捷连接之前执行BN。

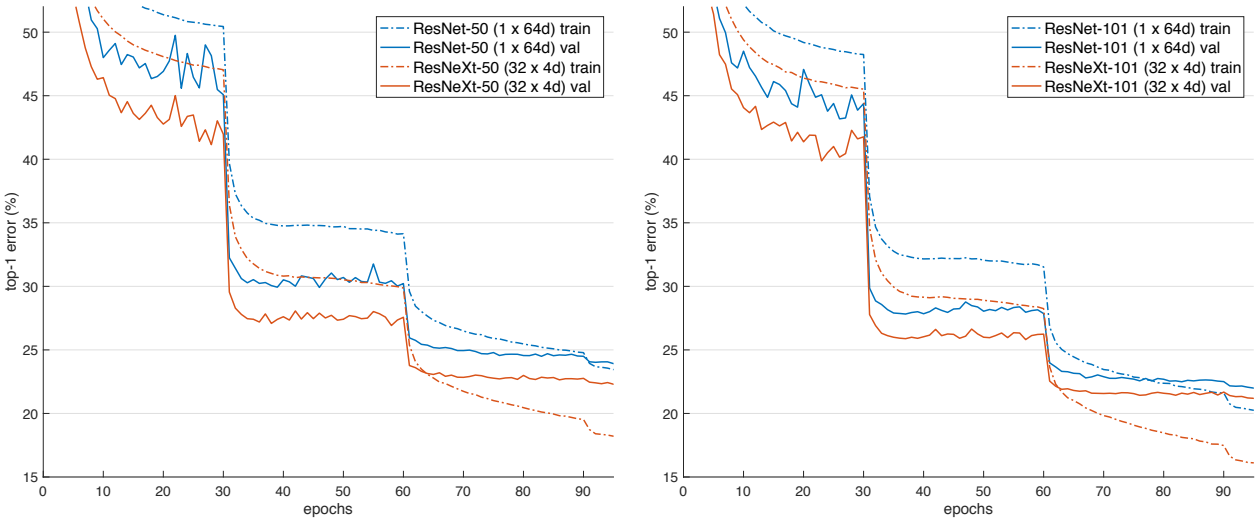

Figure 5. 在ImageNet-1K上的训练曲线。 (左): 保持复杂性的ResNet/ResNeXt-50（约4.1十亿次FLOPs，约2500万个参数）; (右): 保持复杂性的ResNet/ResNeXt-101（约7.8十亿次FLOPs，约4400万个参数）。

ResNeXt-50 的验证误差为 22.2%，比 ResNet 基准线的 23.9% 低 **1.7%**。当基数 C 从 1 增加到 32 但保持复杂度时，误差率持续降低。此外，32×4d ResNeXt 的训练误差也远远低于 ResNet 对应部分，表明收益来自更强大的表示，而不是来自正则化。

类似的趋势也在 ResNet-101 的情况中观察到（图 5 右侧，表 3 底部），32×4d ResNeXt-101 在验证误差上表现优于 ResNet-101 对应部分 0.8%。虽然验证误差的改进幅度比 50 层情况小，但训练误差的改进仍然很大（ResNet-101 20%，32×4d ResNeXt-101 16%，图 5 右侧）。事实上，更多的训练数据会扩大验证误差的差距，我们将在下一小节显示在 ImageNet-5K 数据集上这一点。

表 3 还建议，在保持复杂度的情况下，以降低宽度为代价来增加基数开始显示出饱和准确性当瓶颈宽度较小时。我们认为在这种权衡中继续降低宽度是不值得的。因此，我们在下文中采用不小于 4d 的瓶颈宽度。

在我们的实验中，我们使用了双腔光学谐振腔（OPO）来实现...

	setting	top-1 error (%)
ResNet-50	1 × 64d	23.9
ResNeXt-50	2 × 40d	23.0
ResNeXt-50	4 × 24d	22.6
ResNeXt-50	8 × 14d	22.3
ResNeXt-50	32 × 4d	**22.2**
ResNet-101	1 × 64d	22.0
ResNeXt-101	2 × 40d	21.7
ResNeXt-101	4 × 24d	21.4
ResNeXt-101	8 × 14d	21.3
ResNeXt-101	32 × 4d	**21.2**

Table 3. 在ImageNet-1K上的消融实验。 (上): 保持复杂度的ResNet-50（约4.1十亿次浮点运算）; (下): 保持复杂度的ResNet-101（约7.8十亿次浮点运算）。误差率是在单个裁剪的224×224像素上评估的。

