Squeeze-and-Excitation Networks

Jie Hu[0000−0002−5150−1003] Li Shen[0000−0002−2283−4976] Samuel Albanie[0000−0001−9736−5134]
Gang Sun[0000−0001−6913−6799] Enhua Wu[0000−0002−2174−1428]

摘要—卷积神经网络（CNNs）的中心构建模块是卷积算子，它使网络能够通过在每一层的局部感受野中融合空间和通道信息来构建信息丰富的特征。以往广泛的研究探讨了这种关系的空间组成部分，试图通过增强特征层次结构中空间编码的质量来强化CNN的表示能力。在这项工作中，我们转而关注通道关系，并提出了一种新颖的架构单元，我们称之为“挤压-激励”（SE）块，通过明确建模通道之间的相互依赖性来自适应地校准通道特征响应。我们展示了这些块可以堆叠在一起形成SENet架构，在不同数据集上实现极其有效的泛化。我们进一步证明，SE块在略微增加计算成本的情况下，为现有的最先进CNNs带来了显著的性能改进。挤压-激励网络构成了我们在ILSVRC 2017分类比赛中获得第一名并将前5错误率降低至2.251%的基础，相对改进率达到约∼25%，超过了2016年获胜作品。模型和代码可在https://github.com/hujie-frank/SENet上找到。

Index Terms—Squeeze-and-Excitation, Image representations, Attention, Convolutional Neural Networks.

1 Introduction

Convolutional神经网络（CNNs）已被证明是处理各种视觉任务的有用模型 [1], [2], [3], [4]。在网络的每个卷积层中，一组滤波器表示沿输入通道的邻域空间连接模式—在局部感受野内将空间和通道信息融合在一起。通过将一系列卷积层与非线性激活函数和下采样运算符交错使用，CNN能够生成捕获分层模式并获得全局理论感受野的图像表示。计算机视觉研究的一个核心主题是寻找能够捕获图像中对于特定任务最显著的属性的更强大表示方式，从而实现更好的性能。作为用于视觉任务的广泛使用的模型族群，新的神经网络架构设计的发展现在代表着这一探索的关键前沿。最近的研究表明，通过将学习机制集成到网络中来捕捉特征之间的空间相关性，可以加强CNN生成的表示。一种广为人知的方法是通过Inception系列架构 [5], [6]将多尺度过程纳入网络模块以实现改进的性能。进一步的工作力图更好地建模空间依赖性 [7], [8]并将空间关注引入网络结构中 [9]。

- *Jie Hu and Enhua Wu are with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, 100190, China.*
  They are also with the University of Chinese Academy of Sciences, Beijing, 100049, China.
  Jie Hu is also with Momenta and Enhua Wu is also with the Faculty of Science and Technology & AI Center at University of Macau.
  E-mail: hujie@ios.ac.cn ehwu@umac.mo
- *Gang Sun is with LIAMA-NLPR at the Institute of Automation, Chinese Academy of Sciences. He is also with Momenta.*
  E-mail: sungang@momenta.ai
- *Li Shen and Samuel Albanie are with the Visual Geometry Group at the University of Oxford.*
  E-mail: {lishen,albanie}@robots.ox.ac.uk

在这篇论文中，我们研究了网络设计的一个不同方面 - 信道之间的关系。我们引入了一个新的架构单元，称为*Squeeze-and-Excitation* (SE) 块，旨在通过明确建模卷积特征信道之间的相互依赖关系来提高网络产生的表示质量。为此，我们提出了一种机制，使网络能够执行特征重校准，通过这种机制，网络可以学习利用全局信息，有选择地强调信息丰富的特征并抑制不太有用的特征。

SE构建块的结构如图1所示。对于任何给定的将输入$\mathbf{X}$映射到特征图$\mathbf{U}$（其中$\mathbf{U} \in \mathbb{R}^{H \times W \times C}$）的变换$\mathbf{F}_{tr}$，例如卷积操作，我们可以构建一个相应的SE块来执行特征重校准。特征$\mathbf{U}$首先通过*squeeze*操作，该操作通过在空间维度（$H \times W$）上聚合特征图来生成信道描述符。这个描述符的作用是生成通道特征响应的全局分布的嵌入，使网络的全局感受野中的信息可以被所有层使用。聚合后是一个*excitation*操作，采用简单的自门控机制，以嵌入为输入并产生一组每个通道的调制权重。这些权重应用于特征图$\mathbf{U}$，生成SE块的输出，可以直接馈送到网络的后续层。

可以通过简单堆叠一系列SE块来构建SE网络（SENet）。此外，这些SE块还可以用作网络架构中各个深度的原始块的替代品（7.4节）。虽然构建块的模板是通用的，但它在不同深度的作用在整个网络中是不同的。在较早的层中，它以一种与类别无关的方式激发信息丰富的特征，加强共享的低级表示。在较后的层中，SE块变得越来越专业化，并以高度类别特定的方式响应不同的输入（8.2节）。因此，SE块执行的特征重校准的益处可以通过网络累积。

设计和开发新的CNN架构是一项困难的工程任务，通常需要选择许多新的超参数和层配置。相反，SE块的结构简单，

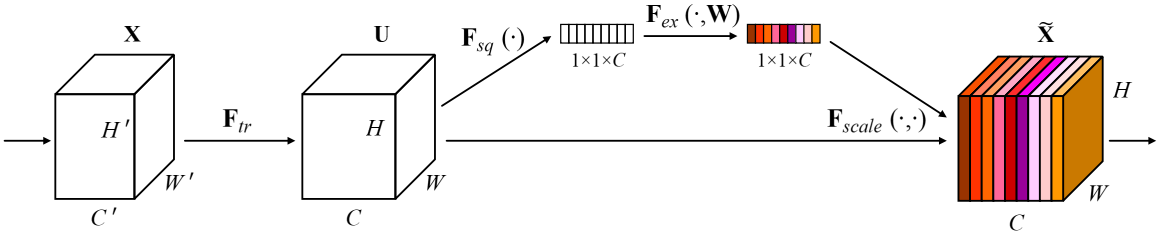


图 1. A Squeeze-and-Excitation block.

并且可以通过将组件替换为其SE对应组件，直接用于现有的最先进架构中，从而有效地增强性能。SE块还具有计算轻量化的特点，并且仅对模型复杂性和计算负担稍微增加。

为了验证这些论断，我们开发了几个SE网络，并在ImageNet数据集上进行了广泛评估。我们还呈现了超出ImageNet范围的结果，表明我们方法的优势并不局限于特定的数据集或任务。通过使用SE网络，我们在ILSVRC 2017分类竞赛中排名第一。我们最佳模型集在测试集上取得了2.251%的前5错误率[1]。与前一年冠军条目（前5错误率为2.991%）相比，这表示大约25%的相对改进。

2 Related Work

更深层次的架构。 VGGNets [11] 和 Inception 模型 [5] 表明增加网络深度可以显著提高其学习到的表示质量。通过调节输入到每一层的分布，Batch Normalization（BN） [6] 在深度网络中增加了学习过程的稳定性，并产生了更平滑的优化表面 [12]。在这些研究基础上，ResNets 表明通过使用基于恒等映射的跳跃连接可以学习到更深更强的网络 [13], [14]。Highway networks [15] 引入了门控机制来调节沿着捷径连接的信息流。继这些研究之后，网络层之间的连接方式有了进一步的改变 [16], [17]，这些改变展示了深度网络学习和表示特性的有益改进。

另外，但密切相关的研究线索集中在改进网络内部计算元素的函数形式上。分组卷积已被证明是提高学习变换的基数的一种流行方法 [18], [19]。更灵活的操作符组合可以通过多分支卷积操作实现 [5], [6], [20], [21]，这可以看作是分组操作符的自然扩展。在先前的工作中，跨通道相关性通常被映射为新的特征组合，要么独立于空间结构 [22], [23]，要么通过使用标准卷积滤波器与 1×1 卷积进行联合 [24]。大部分的研究集中于减小模型和计算复杂度的目标，反映了一个假设，即通道关系可以被构建为具有局部感受野的实例不可知函数的组合。相反，我们认为为该单元提供一个机制来明确地建模通道之间的动态、非线性依赖关系，并利用全局信息可以简化学习过程，并显著增强网络的表示能力。

