挤压与激励网络

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：卷积算子是卷积神经网络的核心组成部分，它通过在每一层的局部感受性 CNN 区域内融合空间和通道信息，使网络能够构造信息特征。先前的大量研究已经探讨了这种关系的空间成分，试图通过提高整个特征层次的空间编码质量来增强 CNN 的表征能力。在这项工作中，我们将重点放在通道关系上，并提出了一种新的体系结构单元，我们称之为"挤压和激励"(SE)块，它通过显式地建模通道之间的相互依赖来自适应地重新校准通道级特征响应。我们证明了这些块可以是堆叠在一起形成 SENet 体系结构，可以在不同的数据集上非常有效地泛化。我们进一步证明了 SE 块为现有的最先进的 cnn 带来了显著的性能提高，但计算成本略有增加。挤压和激励网络构成了我们 ILSVRC 2017 分类提交的基础，该分类提交获得了第一名，并将前 5 个错误减少到 2.251%，比 2016 年的获奖条目相对提高了约 25%。模型和代码可在 https://github.com/hujie-frank/SENet 获得。

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

1 简介

事实证明，卷积神经网络（CNN）是处理各种视觉任务的有用模型[1], [2], [3], [4]。在网络中的每个卷积层，滤波器集合表达了沿输入通道的邻域空间连接模式--在局部感受野内将空间信息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降采样运算符交织在一起，CNN 能够生成图像表征，捕捉分层模式并实现全局理论感受野。计算机视觉研究的一个核心主题是寻找更强大的表征，只捕捉图像中对特定任务而言最突出的属性，从而提高性能。作为视觉任务中广泛使用的模型系列，开发新的神经网络架构设计是这一探索的关键前沿。最新研究表明，将学习机制整合到网络中，有助于捕捉特征之间的空间相关性，从而加强 CNN 生成的表征。Inception 系列架构[5]、[6]所采用的一种方法就是将多尺度过程整合到网络模块中，从而提高性能。进一步的研究试图更好地模拟空间依赖关系[7]、[8]，并将空间注意力纳入网络结构[9]。

在本文中，我们研究了网络设计的一个不同方面--通道之间的关系。我们引入了一个新的结构单元，并将其称为 "挤压与激发（SE）"块，目的是通过明确地模拟卷积特征通道之间的相互依存关系，提高网络生成的表征质量。为此，我们提出了一种允许网络执行特征重新校准的机制，通过这种机制，网络可以学会使用全局信息，有选择地强调信息量大的特征，抑制用处不大的特征。

SE 构件的结构如图 1 所示。对于任何将输入 X 映射到特征图 U（其中 $U \in R^{H \times W \times C}$ ）的给定变换 F_{tr}（例如卷积），我们都可以构建一个相应的 SE 模块来执行特征再校准。首先对特征 U 进行挤压运算，通过在空间维度（H×W）上聚合特征图来生成信道描述符。该描述符的功能是对通道式特征响应的全局分布进行嵌入，使网络的所有层都能使用来自全局感受野的信息。聚合之后是激励操作，其形式是一种简单的自门控机制，它将嵌入作为输入，并产生一系列每个通道的调制权重。这些权重应用于特征图 U，生成 SE 模块的输出，并直接输入网络的后续层。

通过简单地堆叠 SE 块集合，可以构建 SE 网络 (SENet)。此外，这些 SE 模块还可以在网络架构的不同深度上替代原始模块（第 6.4 节）。虽然构建块的模板是通用的，但它在整个网络的不同深度所发挥的作用却各不相同。在较早的层中，它以与类无关的方式激发信息特征，加强共享的低层表征。在后几层，SE 模块变得越来越专业化，并以高度特定的方式对不同的输入做出反应（第 7.2 节）。因此，由 SE 块执行的特征重新校准所带来的好处可以在整个网络中累积。

设计和开发新的 CNN 架构是一项艰巨的工程任务，通常需要选择许多新的超参数和层配置。相比之下，SE 块的结构简单，可以直接用于现有的最先进架构中，只需将组件替换为 SE 对应组件，即可有效提高性能。此外，SE 模块在计

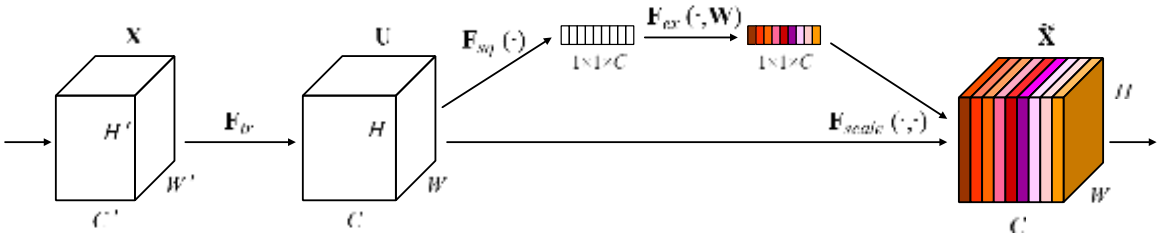


图 1: 挤压-激发模块。

算上也很轻便，只略微增加了模型复杂度和计算负担。

为了证明这些说法，我们开发了几个 SENets，并在 ImageNet 数据集 [10] 上进行了广泛的评估。我们还展示了 ImageNet 以外的结果，这些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 SENets，我们在 ILSVRC 2017 分类竞赛中排名第一。我们的最佳模型组合在测试集上取得了 2.251% 的前五名错误率 1。与上一年的冠军作品（前五名误差为 2.991%）相比，这意味着大约 25% 的相对改进。