6. 增加基数 vs 加深/加宽

接下来我们通过增加基数C或增加深度或宽度来研究增加复杂性。以下比较也可以参考ResNet-101基线的2倍FLOPs。我们比较以下变体，其FLOPs约为150亿。 (i) **增加深度**至200层。我们采用在[11]中实现的ResNet-200 [15]。(ii) **增加宽度**通过增加瓶颈宽度。(iii) **增加基数**通过将C加倍。

表 4 显示，通过2倍复杂性一致地降低了误差，相对于ResNet-101基线（22.0%）。但是当增加深度（ResNet-200，仅提高0.3%）或加宽度（更宽的ResNet-101，仅提高0.7%）时，改进很小。

相反，增加基数C显示出比增加深度或宽度更好的结果。2×64d ResNeXt-101（即，在1×64d ResNet-

	setting	top-1 err (%)	top-5 err (%)
1× complexity references:			
ResNet-101	1 × 64d	22.0	6.0
ResNeXt-101	32 × 4d	21.2	5.6
2× complexity models follow:			
ResNet-**200** [15]	1 × 64d	21.7	5.8
ResNet-101, wider	1 × **100**d	21.3	5.7
ResNeXt-101	**2** × 64d	20.7	5.5
ResNeXt-101	**64** × 4d	**20.4**	**5.3**

Table 4. 当 FLOPs 数增加到 ResNet-101 的 2 倍时，在 ImageNet-1K 上进行比较。误差率是基于 224×224 像素的单个裁剪进行评估。突出显示的因素是增加复杂性的因素。

101基线上将C加倍并保持宽度）将top-1错误率降低了1.3%，至20.7%。64×4d ResNeXt-101（即，在32×4d ResNeXt-101上将C加倍并保持宽度）将top-1错误率降低至20.4%。

我们还注意到，32×4d ResNet-101（21.2%）的性能优于更深的ResNet-200和更宽的ResNet-101，尽管其复杂性仅为其约50%。这再次表明，基数比深度和宽度更有效。

残差连接。 The following table shows the effects of the residual (shortcut) connections:

	setting	w/ residual	w/o residual
ResNet-50	1 × 64d	23.9	31.2
ResNeXt-50	32 × 4d	**22.2**	26.1

从 ResNeXt-50 中移除快捷连接会使误差增加 3.9 个百分点，达到 26.1%。从其 ResNet-50 对应模型中移除快捷连接则更糟糕（31.2%）。这些比较表明残差连接对于优化是有帮助的，而聚合变换则具有更强的表示能力，因为它们通常表现比具有或不具有残差连接的对应模型更好。

性能。 为简单起见，我们使用了Torch内置的分组卷积实现，没有进行特殊优化。我们注意到这种实现是蛮力的，不适合并行化。在8块NVIDIA M40 GPU上，训练32×4d的ResNeXt-101在表 3中每个小批次需要0.95秒，与具有相似FLOPs的ResNet-101基线的0.70秒相比。我们认为这是一个合理的开销。我们期望经过精心设计的底层实现（例如，在CUDA中）将减少这种开销。我们也预期在CPU上的推理时间会更少开销。对于复杂度模型2×（64×4d的ResNeXt-101）每个小批次需要1.7秒，总共在8块GPU上需时10天。

与最先进结果的比较。 表 5 展示了在ImageNet验证集上进行的单裁剪测试的更多结果。除了对 224×224 的裁剪进行测试外，我们还根据 [15] 评估了一种 320×320 的裁剪。我们的结果与 ResNet、Inception-v3/v4 以及 Inception-ResNet-v2 相比表现良好，实现了单裁剪的前5错误率为4.4%。此外，我们的架构设计比所有 Inception 模型都简单得多，需要手动设置的超参数也要少得多。

ResNeXt 是我们参加 ILSVRC 2016 分类任务的基础，在该任务中我们获得了第二名。我们注意到，许多模型（包括我们的模型）在使用多尺度和/或多裁剪测试后开始在这个数据集上达到饱和状态。我们通过在 [14] 中使用多尺度密集测试实现了单模型的17.7%/3.7%的前1/前5错误率，与采用多尺度、多裁剪测试的 Inception-ResNet-v2 的单模型结果17.8%/3.7%相当。我们在测试集上获得了3.03%的前5错误率的集成结果，与冠军的2.99%以及 Inception-v4/Inception-ResNet-v2 的3.08%相当 [37]。