算法架构搜索。 除了上述描述的工作之外，还有一系列旨在放弃手动架构设计，而是试图自动学习网络结构的研究历史悠久。这个领域的早期工作大多在神经进化社区中进行，该社区通过演化方法建立了在网络拓扑中进行搜索的方法 [25], [26]。尽管演化搜索通常计算密集，但已经取得了显著的成功，包括为序列模型找到良好的记忆单元 [27], [28]，以及为大规模图像分类学习复杂架构 [29], [30], [31]。为了减少这些方法的计算负担，基于拉马克继承 [32]和可微架构搜索 [33]等高效替代方案已被提出。

通过将架构搜索表述为超参数优化，随机搜索 [34]和其他更复杂的基于模型的优化技术 [35], [36]也可以用来解决问题。将拓扑选择视为可能设计的织物路径 [37]和直接架构预测 [38], [39]已被提出作为额外的可行架构搜索工具。通过强化学习技术取得了特别强大的结果 [40], [41], [42], [43], [44]。SE块可以作为这些搜索算法的基本构建模块，并且在同时进行的工作中被证明在这方面非常有效 [45]。

注意力与门控机制。

注意力可以解释为一种将可用计算资源偏向信号中最具信息性组件的分配的手段 [46], [47], [48], [49], [50], [51]。注意力机制已经证明在许多任务中具有实用性，包括序列学习 [52], [53]，图像中的定位和理解 [9], [54]，图像字幕生成 [55], [56] 和唇读 [57]。在这些应用中，注意力可以作为一个操作符，跟随着代表高层抽象的一个或多个层，用于在模态之间进行适应。一些工作提供了有趣的研究，结合了空间和通道注意力的使用 [58], [59]。王等 [58] 引入了一种基于沙漏模块的强大的主干和掩码注意力机制，该机制插入在深度残差网络的中间阶段。相比之下，我们提出的SE块包括一种轻量级的门控机制，其重点在于通过以一种计算效率高的方式建模通道间关系，从而增强网络的表征能力。

3 Squeeze-and-Excitation Blocks

Squeeze-and-Excitation 模块是一个计算单元，可以建立在将输入 $\mathbf{X} \in \mathbb{R}^{H' \times W' \times C'}$ 映射到特征图 $\mathbf{U} \in \mathbb{R}^{H \times W \times C}$ 的变换 $\mathbf{F}_{tr}$ 上。在接下来的符号表示中，我们将 $\mathbf{F}_{tr}$ 视为卷积算子，并使用 $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_C]$ 来表示学习到的滤波器核集，其中 $\mathbf{v}_c$ 是第 c 个滤波器的参数。我们可以将输出写为 $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_C]$，其中

1. http://image-net.org/challenges/LSVRC/2017/results

$$\mathbf{u}_c = \mathbf{v}_c * \mathbf{X} = \sum_{s=1}^{C'} \mathbf{v}_c^s * \mathbf{x}^s. \tag{1}$$

这里 $*$ 表示卷积，$\mathbf{v}_c = [\mathbf{v}_c^1, \mathbf{v}_c^2, \ldots, \mathbf{v}_c^{C'}]$，$\mathbf{X} = [\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{C'}]$ 并且 $\mathbf{u}_c \in \mathbb{R}^{H \times W}$。$\mathbf{v}_c^s$ 是一个二维空间核，表示$\mathbf{v}_c$的单个通道，其作用于$\mathbf{X}$的对应通道。为简化表示，偏置项被省略。由于输出是通过所有通道的求和产生的，则使得通道依赖性隐式地嵌入在$\mathbf{v}_c$中，但与滤波器捕获的局部空间相关性缠绕在一起。被卷积建模的通道关系是本质上隐式并且局部的（除了顶部层）。我们期望通过显式地建模通道间的相互依赖性来增强卷积特征的学习，从而使网络能够增加对信息特征的敏感性，这些特征可以被后续变换利用。因此，在输入下一个变换之前，我们希望通过两个步骤，即“压缩”和“激励”，为其提供全局信息并重新校准滤波器响应。图 1展示了SE块结构的示意图。

3.1 Squeeze: Global Information Embedding

为了解决利用信道依赖性的问题，我们首先考虑输出特征中每个通道的信号。每个学习的滤波器都使用局部感受野，因此转换输出 $\mathbf{U}$ 的每个单元无法利用该区域之外的上下文信息。

为了缓解这个问题，我们提出将全局空间信息“挤压”为通道描述符。这是通过使用全局平均池化来生成通道-wise统计信息实现的。形式上，通过在空间维度 $H \times W$ 上缩小 $\mathbf{U}$ 生成一个统计量 $\mathbf{z} \in \mathbb{R}^C$，使得 $\mathbf{z}$ 的第 c 个元素由以下方式计算：

$$z_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j). \tag{2}$$

讨论. 变换 $\mathbf{U}$ 的输出可以被解释为局部描述符的集合，这些描述符的统计特性对整个图像具有表现力。利用这样的信息在先前的特征工程工作中是普遍存在的 [60], [61], [62]。我们选择了最简单的聚合技术，即全局平均池化，尽管在这里也可以采用更复杂的策略。

3.2 Excitation: Adaptive Recalibration

为了利用聚合在*squeeze*操作中的信息，我们跟随着第二个旨在完全捕捉通道间依赖关系的操作。为了实现这一目标，该函数必须满足两个标准：首先，它必须灵活（特别是它必须能够学习通道之间的非线性交互作用），其次，它必须学习非互斥关系，因为我们希望确保允许强调多个通道（而不是强制一个独热激活）。为了满足这些条件，我们选择采用带有sigmoid激活的简单门控机制：

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z})), \tag{3}$$

其中 δ 指的是 ReLU [63] 函数，$\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ 和 $\mathbf{W}_2 \in \mathbb{R}^{C \times \frac{C}{r}}$。为了限制模型复杂性并促进泛化能力，我们通过在非线性函数周围形成具有两个全连接（FC）层的瓶颈来参数化门控机制，即具有降维比率 r 的降维层（这个参数选择在第 7.1 节中讨论），一个 ReLU，然后一个将维度增加至变换输出 $\mathbf{U}$ 的通道维度的层。块的最终输出是通过使用激活 $\mathbf{s}$ 对 $\mathbf{U}$ 进行重新缩放得到的：

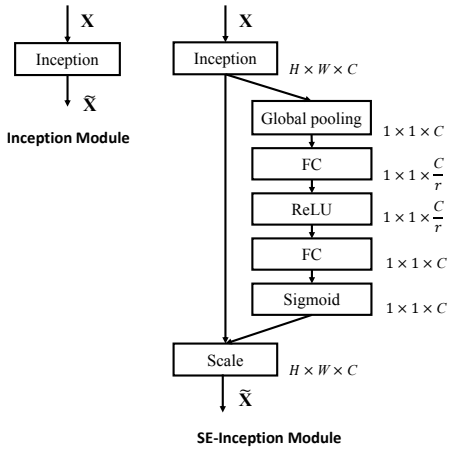


图 2. 原始Inception模块的模式（左）和SE-Inception模块（右）的模式。

$$\widetilde{\mathbf{x}}_c = \mathbf{F}_{scale}(\mathbf{u}_c, s_c) = s_c \, \mathbf{u}_c, \tag{4}$$

其中，$\widetilde{\mathbf{X}} = [\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2, \ldots, \widetilde{\mathbf{x}}_C]$，而$\mathbf{F}_{scale}(\mathbf{u}_c, s_c)$指的是标量$s_c$与特征图$\mathbf{u}_c \in \mathbb{R}^{H \times W}$之间的逐通道乘法。

讨论。激发算子将输入特定描述符 $\mathbf{z}$ 映射到一组通道权重。在这方面，SE块从本质上引入了取决于输入的动态性，可以被视为对通道进行自注意力函数操作，这与卷积滤波器响应的局部感受野无关。