2 相关工作

更深的结构. VGGNets 模型[11] 和 Inception 模型[5] 表明，增加网络的深度可以显著提高网络所能学习的表征质量。通过调节每层输入的分布，批量归一化（BN）[6] 增加了深度网络学习过程的稳定性，并产生了更平滑的优化表面[12]。在这些工作的基础上，ResNets 证明，通过使用基于身份的跳过连接，可以学习到更深、更强的网络[13], [14]。高速公路网络[15]引入了一种门控机制，以调节沿捷径连接的信息流。在这些工作之后，人们对网络层之间的连接进行了进一步的重构[16]、[17]，结果表明深度网络的学习和表征特性有了很好的改善。

另一个密切相关的研究方向是改进网络中计算元素功能形式的方法。分组卷积已被证明是增加学习到的变换基数的流行方法[18] [19]。多分支卷积可以实现更灵活的算子组合 [5]、[6]、[20]、[21]，这可以看作是分组算子的自然扩展。在之前的研究中，跨信道相关性通常被映射为新的特征组合，这些特征组合可以独立于空间结构[22]、[23]，也可以通过使用标准卷积滤波器[24]与 1 × 1 卷积联合实现。这些研究大多集中在降低模型和计算复杂度的目标上，反映了一种假设，即信道关系可以表述为具有局部感受野的实例无关函数的组合。相比之下，我们认为，为单元提供一种机制，利用全局信息对通道之间的动态非线性依赖关系进行明确建模，可以简化学习过程，并显著增强网络的表征能力。

算法架构搜索：除上述工作外，放弃人工架构设计、转而寻求自动学习网络结构的研究也有着丰富的历史。这一领域的大部分早期工作都是在神经进化社区进行的，该社区建立了用进化方法搜索整个网络拓扑结构的方法 [25], [26]。虽然进化搜索通常对计算要求很高，但也取得了显著的成功，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的架构[29], [30], [31]。为了减轻这些方法的计算负担，人们提出了基于拉马克继承[32]和可微分架构搜索[33]的高效替代方法。

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]也可用于解决这一问题。拓扑选择作为可能设计结构的路径[37]，以及直接架构预测[38]、[39]已被提出作为其他可行的架构搜索工具。利用强化学习技术[40]、[41]、[42]、[43]、[44]取得的效果尤为显著。SE 模块可用作这些搜索算法的原子构件，并在同时进行的工作 [45] 中被证明在这方面非常有效。

注意力和门控机制：注意力可被解释为一种将可用计算资源偏向于信号中信息量最大成分的分配方式 [46]、[47]、[48]、[49]、[50]、[51]。注意力机制已在许多任务中证明了其实用性，包括序列学习[52]、[53]、图像定位和理解[9]、[54]、图像字幕[55]、[56]和读唇[57]。在这些应用中，它可以作为一个运算符，跟随一个或多个代表更高层次抽象的层，用于模态之间的适应。一些著作对空间注意力和通道注意力的结合使用进行了有趣的研究[58]、[59]。Wang 等人[58]基于沙漏模块[8]引入了一种强大的 "中继-掩码 "注意力机制，该机制被插入深度残差网络的中间阶段。相比之下，我们提出的 SE 模块包括一个轻

量级的门控机制，其重点是通过以计算效率高的方式模拟通道关系来增强网络的表征能力。

3 挤压和激励

挤压激励区块是一个计算单元，可以建立在将输入 $\mathbf{X} \in \mathrm{R}_{H0\times W0\times C0}$ 映射到特征映射 $\mathbf{U} \in \mathrm{R}_{H\times W\times C}$ 的变换 $\mathbf{F}_{tr}$ 上。在接下来的符号中，我们将 F_{tr} 视为卷积算子，并使用 $\mathrm{V} = [\mathrm{v}_1,\mathrm{v}_2,...\mathrm{v}_C]$ 表示学习到的滤波器核集，其中 v_c 指的是第 c 个滤波器的参数。然后，我们可以把输出写成 U = [u1,u2,...,uC]，其中