	224×224		320×320 / 299×299	
	top-1 err	top-5 err	top-1 err	top-5 err
ResNet-101 [14]	22.0	6.0	-	-
ResNet-200 [15]	21.7	5.8	20.1	4.8
Inception-v3 [39]	-	-	21.2	5.6
Inception-v4 [37]	-	-	20.0	5.0
Inception-ResNet-v2 [37]	-	-	19.9	4.9
ResNeXt-101 (**64** × **4d**)	20.4	5.3	**19.1**	**4.4**

Table 5. 在ImageNet-1K验证集（单裁剪测试）上最先进的模型。ResNet/ResNeXt的测试尺寸为224×224和320×320，如[15]中所述，而Inception模型的尺寸为299×299。

重新定义1.11.1

6.1. Experiments on ImageNet-5K

在ImageNet-1K数据集上的表现似乎已经饱和。但我们认为这并不是因为模型的能力有限，而是因为数据集的复杂性。接下来，我们将在一个包含5000个类别的更大的ImageNet子集上评估我们的模型。

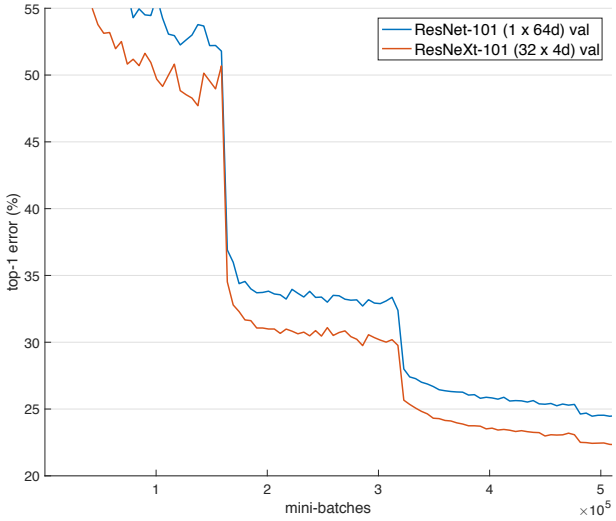

Figure 6. **ImageNet-5K** 实验。模型在 5K 数据集上进行训练，并在原始的 1K 验证集上进行评估，作为一个 1K 分类任务进行绘图。ResNeXt 和其 ResNet 对应模型具有类似的复杂度。

	setting	5K-way classification		1K-way classification	
		top-1	top-5	top-1	top-5
ResNet-50	1 × 64d	45.5	19.4	27.1	8.2
ResNeXt-50	32 × 4d	**42.3**	**16.8**	**24.4**	**6.6**
ResNet-101	1 × 64d	42.4	16.9	24.2	6.8
ResNeXt-101	32 × 4d	**40.1**	**15.1**	**22.2**	**5.7**

Table 6. 在**ImageNet-5K**上的错误率（%）。这些模型在ImageNet-5K上进行训练，并在*ImageNet-1K*验证集上进行测试，测试时被视为一个5K路分类任务或一个1K路分类任务。ResNeXt及其ResNet对应的模型复杂度相似。错误率是在单个224×224像素裁剪上进行评估的。

我们的5K数据集是全面的ImageNet-22K集合的一个子集[33]。这5000个类别包括原始的ImageNet-1K类别和全面的ImageNet集合中图片数量最多的额外4000个类别。5K集合包含680万张图片，大约是1K集合的5倍。由于没有官方的训练/验证集分割可用，因此我们选择在原始的ImageNet-1K验证集上进行评估。在这个1K类别的验证集上，模型可以被评估为一个5K路分类任务（所有被预测为其他4K类别的标签都是错误的）或者作为一个1K路分类任务（仅在1K类别上应用softmax函数）。

实现细节与第 4节中相同。5K训练模型都是从头开始训练的，并且训练的迷你批次数量与1K训练模型相同（因此1/5倍的epochs）。表 6和图 6显示在保持复杂度的情况下的比较结果。ResNeXt-50将5K路的top-1错误率与ResNet-50相比降低了**3.2%**，而ResNeXt-101将5K路的top-1错误率与ResNet-101相比降低了**2.3%**。在1K路错误率上也观察到相似的差距。这证明了ResNeXt更强大的表征能力。

此外，我们发现在5K集上训练的模型（表6中1K路错误率为22.2%/5.7%）在相同的1K路分类任务的验证集上，与在1K集上训练的模型（在表 3中为21.2%/5.6%）性能相当。这一结果是在不增加训练时间（因为迷你批次数量相同）和不进行微调的情况下实现的。我们认为这是一个令人鼓舞的结果，考虑到将5K个类别进行分类的训练任务更具挑战性。