3.3 Instantiations

SE模块可以集成到标准架构中，例如VGGNet [11]，可以在每个卷积之后的非线性层后插入。此外，SE模块的灵活性意味着它可以直接应用于超出标准卷积的转换。为了阐明这一点，我们通过将SE模块整合到更复杂架构的几个示例中，开发了SENets，具体描述如下。

首先考虑为Inception网络 [5] 构建SE模块。在这里，我们简单地将变换 $\mathbf{F}_{tr}$ 视为整个Inception模块（参见图2），通过对架构中的每个此类模块进行此更改，我们得到了一个*SE-Inception*网络。SE模块也可以直接与残差网络一起使用（图 3 展示了*SE-ResNet*模块的结构）。在这里，SE模块的变换 $\mathbf{F}_{tr}$ 被视为残差模块的非恒等分支。在与恒等分支求和之前，*Squeeze* 和 *Excitation* 都发挥作用。通过采用类似方案，还可以构建将SE模块与ResNeXt [19]、Inception-ResNet [21]、MobileNet [64]和ShuffleNet [65]集成的其他变体。对于SENet架构的具体示例，详细说明了SE-ResNet-50和SE-ResNeXt-50，具体内容见表 1。

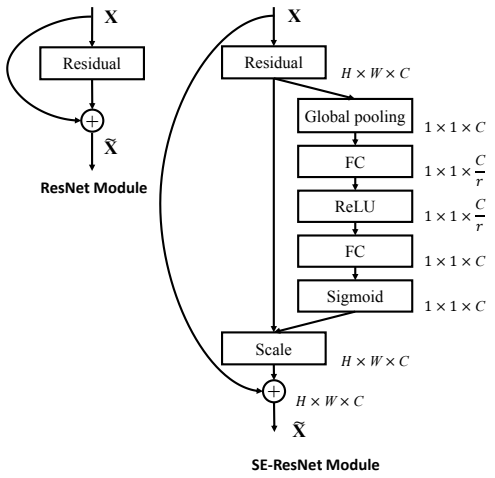


图 3. 原始残差模块的结构图（左）和SE-ResNet模块（右）。

由于SE模块的灵活性，它可以以几种可行的方式集成到这些架构中。因此，为了评估用于将SE模块整合到网络架构中所采用的集成策略对网络架构的敏感性，我们还提供了在第 7.5 节中探索不同设计的去除实验。

4 Model and Computational Complexity

对于所提出的SE块设计而言，它必须在提高性能和增加模型复杂度之间取得良好的平衡才能具有实际用途。为了说明与该模块相关的计算负担，我们以ResNet-50和SE-ResNet-50进行比较。ResNet-50需要在单次前向传递中为224 × 224像素输入图像执行∼3.86 GFLOPs。每个SE块在*挤压*阶段采用全局平均池化操作，在*激励*阶段使用两个小的全连接层，然后进行廉价的逐通道缩放操作。总体而言，当将缩减比率r（在第3.2节中介绍）设置为16时，SE-ResNet-50需要∼3.87 GFLOPs，相比原始的ResNet-50增加了0.26%。为了弥补这种轻微的额外计算负担，SE-ResNet-50的准确性超越了ResNet-50，并且接近需要∼7.58 GFLOPs的更深ResNet-101网络（表 **??**）。

在实际操作中，通过ResNet-50进行前向和后向传递需要190毫秒，而通过具有256张图片的训练小批量的SE-ResNet-50需要209毫秒（这两个时间是在具有8个NVIDIA Titan X GPU的服务器上执行）。我们认为这代表了一个合理的运行时开销，可以通过进一步优化流行的GPU库中的全局池化和小内积操作来进一步降低。由于对嵌入式设备应用的重要性，我们进一步对每个模型进行CPU推断时间基准测试：对于一个224 × 224像素输入图像，ResNet-50的时间是164毫秒，相比之下，SE-ResNet-50为167毫秒。我们认为SE块引起的轻微额外计算成本是合理的，因为它对模型性能有贡献。

接下来我们考虑所建议的SE块引入的附加参数。这些附加参数仅由门控机制的两个全连接层产生，因此构成了网络容量的一小部分。具体来说，这些FC层的权重参数引入的总数为：

$$\frac{2}{r}\sum_{s=1}^{S} N_s \cdot {C_s}^2, \tag{5}$$

其中r表示减少比率，S表示阶段数（一个阶段指的是在具有相同空间尺寸的特征图上操作的块的集合），C_s表示输出通道的维度，N_s表示阶段s中重复块的数量（当在FC层中使用偏置项时，引入的参数和计算成本通常可以忽略不计）。SE-ResNet-50引入了∼2.5 million额外参数，超出了ResNet-50需要的∼25 million参数，相当于∼10%的增加。在实践中，这些参数的绝大部分来自网络的最后阶段，在那里激励操作跨越了最多通道数。然而，我们发现，相对昂贵的SE块的最终阶段在性能上的表现成本很小（在ImageNet上的top-5误差小于<0.1%）去除后，相对参数增加降至∼4%，这在参数使用是关键考虑因素的情况下可能会有用（有关更多讨论，请参见第7.4节和第8.2节）。

5 Experiments

在本节中，我们进行实验来研究 SE blocks 在一系列任务、数据集和模型架构上的有效性。

5.1 Image Classification

为了评估SE块的影响，我们首先在ImageNet 2012数据集上进行实验 [10]，该数据集包括来自1000个不同类别的1.28百万训练图像和50K验证图像。我们在训练集上训练网络，并在验证集上报告top-1和top-5的错误率。

每个基线网络架构及其对应的SE对应物都采用相同的优化方案进行训练。我们遵循标准做法，并使用随机裁剪进行数据增强，使用尺度和宽高比 [5]裁剪到224x224像素（Inception-ResNet-v2为299x299像素和SE-Inception-ResNet-v2），并进行随机水平翻转。每个输入图像通过均值RGB通道减法进行归一化处理。所有模型都在我们设计用于高效并行训练大型网络的分布式学习系统ROCS上进行训练。优化使用同步SGD进行，动量为0.9，小批量大小为1024。初始学习率设置为0.6，并在每30个epochs降低一个数量级。模型从头开始训练100个epochs，使用 [66]中描述的权重初始化策略。减少比例r（在第 3.2节中）默认设置为16（除非另有说明）。

在评估模型时，我们应用中心裁剪，因此从每个图像中裁剪出224x224像素，此前将其较短的边调整为256个像素（Inception-ResNet-v2和SE-Inception-ResNet-v2中裁剪自每个图像，其较短的边调整为352个像素）。

网络深度。 我们首先将SE-ResNet与不同深度的ResNet架构进行比较，并在表 **??** 中报告结果。我们观察到，SE模块在不同深度上始终能提升性能，而计算复杂度仅略微增加。值得注意的是，SE-ResNet-50 实现了 6.62% 的单剪裁 top-5 验证错误率，超过 ResNet-50（7.48%）约 0.86%，仅用一半的总计算负担（3.87 GFLOPs vs. 7.58 GFLOPs）接近更深

表 1

(左) ResNet-50 [13]。(中) SE-ResNet-50。(右) 具有32×4d模板的SE-ResNeXt-50。在括号内列出了一个残差构建块的形状和特定参数设置的操作，并在括号外展示了每个阶段中堆叠块的数量。在*fc*之后的内括号指示了SE模块中两个全连接层的输出维度。