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

这里，$*$ 表示卷积，$\mathbf{v}_c = [\mathbf{v}_c^1,\mathbf{v}_c^2,...,\mathbf{v}_c^{C0}]$，$\mathbf{X} = [\mathbf{x}^1,\mathbf{x}^2,...,\mathbf{x}^{C0}]$，并且 $\mathrm{u}_c \in \mathbb{R}^{H\times W}$，$\mathbf{v}_c^s$ 是二维空间核，代表 vc 的单通道，作用于 X 的相应通道。由于输出是通过所有通道求和产生的，因此通道相关性隐含在 v_c 中，但与滤波器捕捉到的局部空间相关性纠缠在一起。卷积模拟的信道关系本质上是隐含的和局部的（最顶层的信道关系除外）。我们希望通过明确地模拟信道相互依存关系来加强卷积特征的学习，从而提高网络对信息特征的灵敏度，以便在后续转换中加以利用。因此，我们希望在将滤波器响应输入下一次变换之前，让它能够获取全局信息，并分两个步骤（挤压和激励）对滤波器响应进行重新校准。图 1 是 SE 模块的结构示意图。

3.1 挤压：全局信息嵌入

为了解决利用通道依赖关系的问题，我们首先考虑信号到每个通道的输出特征。每个学习到的滤波器都具有一个局部感受野，因此变换输出 U 的每个单元都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压缩到一个通道描述符中。这是通过使用全局平均池化来生成通道级统计来实现的。形式上，一个统计量 $\mathrm{z}\in\mathrm{R}_C$ 通过其空间维度 H × W 对 U 进行收缩生成，使得 z 的第 c 个元素计算为：

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

讨论：变换 U 的输出可以理解为局部描述符的集合，其统计量对整幅图像具有表达能力。利用这些信息在以前的特征工程工作中普遍存在[60]，[61]，[62]。我们选择最简单的聚合技术，全局平均池化，注意到更复杂的策略也可以在这里使用。

3.2 激励：自适应重标定

为了利用压缩操作中聚合的信息，我们在其后进行了第二个操作，其目的是充分捕获通道依赖关系。为了实现这一目标，该功能必须满足两个标准：第一，它必须是灵活的(特别地,它必须能够学习通道间的非线性相互作用)；第二，它必须学习一种非互斥的关系，因为我们希望确保允许多个渠道被强调(而不是强制执行独热激活)。为了满足这些条件，我们选择使用一个带有 sigmoid 激活的简单门控机制：

$$s = \mathrm{F}_{ex}(z,W) = \sigma(g(z,W)) = \sigma(W_2\delta(W_1 z)), \quad (3)$$

其中 δ 是指 ReLU [63]函数，$\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r}\times C}$，$\mathbf{W}_2 \in \mathbb{R}^{C\times\frac{C}{r}}$. 为了限制模型复杂度和帮助泛化，我们通过在非线性周围形成一个具有两个全连接(FC)层的瓶颈来参数化门机制，即一个具有还原比 r (这个参数的选择在 6.1 节中讨论)的降维层，一个 ReLU，然后一个增维层返回转换输出 U 的通道维度。块的最终输出是通过激活 s 对 U 进行缩放得到的：

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

其中 $\mathbf{X}_e = [\mathbf{x}_{e1},\mathbf{x}_{e2},...,\mathbf{x}_{eC}]$，$\mathbf{F}_{scale}(\mathbf{u}_c,s_c)$是指标量 s_c 和特征映射 $\mathbf{u}_c \in \mathrm{R}^{H\times W}$ 之间的通道级乘法。

讨论：激励算子将输入的特定描述符 z 映射为一组信道权重。就此而言，SE 块内在地引入了以输入为条件的动态，可以将其视为通道上的自注意力函数，其关系不局限于卷积滤波器响应的局部感受野。

3.3 实例分析

SE 模块可以通过在每次卷积后的非线性之后插入来集成到标准架构中，例如 VGGNet [11]。此外，SE 块的灵活性意味着它可以直接应用于标准卷积以外的变换。为了说明这一点，我们通过将 SE 块合并到下面描述的更复杂架构的几个例子中，开发了 SENets。