6.2. Experiments on CIFAR

我们对CIFAR-10和CIFAR-100数据集进行了更多实验[23]。我们采用了[14]中的架构，并将基本残差块替换为$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}$我们的网络以一个单独的3×3卷积层开始，接着是3个阶段，每个阶段包含3个残差块，并以平均池化和全连接分类器结束（总共29层深），遵循[14]。我们采用与[14]相同的翻译和翻转数据增强方法。实现细节在附录中。

我们基于以上基准比较了两种增加复杂性的情况：（i）*增加基数* 并固定所有宽度，或（ii）*增加瓶颈宽度* 并固定基数= 1。我们在这些变化下训练和评估一系列网络。图 7 显示了测试错误率对模型尺寸的比较。我们发现增加基数比增加宽度更有效，与我们在ImageNet-1K上观察到的结果一致。表 7 显示了结果和模型尺寸，与Wide ResNet [43]进行了比较，后者是已知的最佳发布记录。我们的模型在相似的模型尺寸（34.4M）下显示出比Wide ResNet更好的结果。我们更大的方法在CIFAR-10上实现了3.58%的测试错误率（平均进行了10次运行），在CIFAR-100上为17.31%。据我们所知，这些是文献中最先进的结果（包括未发表的技术报告），采用了类似的数据增强。

重新定义命令arraystretch为1.05。设置表格列间距为4pt。

在本研究中，我们使用了定量问卷调查来收集数据。问卷包括了关于消费者购买行为和偏好的问题。我们采用了随机抽样的方法，以确保样本的代表性。调查结果显示，在价格和品质方面，消费者更注重品

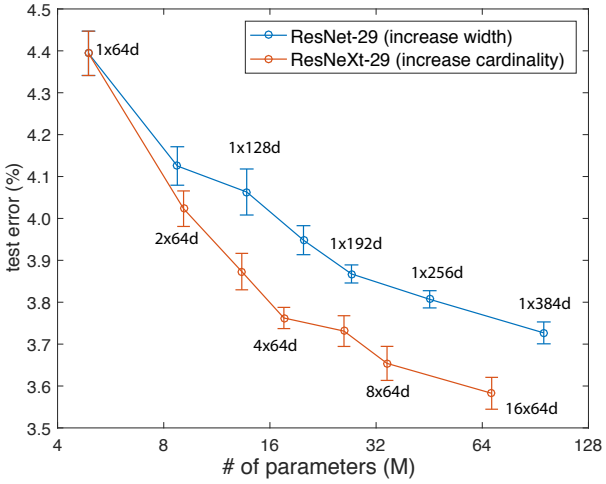

Figure 7. 在 CIFAR-10 数据集上测试误差与模型大小的比较。结果是通过 10 次运行计算得出的，用标准误差条显示。标签显示了模板的设置。

	# params	CIFAR-10	CIFAR-100
Wide ResNet [43]	36.5M	4.17	20.50
ResNeXt-29, 8×64d	34.4M	3.65	17.77
ResNeXt-29, 16×64d	68.1M	**3.58**	**17.31**

Table 7. 在CIFAR数据集上的测试误差（%）和模型大小。我们的结果是10次运行的平均值。

	setting	AP@0.5	AP
ResNet-50	1 × 64d	47.6	26.5
ResNeXt-50	32 × 4d	**49.7**	**27.5**
ResNet-101	1 × 64d	51.1	29.8
ResNeXt-101	32 × 4d	**51.9**	**30.0**

Table 8. 在 COCO `minival` 数据集上的目标检测结果。ResNeXt 及其 ResNet 对应模型具有类似的复杂性。

质。此外，消费者对品牌声誉和社交媒体的影响力也产生了显著影响。

6.3. Experiments on COCO object detection

接下来，我们在COCO对象检测数据集[27]上评估泛化能力。我们在80k的训练集以及一个包含35k验证子集的数据集上训练模型，并在一个包含5k验证子集的数据集上进行评估（称为`minival`），按照[1]的方法。我们评估COCO风格的平均精度（AP）以及AP@IoU=0.5[27]。我们采用基本的Faster R-CNN[32]，并按照[14]的方法将ResNet/ResNeXt嵌入其中。模型在ImageNet-1K上进行了预训练，并在检测集上进行微调。具体的实现细节见附录。