Output size	ResNet-50	SE-ResNet-50	SE-ResNeXt-50 (32 × 4d)
112 × 112	conv, 7 × 7, 64, stride 2		
56 × 56	max pool, 3 × 3, stride 2		
	$\begin{bmatrix}\text{conv}, 1\times1, 64\\ \text{conv}, 3\times3, 64\\ \text{conv}, 1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}\text{conv}, 1\times1, 64\\ \text{conv}, 3\times3, 64\\ \text{conv}, 1\times1, 256\\ fc, [16, 256]\end{bmatrix}\times3$	$\begin{bmatrix}\text{conv}, 1\times1, 128\\ \text{conv}, 3\times3, 128 \quad C=32\\ \text{conv}, 1\times1, 256\\ fc, [16, 256]\end{bmatrix}\times3$
28 × 28	$\begin{bmatrix}\text{conv}, 1\times1, 128\\ \text{conv}, 3\times3, 128\\ \text{conv}, 1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}\text{conv}, 1\times1, 128\\ \text{conv}, 3\times3, 128\\ \text{conv}, 1\times1, 512\\ fc, [32, 512]\end{bmatrix}\times4$	$\begin{bmatrix}\text{conv}, 1\times1, 256\\ \text{conv}, 3\times3, 256 \quad C=32\\ \text{conv}, 1\times1, 512\\ fc, [32, 512]\end{bmatrix}\times4$
14 × 14	$\begin{bmatrix}\text{conv}, 1\times1, 256\\ \text{conv}, 3\times3, 256\\ \text{conv}, 1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}\text{conv}, 1\times1, 256\\ \text{conv}, 3\times3, 256\\ \text{conv}, 1\times1, 1024\\ fc, [64, 1024]\end{bmatrix}\times6$	$\begin{bmatrix}\text{conv}, 1\times1, 512\\ \text{conv}, 3\times3, 512 \quad C=32\\ \text{conv}, 1\times1, 1024\\ fc, [64, 1024]\end{bmatrix}\times6$
7×7	$\begin{bmatrix}\text{conv}, 1\times1, 512\\ \text{conv}, 3\times3, 512\\ \text{conv}, 1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}\text{conv}, 1\times1, 512\\ \text{conv}, 3\times3, 512\\ \text{conv}, 1\times1, 2048\\ fc, [128, 2048]\end{bmatrix}\times3$	$\begin{bmatrix}\text{conv}, 1\times1, 1024\\ \text{conv}, 3\times3, 1024 \quad C=32\\ \text{conv}, 1\times1, 2048\\ fc, [128, 2048]\end{bmatrix}\times3$
1 × 1	global average pool, 1000-d fc, softmax		

表 2

在ImageNet验证集上的单裁剪错误率（%）和复杂性比较。 原始列指的是原始论文中报告的结果（ResNets的结果取自网站：https://github.com/Kaiminghe/deep-residual-networks）。为了进行公平比较，我们重新训练基线模型，并在重新实施列中报告分数。 *SENet*列指的是已添加SE块的相应架构。括号中的数字表示与重新实施基线的性能改进。 †表示该模型已在验证集的非黑名单子集上进行评估（这将在 [21]中更详细讨论），这可能略有改善结果。 VGG-16和SE-VGG-16均使用批量归一化训练。

	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [13]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [13]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [13]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50 [19]	22.2	-	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101 [19]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
VGG-16 [11]	-	-	27.02	8.81	15.47	$25.22_{(1.80)}$	$7.70_{(1.11)}$	15.48
BN-Inception [6]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2 [21]	$19.9^{\dagger}$	$4.9^{\dagger}$	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

的 ResNet-101 网络（6.52% top-5 错误率）的性能。这种模式在更深处得到重复，其中 SE-ResNet-101（6.07% top-5 错误率）不仅匹配，而且优于更深的 ResNet-152 网络（6.34% top-5 错误率）约 0.27%。虽然应该注意到，SE模块本身会增加深度，但它们以极其高效的方式增加深度并产生良好的回报，即使在基础架构的深度达到收益递减点时仍是如此。此外，我们看到这种收益在一系列不同网络深度上是一致的，这表明，SE模块带来的改进可能是对简单增加基础架构深度所获得改进的一种补充。

6 现代架构集成

接下来我们研究将SE块与两种最先进的架构Inception-ResNet-v2 [21]和ResNeXt（使用32 × 4d设置） [19]集成的效果，这两种架构在基础网络中引入了额外的计算构建块。我们构建了这些网络的SE块等效版本，SE-Inception-ResNet-v2和SE-ResNeXt（SE-ResNeXt-50的配置在表1中给出），并在表**??**中报告了结果。与先前的实验一样，我们观察到将SE块引入这两种架构中会显著提高性能。特别是，SE-ResNeXt-50的top-5误差为5.49%，优于其直接对应的ResNeXt-50（top-5误差为5.90%）以及更深的ResNeXt-101（top-5误差为5.57%），该模型的参数数量和计算负荷几乎是后者的两倍。我们注意到我们重新实现的Inception-

表 3
在ImageNet验证集上，单作物错误率（%）和复杂性比较。MobileNet 指的是 [64] 中的“1.0 MobileNet-224”，ShuffleNet 指的是 [65] 中的“ShuffleNet $1 \times (g = 3)$”。括号中的数字表示相对重新实现的性能提升。

	original		re-implementation				SENet			
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	MFLOPs	Params	top-1 err.	top-5 err.	MFLOPs	Params
MobileNet [64]	29.4	-	28.4	9.4	569	4.2M	$25.3_{(3.1)}$	$7.7_{(1.7)}$	572	4.7M
ShuffleNet [65]	32.6	-	32.6	12.5	140	1.8M	$31.0_{(1.6)}$	$11.1_{(1.4)}$	142	2.4M

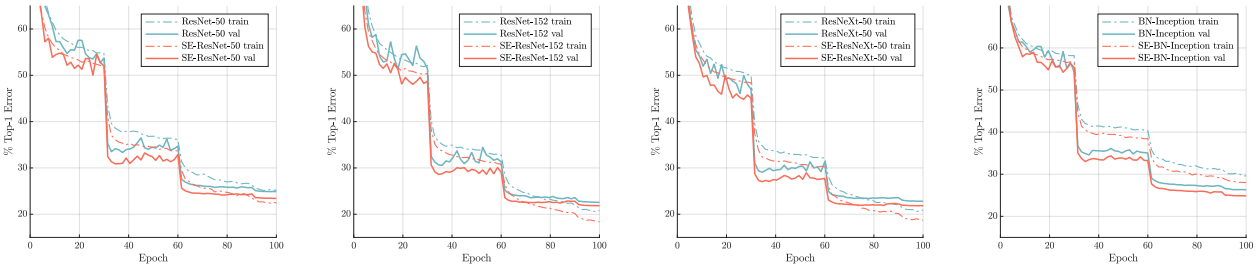


图 4. Training baseline architectures and their SENet counterparts on ImageNet. SENets exhibit improved optimisation characteristics and produce consistent gains in performance which are sustained throughout the training process.

ResNet-v2与 [21]中报告的结果略有不同。但是，就SE块的效果而言，我们观察到了类似的趋势，发现SE对应版本的top-5误差为4.79%，优于我们重新实现的Inception-ResNet-v2基线（top-5误差为5.21%）0.42%，同时也优于 [21]中报告的结果。

我们还评估了在操作非残差网络时SE块的效果，通过使用VGG-16 [11]和BN-Inception架构 [6]进行实验。为了从头开始训练VGG-16，我们在每个卷积层之后添加了批量归一化层。我们对VGG-16和SE-VGG-16使用相同的训练方案。比较结果显示在表**??**中。与残差基线架构的结果一样，我们观察到SE块在非残差设置下提高了性能。

为了提供有关SE块对这些模型优化的影响的一些见解，我们绘制了基线架构及其相应SE对应版本的示例训练曲线，如图4所示。我们观察到SE块在整个优化过程中持续改善。此外，这一趋势在考虑的一系列基线网络架构中基本保持一致。

移动端设置。 最后，我们考虑了移动优化网络类别中的两种代表性架构，MobileNet [64] 和 ShuffleNet [65]。在这些实验中，我们使用了一个大小为 256 的小批量，并采用了略少激进的数据增强和正则化，如 [65] 中所述。我们使用 SGD 优化算法在 8 个 GPU 上训练模型，动量设置为 0.9，初始学习率为 0.1，当验证损失达到平稳状态时，学习率会每次缩减 10 倍。总训练过程需要约 400 个 epochs（使我们能够复现 [65] 的基准性能）。表 3 中的结果表明，SE 块在最小化增加计算成本的情况下，始终能大幅提高准确性。

表 4
Classification error (%) on CIFAR-10.

	original	SENet
ResNet-110 [14]	6.37	5.21
ResNet-164 [14]	5.46	4.39
WRN-16-8 [67]	4.27	3.88
Shake-Shake 26 2x96d [68] + Cutout [69]	2.56	2.12