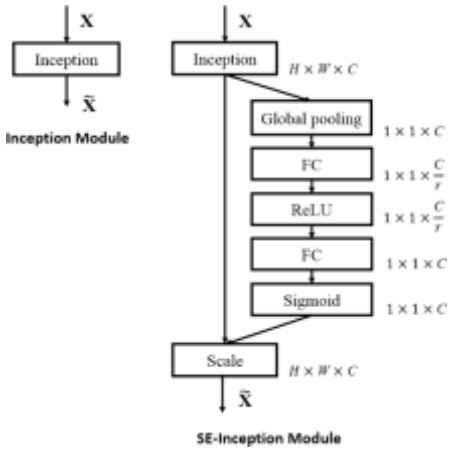


图 2 .原始 Inception 模块(左)和 SEInception 模块(右)的模式。

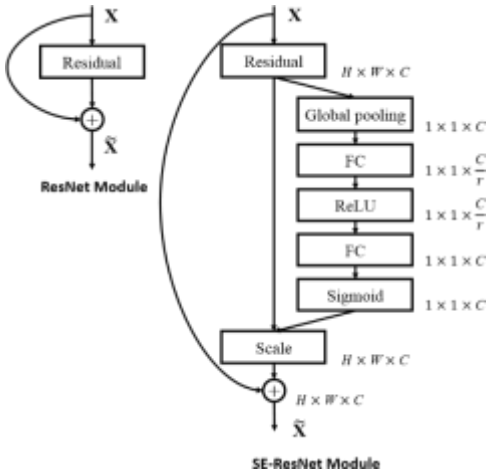


图 3 .原始 Residual 模块(左)和 SEResNet 模块(右)的架构。

我们首先考虑 Inception 网络 SE 模块的构造[5]。在这里，我们简单地将转换 Ftr 看作一个完整的 Inception 模块(见图 2)，并通过对架构中的每个这样的模块进行这种更改，我们得到一个 SE - Inception 网络。 SE 块也可以与残差网络(图 3 描述了一个 SE - ResNet 模块的结构示意图)直接使用。这里取 SE 块变换 Ftr 为残差模块的非恒等分支。挤压和激发都是在与身份分支求和之前起作用。进一步将 SE 块与 ResNeXt [19]、Inception - ResNet [21]、MobileNet [64]和混洗网[65]集成的变体可以按照类似的方案构建。对于 SENet 架构的具体实例，表 1 给出了 SE - ResNet - 50 和 SE - ResNeXt - 50 的详细描述。

SE 块的灵活性带来的一个结果是，有几种可行的方法可以集成到这些架构中。因此，为了评估对用于将 SE 块合并到网络架构中的集成策略的敏感性，我们还在 6.5 节中提供了探索块包含的不同设计的消融实验。

4 模型和计算复杂度

为了使所提出的 SE 块设计具有实际应用价值，它必须在改进的性能和增加的模型复杂度之间提供一个良好的折衷。为了说明与模块相关的计算负担，我们以 ResNet - 50 和 SE - ResNet - 50 为例进行了比较。对于 224 × 224 像素的输入图像，ResNet - 50 在一次前向通道中需要消耗 3.86 GFLOPs（每秒千兆次浮点运算）。每个 SE 块在压缩阶段使用一个全局平均池化操作，在激励阶段使用两个小的 FC 层，然后使用一个便宜的通道尺度缩放操作。总体而言，当设置缩减比例 r (在 3.2 节中介绍)为 16 时，SE - ResNet - 50 所需 GFLOPs 为 3.87 G，相对于原始 ResNet - 50 提升了 0.26 %。为了换取这一点额外的计算负担，SE-ResNet-50 的准确率超过了 ResNet - 50，甚至接近了需要 7.58 GFLOPs 的更深层的 ResNet - 101 网络(表 2)。

在实际应用中，通过 ResNet - 50 进行一次前向和后向传递需要 190 ms，而 SE - ResNet - 50 的训练小批量数据为 256 张图片(两种计时都是在带有 8 个 NVIDIA Titan X GPU 的服务器上进行的)，则需要 209 ms。我们认为这是一个合理的运行时开销，随着全局池化和小的内积操作在流行的 GPU 库中得到进一步优化，这个开销可能会进一步降低。由于 ResNet - 50 对于嵌入式设备应用的重要性，我们进一步测试了每个模型的 CPU 推理时间：对于 224 × 224 像素的输入图像，ResNet - 50 需要 164 ms，而 SE - ResNet - 50 需要 167 ms。我们认为 SE 块产生的较小的额外计算成本是由其对模型性能的贡献所证明的。

我们接下来考虑所提出的 SE 块引入的额外参数。这些额外参数仅来自门控机制的两个 FC 层，因此构成了网络总容量的一小部分。具体地，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 表示缩减比，S 表示阶段(一个阶段是指在一个共同空间维度的特征图上操作的块的集合)的个数，Cs 表示输出通道的维数，Ns 表示阶段 s (当 FC 层使用偏置项时,引入的参数和计算成本通常可以忽略不计)的重复块数。SE - ResNet - 50 在 ResNet - 50 要求的 2 500 万参数之外增加了 2 500 万个参数，相当于增加了 10 %。在实际中，

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 1 (左) Resnet-50 [13]。(中) Se - Resnet - 50。(右) Se-Resnext-50，32 × 4D 模板。在支架内部列出了残余积木块的形状和具有特定参数设置的操作，在外部给出了一个阶段中堆叠块的数量。fc 后面的内括号表示 SE 模块中两个全连接层的输出维度。