表 8显示了比较结果。在50层基准模型上，ResNeXt将AP@0.5提高了2.1%，AP提高了1.0%，而复杂度没有增加。ResNeXt在101层基准模型上的改进较小。我们推测更多的训练数据将会导致更大的差距，就像在ImageNet-5K数据集上观察到的那样。

值得注意的是，最近ResNeXt已经被应用于Mask R-CNN[12]中，在COCO实例分割和对象检测任务上取得了最先进的结果。

致谢

S.X.和Z.T.的研究部分受到NSF IIS-1618477项目的支持。作者要感谢Tsung-Yi Lin和Priya Goyal进行有价值的讨论。

A. Implementation Details: CIFAR

我们在50k训练集上训练模型，并在10k测试集上进行评估。输入图像大小为从补零40×40图像中随机裁剪出的32×32，或者其翻转，按照[14]的方法。没有使用其他数据增强方法。第一层是具有64个滤波器的3×3卷积。共有3个阶段，每个阶段有3个残差块，各阶段的输出尺寸分别为32、16和8[14]。网络最后通过全局平均池化和全连接层结束。当阶段改变（下采样）时，宽度增加2×，如在第3.1节中。模型在8个GPU上训练，每个小批量大小为128，权重衰减为0.0005，动量为0.9。我们从学习率0.1开始，训练模型300个时代，将学习率在第150和第225个时代降低。其他实施细节与[11]中相同。

B. Implementation Details: Object Detection

我们采用了Faster R-CNN系统[32]。为了简化，我们不在RPN和Fast R-CNN之间共享特征。在RPN步骤中，我们使用8个GPU进行训练，每个GPU承载2个图像每批次和每个图像256个锚点。我们以学习速率0.02进行120k批次的RPN训练，然后以0.002进行60k批次的训练。在Fast R-CNN步骤中，我们使用8个GPU进行训练，每个GPU承载1个图像和每批次64个区域。我们以学习速率0.005进行120k批次的Fast R-CNN训练，然后以0.0005进行60k批次的训练。我们采用0.0001的权重衰减和0.9的动量。其他实现

细节可参见https://github.com/rbgirshick/py-faster-rcnn.

References

[1] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016.

[2] G. Cantor. Über unendliche, lineare punktmannichfaltigkeiten, arbeiten zur mengenlehre aus den jahren 1872-1884. 1884.

[3] R. Collobert, S. Bengio, and J. Mariéthoz. Torch: a modular machine learning software library. Technical report, Idiap, 2002.

[4] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun. Very deep convolutional networks for natural language processing. *arXiv:1606.01781*, 2016.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[6] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014.

[7] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

[8] D. Eigen, J. Rolfe, R. Fergus, and Y. LeCun. Understanding deep architectures using a recursive convolutional network. *arXiv:1312.1847*, 2013.

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

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

[11] S. Gross and M. Wilber. Training and investigating Residual Nets. https://github.com/facebook/fb.resnet.torch, 2016.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. *arXiv:1703.06870*, 2017.

[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] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[16] Y. Ioannou, D. Robertson, R. Cipolla, and A. Criminisi. Deep roots: Improving cnn efficiency with hierarchical filter groups. *arXiv:1605.06489*, 2016.

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

[18] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.

[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] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. v. d. Oord, A. Graves, and K. Kavukcuoglu. Neural machine translation in linear time. *arXiv:1610.10099*, 2016.

[21] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. In *ICLR*, 2016.

[22] P. Kontschieder, M. Fiterau, A. Criminisi, and S. R. Bulò. Deep convolutional neural decision forests. In *ICCV*, 2015.

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

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

[25] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backprop-

agation applied to handwritten zip code recognition. *Neural computation*, 1989.

[26] M. Lin, Q. Chen, and S. Yan. Network in network. In *ICLR*, 2014.

[27] 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.

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

[29] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[30] A. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

[31] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In *NIPS*, 2015.

[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] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[34] 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.

[35] L. Sifre and S. Mallat. Rigid-motion scattering for texture classification. *arXiv:1403.1687*, 2014.

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

[37] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. In *ICLR Workshop*, 2016.

[38] 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.

[39] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[40] A. Veit, M. Wilber, and S. Belongie. Residual networks behave like ensembles of relatively shallow network. In *NIPS*, 2016.

[41] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144*, 2016.

[42] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig. The Microsoft 2016 Conversational Speech Recognition System. *arXiv:1609.03528*, 2016.

[43] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016.

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