附加数据集。

接下来我们研究SE块的好处是否适用于超出ImageNet范围的数据集。我们在CIFAR-10和CIFAR-100数据集上对几种流行的基线架构和技术（ResNet-110 [14]，ResNet-164 [14]，WideResNet-16-8 [67]，Shake-Shake [68]和Cutout [69]）进行实验。这些数据集包含了一组包含50k训练和10k测试 32×32 像素的RGB图像，分别标记了10和100个类别。SE块集成到这些网络中的方法与第3.3节中描述的相同。对每种基线及其SENet对应项都采用了标准的数据增强策略 [24], [71] 进行训练。训练过程中，图像被随机水平翻转，并且在每一侧用四个像素进行零填充，然后随机取出一个 32×32 的裁剪。同时也进行均值和标准差归一化处理。训练超参数的设定（例如，小批量大小，初始学习率，权重衰减）与原始论文建议的相匹配。我们在Table 4中报告了每种基线及其SENet对应项在CIFAR-10上的性能，并在Table 5中报告了在CIFAR-100上的性能。我们观察到，在每种比较中，SENet都优于基线架构，表明SE块的好处不仅限于ImageNet数据集。

6.1 Scene Classification

我们还在Places365-Challenge数据集 [73]上进行实验以进行场景分类。该数据集包含365个类别中的800万张训练图像

表 5
Classification error (%) on CIFAR-100.

	original	SENet
ResNet-110 [14]	26.88	23.85
ResNet-164 [14]	24.33	21.31
WRN-16-8 [67]	20.43	19.14
Shake-Even 29 2x4x64d [68] + Cutout [69]	15.85	15.41

表 6
在Places365验证集上的单作物错误率（%）。

	top-1 err.	top-5 err.
Places-365-CNN [72]	41.07	11.48
ResNet-152 (ours)	41.15	11.61
SE-ResNet-152	**40.37**	**11.01**

和36500张验证图像。相对于分类，场景理解的任务提供了对模型泛化能力和处理抽象能力的替代评估。这是因为它通常需要模型处理更复杂的数据关联并且对外观变化的水平更具鲁棒性。

我们选择使用ResNet-152作为一种强大的基线来评估SE块的有效性，并遵循 [72], [74]中描述的训练和评估协议。在这些实验中，模型是从头开始训练的。我们在表 6中报告结果，并与先前的工作进行比较。我们观察到SE-ResNet-152（11.01%的top-5错误率）比ResNet-152（11.61%的top-5错误率）具有更低的验证错误率，并且证明了SE块也可以提高场景分类的性能。这个SENet超越了先前的现有模型Places-365-CNN [72]，在此任务上的top-5错误率为11.48%。

6.2 Object Detection on COCO

我们进一步评估SE块在使用COCO数据集进行目标检测任务的泛化 [75]。与之前的工作 [19] 一样，我们使用*minival*协议，即在80k个训练集和35k个验证子集的并集上训练模型，并在剩余的5k个验证子集上进行评估。权重是通过在ImageNet数据集上训练的模型的参数进行初始化的。我们使用Faster R-CNN [4] 检测框架作为评估我们模型的基础，并遵循 [76] 中描述的超参数设置（即使用'2x'学习计划进行端到端训练）。我们的目标是评估将物体检测器中的干线架构（ResNet）替换为SE-ResNet的效果，以便任何性能变化都可以归因于更好的表示。表 7 报告了使用ResNet-50，ResNet-101及其SE对应架构作为干架构的物体检测器在验证集上的性能。SE-ResNet-50在COCO标准AP指标上比ResNet-50提高了2.4%（相对提升6.3%），在AP@IoU=0.5上提高了3.1%。SE块也使深层ResNet-101架构受益，对AP指标实现了2.0%的提升（相对提升5.0%）。总之，这组实验表明了SE块的泛化能力。这种改进可以在广泛的架构、任务和数据集中实现。

表 7
在 COCO 的 minival 集上，Faster R-CNN 目标检测结果（%）。

	AP@IoU=0.5	AP
ResNet-50	57.9	38.0
SE-ResNet-50	61.0	40.4
ResNet-101	60.1	39.9
SE-ResNet-101	62.7	41.9

表 8
在ImageNet验证集上，现有最先进的CNNs使用尺寸为224×224和320×320 / 299×299的裁剪，的单作物错误率（%）。

	224×224		320×320 / 299×299	
	top-1 err.	top-5 err.	top-1 err.	top-5 err.
ResNet-152 [13]	23.0	6.7	21.3	5.5
ResNet-200 [14]	21.7	5.8	20.1	4.8
Inception-v3 [20]	-	-	21.2	5.6
Inception-v4 [21]	-	-	20.0	5.0
Inception-ResNet-v2 [21]	-	-	19.9	4.9
ResNeXt-101 (64×4d) [19]	20.4	5.3	19.1	4.4
DenseNet-264 [17]	22.15	6.12	-	-
Attention-92 [58]	-	-	19.5	4.8
PyramidNet-200 [77]	20.1	5.4	19.2	4.7
DPN-131 [16]	19.93	5.12	18.55	4.16
SENet-154	**18.68**	**4.47**	**17.28**	**3.79**

表 9
使用更大的裁剪尺寸/额外训练数据在ImageNet验证集上与最先进的CNNs进行比较（%）。†该模型是使用320×320的裁剪尺寸进行训练的。

	extra data	crop size	top-1 err.	top-5 err.
Very Deep PolyNet [78]	-	331	18.71	4.25
NASNet-A (6 @ 4032) [42]	-	331	17.3	3.8
PNASNet-5 (N=4,F=216) [35]	-	331	17.1	3.8
SENet-154†	-	320	16.88	3.58
AmoebaNet-C [79]	-	331	16.5	3.5
ResNeXt-101 32×48d [80]	✓	224	14.6	2.4

6.3 ILSVRC 2017 Classification Competition

SENets构建了我们参加ILSVRC比赛的基础，我们在比赛中获得了第一名。我们获奖的参赛作品由一组小型SENets组成，采用了标准的多尺度和多裁剪融合策略，以在测试集上获得2.251%的前5错误率。作为这次提交的一部分，我们构建了一个额外的模型，*SENet-154*，通过将SE块与修改后的ResNeXt [19] 结合（架构的细节在附录中提供）。我们使用标准裁剪尺寸（224×224 和 320×320）在ImageNet验证集上比较这个模型与以前的工作。我们观察到，SENet-154使用224×224 中央裁剪评估得到了18.68%的前1错误率和4.47%的

表 10
在不同的降维比率下，SE-ResNet-50在ImageNet上的单作物错误率（%）和参数大小。这里，*original*指的是ResNet-50。

Ratio r	top-1 err.	top-5 err.	Params
2	22.29	6.00	45.7M
4	22.25	6.09	35.7M
8	22.26	5.99	30.7M
16	22.28	6.03	28.1M
32	22.72	6.20	26.9M
original	23.30	6.55	25.6M

前5错误率，这是迄今为止报告的最佳结果。

在挑战之后，ImageNet基准测试取得了很大的进展。为了进行比较，我们在表 9中包括了目前我们所知道的最强结果。最近由 [79]报告了仅使用ImageNet数据的最佳性能。该方法使用强化学习来在训练期间开发新的数据增强策略，以改善 [31]搜索到的架构的性能。最佳整体表现由 [80]报告，使用了一个ResNeXt-101 $32 \times 48d$架构。他们在大约10亿个弱标记图像上预训练他们的模型，并在ImageNet上进行微调。更复杂的数据增强 [79]和广泛的预训练 [80]带来的改进可能与我们提出的对网络架构的变化相辅相成。