	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

表 2：ABLE 2 ImageNet 验证集上的单次裁剪错误率（%）和复杂度比较。原始列指的是原始论文中报告的结果（ResNets 的结果可从网站获取：https://github.com/Kaiminghe/deep-residual-networks）。为了进行公平比较，我们重新训练了基线模型，并在重新实施一栏中报告了得分。SENet 一栏指的是添加了 SE 块的相应架构。括号中的数字表示与重新实现的基线相比的性能改进。† 表示在验证集的非黑名单子集上对模型进行了评估（[21] 对此有更详细的讨论），这可能会略微改善结果。VGG-16 和 SE-VGG-16 采用批量标准化训练。

这些参数大部分来自于网络的最后阶段，其中激励操作是在最多数量的通道中执行的。然而，我们发现，在性能(在 ImageNet 上 Top - 5 误差< 0.1 %)中将相对参数增加降低到~ 4 %的情况下，仅以较小的代价就可以移除 SE 块的这一相对昂贵的最后阶段，这可能证明在参数使用是关键考虑因素(参见第 6.4 节和第 7.2 节进一步讨论)的情况下是有用的。

5 实验

在本节中，我们通过实验来研究 SE 模块在一系列任务、数据集和模型架构中的有效性。

5.1 图像分类

为了评估 SE 块的影响，我们首先在 ImageNet 2012 数据集[10]上进行实验，该数据集包含 128 万张训练图像和来自 1000 个不同类别的 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 错误。

每个基线网络架构及其对应的 SE 对应物都用相同的优化方案进行训练。我们遵循标准做法，使用尺度和纵横比[5]进行随机裁剪的数据增强，大小为 224 × 224 像素(或 299 × 299 用于 Inception - ResNet - v2 和 SE - Inception - ResNet - v2)，并进行随机水平翻转。每个输入图像通过平均 RGB 通道减法进行归一化。所有模型都在我们的分布式学习系统 ROCS 上进行训练，该系统旨在处理大型网络的高效并行训练。使用动量为 0.9，小批量数据大小为 1024 的同步 SGD 进行优化。初始学习率设置为 0.6，每 30 个周期下降 10 倍。模型从头开始训练 100 个周期，使用文献[66]中描述的权重初始化策略。还原比 r (在 3.2 节中)默认设置为(除非另有说明) = 16。

在评估模型时，我们采用中心裁剪，以便从每幅图像中裁剪 224 × 224 像素，然后将其较短的边缘首次调整为 256 (对于 Inception - ResNet -

	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	$31.0_{(1.0)}$	$11.1_{(1.4)}$	142	2.4M

表 3 在 ImageNet 验证集上的单作物错误率(%)和复杂度比较。MobileNet 是指" 1.0 MobileNet-224 "在[64]和 ShuffleNet 是指" ShuffleNet1 × (g = 3) " [65]。括号中的数字表示对重新实现的性能改进。