7 Ablation Study

在本节中，我们进行消融实验，以更好地理解在SE块的组件上使用不同配置的影响。所有消融实验都在单台机器（带有8个GPU）上在ImageNet数据集上执行。ResNet-50被用作骨干架构。我们在经验上发现，在ResNet架构中，删除激励操作中FC层的偏置有助于模拟通道依赖性，并在下面的实验中使用此配置。数据增强策略遵循第5.1节中描述的方法。为了使我们能够研究每个变体的性能上限，学习率初始化为0.1，并持续训练直到验证损失稳定（参考：使用270个epoch的固定计划进行训练（在第125、200和250个epoch减小学习率），ResNet-50和SE-ResNet-50的top-1和top-5错误率分别为（23.21%，6.53%）和（22.20%，6.00%）。总共训练约300个epoch）。然后学习率减小10倍，然后重复此过程（总共三次）。训练期间使用标签平滑正则化 [20]。

7.1 Reduction ratio

引入在公式5中的减少比r是一个超参数，它使我们能够改变网络中SE块的容量和计算成本。为了研究通过这个超参数调节性能和计算成本之间的权衡，我们对一系列不同r值的SE-ResNet-50进行了实验。表10中的比较显示，性能对一系列减少比值都表现出了鲁棒性。增加复杂度并不会单调地提高性能，而较小的比值会大幅增加模型的参数大小。设置$r = 16$实现了准确性和复杂性之间的良好平衡。在实践中，一直使用相同的比值可能不是最佳选择（由于不同层执行的独特作用），因此通过调整比值以满足给定基础架构的需求可能可以进一步改进。

表 11
在 ImageNet 数据集上使用不同的 squeeze 操作符对 SE-ResNet-50 的影响 (错误率 %)。

Squeeze	top-1 err.	top-5 err.
Max	22.57	6.09
Avg	**22.28**	**6.03**

表 12
在ImageNet数据集上采用不同非线性激励操作符对SE-ResNet-50模型的影响（错误率%）。

Excitation	top-1 err.	top-5 err.
ReLU	23.47	6.98
Tanh	23.00	6.38
Sigmoid	**22.28**	**6.03**

7.2 Squeeze Operator

我们探讨了选择全局平均池化作为我们挤压算子的选择的重要性（因为这个方法效果很好，我们没有考虑更复杂的替代方法）。结果报告在表 11 中。虽然最大池化和平均池化都有效，但平均池化实现了稍微更好的性能，从而证实了其作为挤压操作基础的选择。然而，我们注意到 SE 模块的性能对于特定聚合算子的选择是相当稳定的。

7.3 Excitation Operator

我们接下来评估激励机制的非线性选择。我们考虑另外两种选项：ReLU和tanh，并尝试用这些替代非线性替换sigmoid。结果见表 12。我们发现将sigmoid替换为tanh会略微降低性能，而使用ReLU的效果显著更差，实际上导致*SE-ResNet-50*的性能低于*ResNet-50*基准。这表明对于SE块要发挥有效作用，重点是仔细构建激励操作符。

7.4 Different stages

我们通过在不同阶段将SE blocks整合到ResNet-50中，逐个阶段探索SE blocks的影响。具体来说，我们将SE blocks添加

表 13
将SE块与ResNet-50在不同阶段集成对ImageNet的影响（错误率%）。

Stage	top-1 err.	top-5 err.	GFLOPs	Params
ResNet-50	23.30	6.55	3.86	25.6M
SE_Stage_2	23.03	6.48	3.86	25.6M
SE_Stage_3	23.04	6.32	3.86	25.7M
SE_Stage_4	22.68	6.22	3.86	26.4M
SE_All	22.28	6.03	3.87	28.1M

表 14
在ImageNet数据集上，结合ResNet-50的不同SE块集成策略对错误率的影响（%）。

Design	top-1 err.	top-5 err.
SE	22.28	6.03
SE-PRE	22.23	6.00
SE-POST	22.78	6.35
SE-Identity	22.20	6.15

表 15
在ResNet-50中，在每个残差分支的3x3卷积层中集成SE块的效果对ImageNet的影响（错误率%）。

Design	top-1 err.	top-5 err.	GFLOPs	Params
SE	22.28	6.03	3.87	28.1M
SE_3×3	22.48	6.02	3.86	25.8M

到中间阶段：stage_2、stage_3和stage_4，并在表 13中报告结果。我们发现，将SE blocks引入架构的每个阶段都能带来性能提升。此外，不同阶段的SE blocks带来的增益是互补的，它们可以有效地结合以进一步提升网络性能。

7.5 Integration strategy

最后，我们进行了切除研究，以评估将SE块集成到现有架构中时位置的影响。除了提出的SE设计外，我们考虑了三种变体：（1）SE-PRE块，其中SE块位于残差单元之前；（2）SE-POST块，其中SE单元移动到与恒等分支求和之后（ReLU之后）；（3）SE-Identity块，其中SE单元与残差单元并行放置在恒等连接上。这些变体在图 5中进行了说明，并且每个变体的性能报告在表 14中。我们观察到SE-PRE、SE-Identity和提出的SE块各自表现相似良好，而使用SE-POST块导致性能下降。这个实验表明，SE单元产生的性能改进对它们的位置相当稳健，只要它们在分支汇聚之前应用。

在上述实验中，每个SE块都放置在残差单元的结构之外。我们还构建了一个设计变体，将SE块移动到残差单元内部，在3×3卷积层之后直接放置。由于3×3卷积层拥有较少的通道，相应SE块引入的参数数量也减少了。表 15中的比较表明，SE_3×3变体在与标准SE块相比参数更少的情况下实现了可比较的分类准确度。虽然超出了本研究的范围，但我们预计通过为特定架构定制SE块使用，可以实现进一步的效率提升。

8 Role of SE blocks

尽管已经证明所提出的SE块能够改善多种视觉任务中的网络性能，我们还想了解挤压操作的相对重要性，以及激励机制在实践中是如何运作的。对深度神经网络学习的表征进行严格的理论分析仍然具有挑战性，因此我们采取经验方法来考察SE块所起的作用，目标是至少获得对其实际功能的基本理解。

表 16
在ImageNet数据集上Squeeze运算符的影响（错误率%）。

	top-1 err.	top-5 err.	GFLOPs	Params
ResNet-50	23.30	6.55	3.86	25.6M
NoSqueeze	22.93	6.39	4.27	28.1M
SE	**22.28**	**6.03**	3.87	28.1M

8.1 Effect of Squeeze

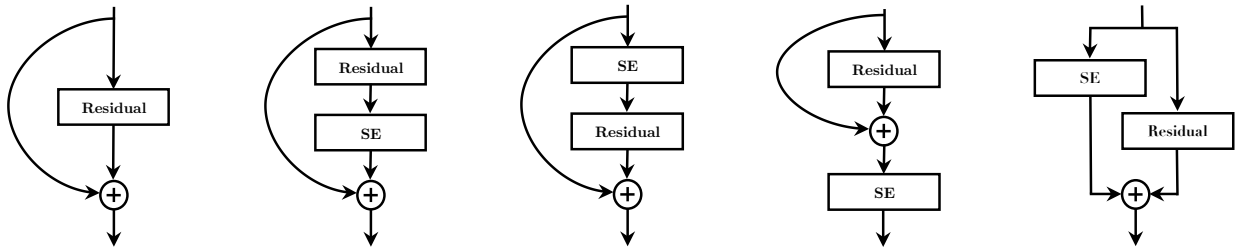
为了评估挤压操作产生的全局嵌入是否对性能起重要作用，我们对SE块的变体进行了实验，该变体增加了相同数量的参数，但不执行全局平均池化。具体来说，我们移除了池化操作，并将激励操作符中的两个全连接层替换为尺寸相同的1×1卷积，在激励操作符中保持空间维度与输入相同的输出，即*NoSqueeze*。与SE块相反，这些逐点卷积仅根据局部操作符的输出重映射通道。在实践中，深度网络的后续层通常具有（理论上的）全局感受野，但是在*NoSqueeze*变体中，全局嵌入不再直接在整个网络中访问。两个模型的准确性和计算复杂度与标准ResNet-50模型进行了比较，如表 16所示。我们注意到使用全局信息对模型性能有显著影响，突显了挤压操作的重要性。此外，与*NoSqueeze*设计相比，SE块可以以一种计算节俭的方式利用这些全局信息。