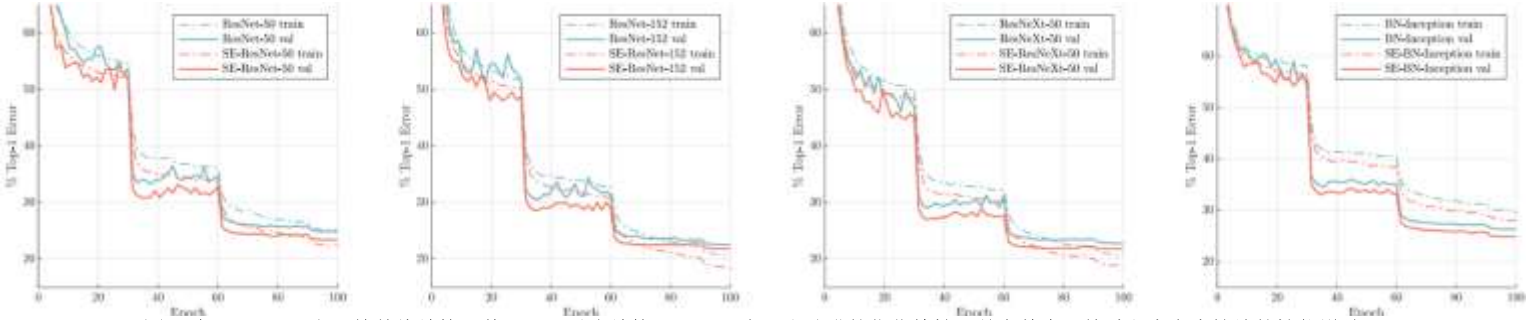


图 4 .在 ImageNet 上训练基线结构及其 SENet 对应结构。SENets 表现出改进的优化特性，并在整个训练过程中产生持续的性能增益。

v2 和 SE - Inception - ResNet - v2 ,首先将较短边缘调整为 352 的每幅图像大小为 299 × 299)。

网络深度：我们首先将 SE - ResNet 与不同深度的 ResNet 结构进行比较，并将结果报告在表 2 中。我们观察到 SE block 在计算复杂度增加极小的情况下，在不同深度上一致地提高了性能。值得注意的是，SE - ResNet - 50 取得了 6.62 %的单作物 top - 5 验证误差，比 ResNet - 50 (7.48 %) 提高了 0.86 %，并且接近于更深层的 ResNet - 101 网络(前 5 位误差为 6.52 %)所取得的性能，仅占总计算量(3.87 GFLOPs vs . 7.58 GFLOPs)的一半。这种模式在更大的深度上重复，其中 SE - ResNet - 101 (前 5 位误差为 6.07 %)不仅匹配，而且比更深的 ResNet - 152 网络(前 5 位误差为 6.34 %)性能提高了 0.27 %。虽然应该注意的是，SE 块本身增加了深度，但它们以极其有效的计算方式做到了这一点，即使在扩展基本架构的深度达到收益递减的点上，也会产生良好的回报。此外，我们看到在不同的网络深度范围内，增益是一致的，这表明由 SE 块引起的改进可能与通过简单地增加基本结构的深度而获得的改进是互补的。

与现代架构的融合：我们接下来研究将 SE 块与另外两种最先进的架构 Inception - ResNet - v2 [21]和 ResNeXt (采用 32 × 4d 的设置) [19]集成的效果，这两种架构都在基础网络中引入了额外的计算构建块。我们构造了这些网络的 SENet 等价类 SE - Inception - ResNet - v2 和 SE - ResNeXt (SE - ResNeXt - 50 的配置见表 1)，并在表 2 中报告了结果。与先前的实验一样，我们观察到在两种架构中引入 SE 块引起的显著性能改善。特别地，SE - ResNeXt - 50 的前 5 个误差为 5.49 %，优于其直接对应物 ResNeXt - 50 (前 5 位误差为 5.90 %)和更深层的 ResNeXt - 101 (前 5 位误差为 5.57 %)，该模型几乎是总参数量和计算开销的两倍。我们注意到我们重新实现的 Inception - ResNet - v2 与文献[21]报道的结果在性能上略有差异。然而，对于 SE 块的效果，我们观察到了类似的趋势，SE 对应的(4.79 %的前 5 位错误)比我们重新实现的 Inception - ResNet - v2 基线(前 5 位误差为 5.21 %)提高了 0.42 %，与文献[21]报道的结果相同。

我们还通过使用 VGG - 16 [11]和 BN - Inception 架构[6]进行实验来评估 SE 块在非残差网络上运行时的效果。为了方便 VGG - 16 从头开始训练，我们在每次卷积后加入 Batch Normalization 层。我们对 VGG - 16 和 SE - VGG - 16 使用相同的训练方案。比较结果见表 2。与已报道的残差基线架构的结果类似，我们观察到 SE 块在非残差设置上带来了性能的提升。

为了深入了解 SE 模块对这些模型优化的影响，在图 4 中描绘了基准体系结构和它们各自的 SE 对应的运行的示例训练曲线。我们观察到 SE 块在整个优化过程中产生稳定的改进。此外，这种趋势在一系列被认为是基线的网络架构中相当一致。

移动设置：最后，我们考虑了 MobileNet [64]和混洗网[65]这两个具有代表性的移动优化网络架构。对于这些实验，我们使用了 256 的

	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

表 4 在 CIFAR - 10 上的分类误差(％)。

	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 5 在 CIFAR - 100 上的分类误差(％)

小批量数据大小和略低于[65]的激进数据增强和正则化。我们使用动量为(设定为 0.9)，初始学习率为 0.1 的 SGD 在 8 个 GPU 上训练模型，每次验证损失平台化时减少 10 倍。整个训练过程需要 400 个历元(使得我们能够重现[65]的基线表现)。表 3 中报告的结果表明 SE 块以最小的计算成本增加为代价，一致地大幅度地提高了精度。

其他数据集：我们接下来考察 SE 块的好处是否泛化到 ImageNet 之外的数据集。我们在 CIFAR - 10 和 CIFAR - 100 数据集上使用几种流行的基线架构和技术(Res Net-110 、Res Net-164 、Wide Res Net-16-8 、Shake - Shake 、Cutout)进行实验[70]。其中包括 50k 个训练和 10k 个测试的 32 × 32 像素 RGB 图像的集合，分别用 10 和 100 类进行标记。SE 模块与这些网络的集成遵循第 3.3 节所述的相同方法。每个基线和它的 SENet 对应物都使用标准的数据增强策略[24]，[71]进行训练。在训练过程中，图像被随机水平翻转并在每侧 4 个像素处填零，然后随机取 32 × 32 的作物。均值和标准差正态化也被应用。训练超参数(例如小批量数据大小,初始学习率,权重衰减等)的设置与原始文献的建议相匹配。我们在表 4 中报告了每个基线及其 SENet 对应物在 CIFAR - 10 上的性能，在表 5 中报告了在 CIFAR - 100 上的性能。我们观察到，在每一个比较中，SENet 都优于基线架构，这表明 SE 块的好处并不局限于 ImageNet 数据集。

5.2 场景分类

我们还在 Places365 - Challenge 数据集[73]上进行了场景分类的实验。该数据集包括 365 个类别的 800 万张训练图像和 36500 张验证图像。相对于分类任务，场景理解任务提供了一种可供选择的评估模型泛化能力和处理抽象能力的方法。

	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**

表 6 单作物错误率(％)在 Places365 验证集上。

	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 7 Faster R - CNN 在 COCO minival 数据集上的目标检测结果(％)。

这是因为它往往需要模型能够处理更复杂的数据关联，并且对更大的外观变化具有鲁棒性。

我们选择使用 Res Net - 152 作为强基线来评估 SE 块的有效性，并遵循文献[72]，[74]中描述的训练和评估协议。在这些实验中，模型都是从头开始训练的。我们将结果报告在表 6 中，并与之前的工作进行了比较。我们观察到 SE - ResNet - 152 (前 5 位误差为 11.01 %)取得了比 ResNet - 152 (前 5 位误差为 11.61 %)更低的验证误差，提供了 SE 块也可以提高单词分类的证据。该 SENet 超过了先前最先进的模型 Places - 365 - CNN [72]，在该任务上排名前 5 的误差为 11.48 %。

5.3 对 COCO 进行检测

我们进一步使用 COCO 数据集[75]评估 SE 块在目标检测任务上的泛化性。与先前的工作[19]一样，我们使用 minival 协议，即在 80k 个训练集和 35k 个 val 子集的并集上训练模型，并在剩余的 5k 个 val 子集上评估模型。权重由在 ImageNet 数据集上训练的模型的参数初始化。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] (即,使用" 2x "学习日程表进行端到端训练)中描述的超参数设置。我们的目标是评估将目标检测器中的主干结构(ResNet)替换为 SE - ResNet 的效果，以便性能的任何变化都可以归因于更好的表示。表 7 报告了使用 ResNet - 50，ResNet - 101 和它们的 SE 对应物作为主干结构的目标检测器的验证集性能。SE - ResNet - 50 在 COCO 标准 AP 指标上比 ResNet - 50 提升了 2.4 % (相对改善 6.3 %)，在 AP @ IoU = 0.5 指标上比 ResNet - 50 提升了 3.1 %。SE 块也有利于更深层的 Res Net - 101 架构，在 AP 指标上实现了 2.0 %的(5 %的相对提高)提升。

	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 8：最新卷积神经网络在作物大小为 224 × 224 和 320 × 320 / 299 × 299 的 ImageNet 验证集上的单作物错误率(%)

	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

表 9：使用更大的作物大小/额外的训练数据在 ImageNet 验证集上与最先进的卷积神经网络进行比较(%)。

综上所述，这组实验证明了 SE 块的通用性。由此引发的改进可以在广泛的架构、任务和数据集上实现。

5.4 ILSVRC 2017 分类竞赛

SENets 奠定了我们在 ILSVRC 竞赛中取得第一名的基础。我们的获胜条目包括一个小型的 SENets 集合，该集合采用了标准的多尺度和多作物融合策略，在测试集上获得了 2.251 %的前 5个误差。作为提交的一部分，我们通过将 SE 块与修改的 ResNeXt [19] (架构详见附录)集成，构建了一个额外的模型 SENet - 154。我们使用标准作物尺寸(224 × 224 和 320 × 320)在表 8 中的 ImageNet 验证集上将该模型与先前的工作进行了比较。我们观察到，SENet - 154 使用 224 × 224 的中心作物评估实现了 18.68 %的 top - 1 误差和 4.47 %的 top - 5 误差，这代表了最强的报告结果。

在挑战之后，ImageNet 基准测试有了进一步的发展。为了进行比较，我们在表 9 中列出了我们目前所知道的最强结果。仅使用 ImageNet 数据的最佳性能最近被报道[79]。该方法利用强化学习在训练过程中开发新的数据增强策略来提高文献[31]中搜索的架构的性能。文献[80]采用 ResNeXt - 101 32 × 48d 架构报道了最佳的综合性能。这是通过在大约 10 亿张弱标记图像上预训练他们的模型并在 ImageNet 上微调来实现的。

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

表 10：在 ImageNet 上的单裁错误率(%)和参数大小为不同缩减比例下的 SE - Res Net - 50。在此，原指采用 ResNet - 50 .