8.2 Role of Excitation

为了更清晰地了解SE块中激活算子的功能，本节我们研究了SE-ResNet-50模型中的示例激活，并在网络中不同深度的不同类别和不同输入图像之间的分布进行了考察。特别是，我们希望理解激活在不同类别图像和同一类别图像之间的变化。

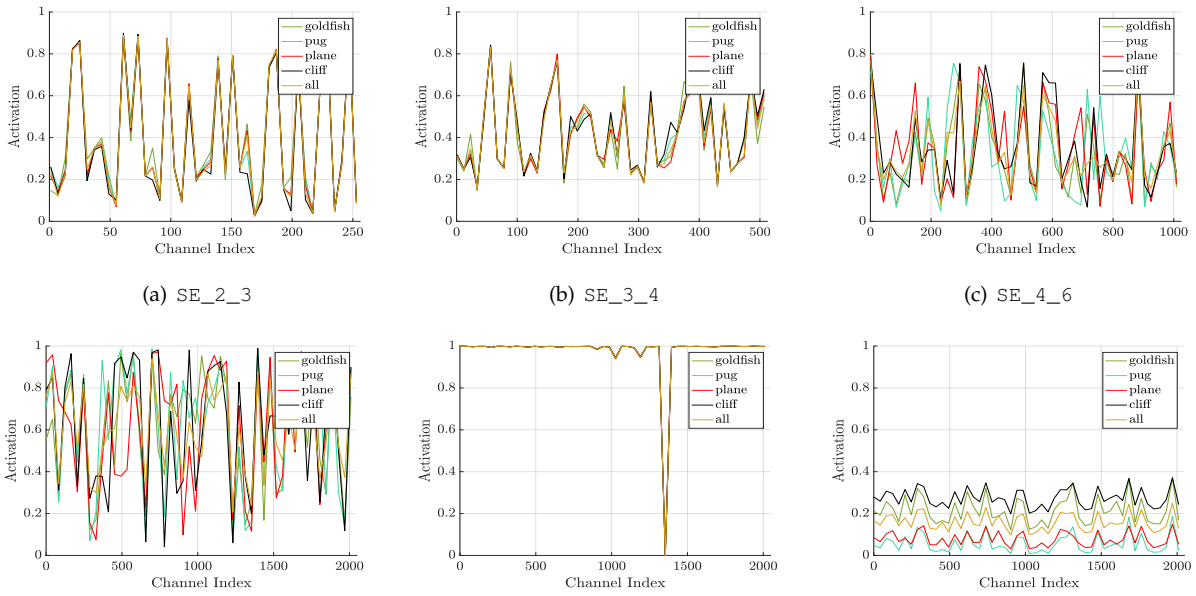
首先，我们考虑不同类别的激活分布。具体来说，我们从ImageNet数据集中选择了展示语义和外观多样性的四个类别，即*goldfish*（金鱼）、*pug*（巴哥犬）、*plane*（飞机）和*cliff*（悬崖）（这些类别的示例图像在附录中展示）。然后我们从验证集中为每个类别抽取50个样本，并计算每个阶段最后一个SE块（在下采样之前）中50个均匀抽样通道的平均激活，并在图 6中绘制它们的分布。作为参考，我们还绘制了所有1000个类别的平均激活分布。

关于激活操作的作用，我们得出以下三点观察结果。首先，在网络的较早层（例如SE_2_3），不同类别之间的分布非常相似。这表明特征通道的重要性可能在早期阶段被不同类别共享。第二个观察结果是，在更深的层次上，每个通道的值更加类别特定，因为不同类别展示出对特征的区分价值具有不同的偏好，例如SE_4_6和SE_5_1。这些观察结果与之前研究的发现一致 [81], [82]，即较早层次的特征通常更加通用（例如在分类任务上是与类别无关的），而较深层次的特征表现出更高的特异性 [83]。



(a) Residual block (b) Standard SE block (c) SE-PRE block (d) SE-POST block (e) SE-Identity block

图 5. 在消融研究中探索的SE块集成设计。



(a) SE_2_3 (b) SE_3_4 (c) SE_4_6

(d) SE_5_1 (e) SE_5_2 (f) SE_5_3

图 6. 在ImageNet上的SE-ResNet-50中，由激励运算符在不同深度引发的激活。每组激活根据以下方案命名：SE_stageID_blockID。除了SE_5_2处的异常行为外，随着深度增加，激活变得越来越具有类别特异性。

接下来，在网络的最后阶段，我们观察到一种略有不同的现象。SE_5_2呈现出一种有趣的趋势，即大多数激活接近于一。当所有激活值都为一时，一个SE块就变成了单位算子。在网络末尾的SE_5_3（紧接着全局池化层和分类器），不同类别之间也出现了类似的模式，尽管存在着适度的尺度变化（可以通过分类器进行调整）。这表明SE_5_2和SE_5_3在为网络提供重新校准方面比之前的块不那么重要。这一发现与第4节中经验调查结果一致，该结果表明通过移除最后阶段的SE块可以显著减少额外的参数数量，而性能损失较小。

最后，我们展示了两个样本类别（*goldfish*和*plane*）内图像实例的激活均值和标准差，如图 7所示。我们观察到了与类间可视化一致的趋势，表明在网络的较后层次，单个类别的表示多样性很大，网络学习利用特征校准来改善其区分性能 [84]。总之，SE块产生特定于实例的响应，但却在架构的不同层次上支持模型日益类别特定的需求。

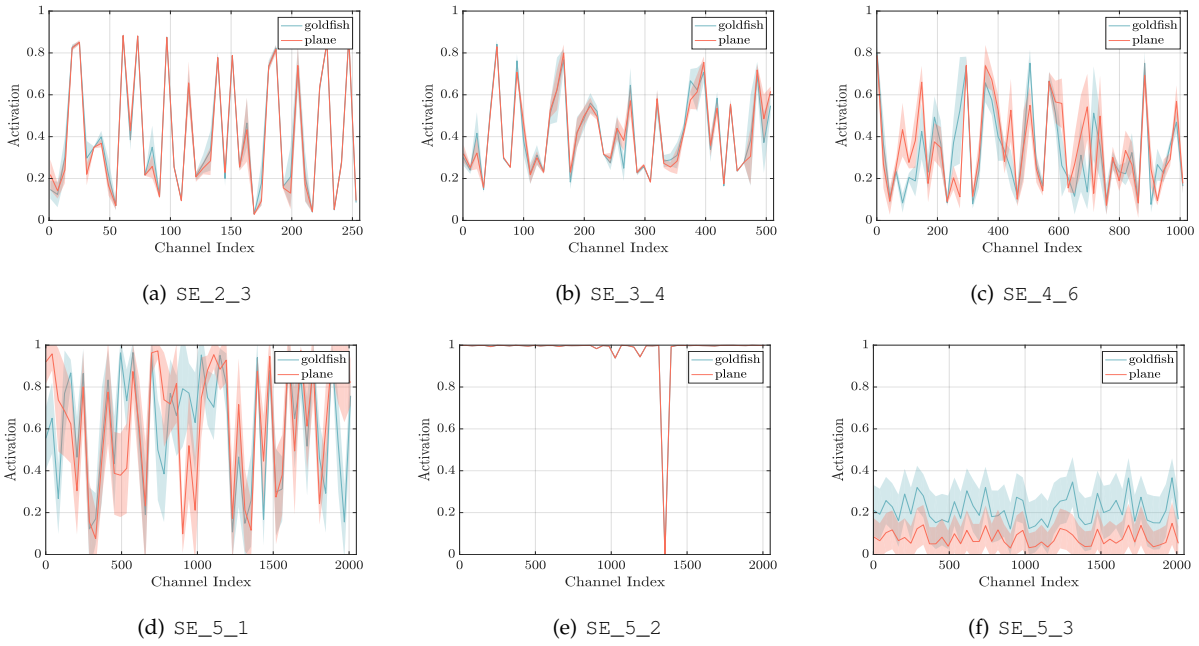
9 Conclusion

在本文中，我们提出了SE块，这是一种旨在通过使网络能够执行动态的逐通道特征校准来提高网络的表征能力的架构单元。广泛的实验表明了SENets的有效性，在多个数据集和任务上实现了最先进的性能。此外，SE块揭示了先前架构无法充分建模逐通道特征依赖性的一些问题。我们希望这种洞察力对于其他需要强有力的判别特征的任务可能会有所帮助。最后，SE块产生的特征重要性数值可能对其他任务如网络剪枝和模型压缩有用。

Acknowledgments

Acknowledgment

作者们特别感谢来自Momenta的李超和王光源在CIFAR数据集的训练系统优化和实验中的贡献。我们还要感谢Andrew Zisserman、Aravindh Mahendran和Andrea Vedaldi的许多有益讨论。本工作部分得到了NSFC资助（61632003、61620106003、61672502、61571439）、中国国



(a) SE_2_3 (b) SE_3_4 (c) SE_4_6

(d) SE_5_1 (e) SE_5_2 (f) SE_5_3

图 7. SE-ResNet-50中不同模块对来自ImageNet金鱼和飞机类别的图像样本所引起的*Excitation*激活。该模块被命名为"SE_stageID_blockID"。

家重点研发计划（2017YFB1002701）和澳门科技发展基金（068/2015/A2）的支持。Samuel Albanie得到了EPSRC AIMS CDT EP/L015897/1的支持。

(a) goldfish (b) pug (c) plane (d) cliff

图 8. 在第7.2节描述的实验中使用的ImageNet四个类别的示例图像。

APPENDIX: DETAILS OF SENET-154

SENet-154是通过将SE块合并到修改版的64×4d ResNeXt-152中构建的，该版本是通过采用ResNet-152的块堆叠策略来扩展原始的ResNeXt-101 [19]。除了使用SE块外，该模型在设计和训练方面的进一步差异如下： (a) 对于每个瓶颈建块，第一个1×1卷积通道的数量减半，以降低模型的计算成本，同时最小程度地降低性能。 (b) 第一个7×7卷积层被替换为三个连续的3×3卷积层。 (c) 用一个3×3步幅为2的卷积替换了1×1下采样投影与步幅为2的卷积，以保持信息。 (d) 在分类层之前插入了一个dropout层（dropout比率为0.2）来减少过拟合。 (e) 在训练过程中使用了标签平滑正则化（如[20]中介绍）。 (f) 所有BN层的参数在最后几个训练epoch被冻结，以确保训练和测试之间的一致性。 (g) 使用8台服务器（64个GPU）并行训练，以实现大批次大小（2048）。初始学习率设定为1.0。

参考文献

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Conference on Neural Information Processing Systems*, 2012.

[2] A. Toshev and C. Szegedy, "DeepPose: Human pose estimation via deep neural networks," in *CVPR*, 2014.

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

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Conference on Neural Information Processing Systems*, 2015.

[5] 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.

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

[7] 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.

[8] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*, 2016.

[9] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Conference on Neural Information Processing Systems*, 2015.

[10] 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," *International Journal of Computer Vision*, 2015.

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

[12] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization? (no, it is not about internal covariate shift)," in *Conference on Neural Information Processing Systems*, 2018.

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

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

[15] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Conference on Neural Information Processing Systems*, 2015.

[16] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Conference on Neural Information Processing Systems*, 2017.

[17] G. Huang, Z. Liu, K. Q. Weinberger, and L. Maaten, "Densely connected convolutional networks," in *CVPR*, 2017.

[18] Y. Ioannou, D. Robertson, R. Cipolla, and A. Criminisi, "Deep roots: Improving CNN efficiency with hierarchical filter groups," in *CVPR*, 2017.

[19] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *CVPR*, 2017.

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

[21] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI Conference on Artificial Intelligence*, 2016.

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

[23] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, 2017.

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

[25] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms." in *ICGA*, 1989.

[26] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, 2002.

[27] J. Bayer, D. Wierstra, J. Togelius, and J. Schmidhuber, "Evolving memory cell structures for sequence learning," in *ICANN*, 2009.

[28] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *ICML*, 2015.

[29] L. Xie and A. L. Yuille, "Genetic CNN," in *ICCV*, 2017.

[30] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *ICML*, 2017.

[31] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *arXiv preprint arXiv:1802.01548*, 2018.

[32] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," *arXiv preprint arXiv:1804.09081*, 2018.

[33] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[34] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *JMLR*, 2012.

[35] C. Liu, B. Zoph, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *ECCV*, 2018.

[36] R. Negrinho and G. Gordon, "Deeparchitect: Automatically designing and training deep architectures," *arXiv preprint arXiv:1704.08792*, 2017.

[37] S. Saxena and J. Verbeek, "Convolutional neural fabrics," in *Conference on Neural Information Processing Systems*, 2016.

[38] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "SMASH: one-shot model architecture search through hypernetworks," in *ICLR*, 2018.

[39] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," in *ICLR Workshop*, 2018.

[40] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *ICLR*, 2017.

[41] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *ICLR*, 2017.

[42] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *CVPR*, 2018.

[43] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *ICLR*, 2018.

[44] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *ICML*, 2018.

[45] M. Tan, B. Chen, R. Pang, V. Vasudevan, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," *arXiv preprint arXiv:1807.11626*, 2018.

[46] B. A. Olshausen, C. H. Anderson, and D. C. V. Essen, "A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information," *Journal of Neuroscience*, 1993.

[47] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.

[48] L. Itti and C. Koch, "Computational modelling of visual attention," *Nature reviews neuroscience*, 2001.

[49] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order boltzmann machine," in *Conference on Neural Information Processing Systems*, 2010.

[50] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *Conference on Neural Information Processing Systems*, 2014.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Conference on Neural Information Processing Systems*, 2017.

[52] T. Bluche, "Joint line segmentation and transcription for end-to-end handwritten paragraph recognition," in *Conference on Neural Information Processing Systems*, 2016.

[53] A. Miech, I. Laptev, and J. Sivic, "Learnable pooling with context gating for video classification," *arXiv:1706.06905*, 2017.

[54] C. Cao, X. Liu, Y. Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, W. Xu, D. Ramanan, and T. S. Huang, "Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks," in *ICCV*, 2015.

[55] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015.

[56] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T. Chua, "SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning," in *CVPR*, 2017.

[57] J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, "Lip reading sentences in the wild," in *CVPR*, 2017.

[58] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *CVPR*, 2017.

[59] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *ECCV*, 2018.

[60] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *CVPR*, 2009.

[61] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International Journal of Computer Vision*, 2013.

[62] L. Shen, G. Sun, Q. Huang, S. Wang, Z. Lin, and E. Wu, "Multi-level discriminative dictionary learning with application to large scale image classification," *IEEE TIP*, 2015.

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

[64] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017.

[65] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *CVPR*, 2018.

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

[67] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *BMVC*, 2016.

[68] X. Gastaldi, "Shake-shake regularization," *arXiv preprint arXiv:1705.07485*, 2017.

[69] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[70] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[71] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *ECCV*, 2016.

[72] L. Shen, Z. Lin, G. Sun, and J. Hu, "Places401 and places365 models," https://github.com/lishen-shirley/Places2-CNNs, 2016.

[73] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[74] L. Shen, Z. Lin, and Q. Huang, "Relay backpropagation for effective learning of deep convolutional neural networks," in *ECCV*, 2016.

[75] 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.

[76] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron," https://github.com/facebookresearch/detectron, 2018.

[77] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," in *CVPR*, 2017.

[78] X. Zhang, Z. Li, C. C. Loy, and D. Lin, "Polynet: A pursuit of structural diversity in very deep networks," in *CVPR*, 2017.

[79] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018.

[80] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, "Exploring the limits of weakly supervised pretraining," in *ECCV*, 2018.

[81] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *ICML*, 2009.

[82] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Conference on Neural Information Processing Systems*, 2014.

[83] A. S. Morcos, D. G. Barrett, N. C. Rabinowitz, and M. Botvinick, "On the importance of single directions for generalization," in *ICLR*, 2018.

[84] J. Hu, L. Shen, S. Albanie, G. Sun, and A. Vedaldi, "Gather-excite: Exploiting feature context in convolutional neural networks," in *Conference on Neural Information Processing Systems*, 2018.