更复杂的数据增强[79]和广泛的预训练[80]所带来的改进可能与我们提出的网络架构的变化相辅相成。

6 讨论研究

在这一部分中，我们进行了消融实验，以更好地了解使用不同配置对 SE 块组件的影响。所有消融实验均在单机(采用 8 个 GPU)上的 ImageNet 数据集上进行。采用 ResNet - 50 作为主干架构。我们在实验中发现，在 Res Net 架构上，去除 FC 层在激励操作中的偏差有利于信道依赖关系的建模，并在后面的实验中使用这种配置。数据增强策略遵循 5.1 节所述方法。为了允许我们研究每个变体的性能上限，学习率被初始化为 0.1，并继续训练，直到验证损失平台 2 (共计 300 个历元)。然后将学习率降低 10 倍，然后重复这个过程(共 3 次)。在训练过程中使用了标签平滑正则化[20]。

6.1 缩减比率

式中引入的折减比例 r(公式 5)是一个超参数，它允许我们改变网络中 SE 块的容量和计算成本。为了考察该超参数介导的性能和计算成本之间的权衡，我们使用 SE - ResNet - 50 进行了一系列不同 r 值的实验。表 10 的对比表明，性能对一定范围的缩减比例具有稳健性。复杂度的增加并不会单调地提高性能，而较小的比例会显著增加模型的参数规模。设置 r = 16 在精度和复杂度之间取得了较好的平衡。在实际中，在整个网络中使用相同的比率可能不是最优的(由于不同层次发挥的作用不同)，因此可以通过调整比率来满足给定基础体系结构的需要。

6.2 挤压算子

我们研究了使用全局平均池化（global average pooling）而不是全局最大池化（global max pooling）作为挤压操作符（squeeze operator）

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

表 11：在 SE - ResNet - 50 上使用不同的压缩算子的效果采用 ImageNet (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**

表 12：SE - ResNet - 50 中的激励算子使用不同非线性度对 ImageNet (错误率%)的影响。

的重要性（由于这种方法效果良好，我们没有考虑更复杂的替代方案）。结果在表 11 中报告。虽然最大池化和平均池化都有效，但平均池化实现了略好的性能，这证明了将其作为挤压操作的基础选择的合理性。然而，我们注意到 SE 块的性能对于特定聚合操作符的选择是相当稳健的。

6.3 激励算子

接下来，我们评估了激励机制非线性度的选择。我们考虑另外两个选项：ReLU 和 tanh，并用这些可选的非线性项替换 sigmoid 进行实验。结果见表 12。我们看到，将 sigmoid 替换为 tanh 略微恶化了性能，而使用 ReLU 则急剧恶化，事实上导致 SE - ResNet - 50 的性能低于 ResNet - 50 基线。这表明对于 SE 块要有效，仔细构造激发算子是重要的。

6.4 不同阶段引入 SE 块

我们通过将 SE 块逐级集成到 ResNet-50 中，探索 SE 块在不同阶段的影响。在 ResNet-50 中一次集成一个 SE 区块。具体来说，我们在中间阶段添加 SE 区块：阶段 2、阶段 3 和阶段 4。并在表 13 中报告了结果。我们发现，在 ResNet-50 的每个阶段引入 SE 块时，在架构的每
个阶段引入时都会带来性能优势。此外，在不同阶段引入 SE 模块所带来的增益是互补的。是互

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 13 在不同阶段将 SE 块与 ResNet-50 集成对 ImageNet 的效果（错误率%）。

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 24 不同 SE 块集成策略与 ResNet-50 对 ImageNet 的效果（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

表 15 在 ResNet-50 每个残差分支的 3x3 卷积层整合 SE 块对 ImageNet 的影响（错误率 %）。ResNet-50 的残差分支对 ImageNet 的影响（误差率 %）。

补的，因为它们可以有效地结合起来，进一步提高网络性能。

6.5 整合战略

最后，我们进行了一项消融研究，以评估将 SE 区块整合到现有架构时 SE 区块位置的影响。除了建议的 SE 设计外，我们还考虑了三种变体：(1) SE-PRE 块，其中 SE 块被移到残差单元之前；(2) SE-POST 块，其中 SE 单元被移到与身份分支相加之后（ReLU 之后）；(3) SE-Identity 块，其中 SE 单元被置于与残差单元并行的身份连接上。图 5 展示了这些变体，表 14 报告了每个变体的性能。我们观察到，SE-PRE、SE-Identity 和建议的 SE 模块性能相似，而使用 SE-POST 模块则导致性能下降。该实验表明，只要在分支聚合之前应用 SE 单元，SE 单元所产生的性能改进对其位置相当稳健。

在上述实验中，每个 SE 块都被置于残差单元结构之外。我们还构建了一个设计变体，将 SE 块移至残差单元内部，直接置于 3 × 3 卷积层之后。由于 3 × 3 卷积层的通道数较少，相应的 SE 模块引入的参数数也减少了。表 15 中的比较显示，与标准 SE 模块相比，SE 3×3 变体以更少的参数达到了相当的分类精度。虽然这超出了本文的研究范围，但我们预计，通过针对特定架构调整 SE 块的使用，还能进一步提高效率。

7 SE 块的作用

尽管所提出的 SE 代码块已被证明可以提高网络在多项视觉任务中的表现，但我们还希望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到的表征进行严格的理论分析仍然具有挑战性，因此我们采

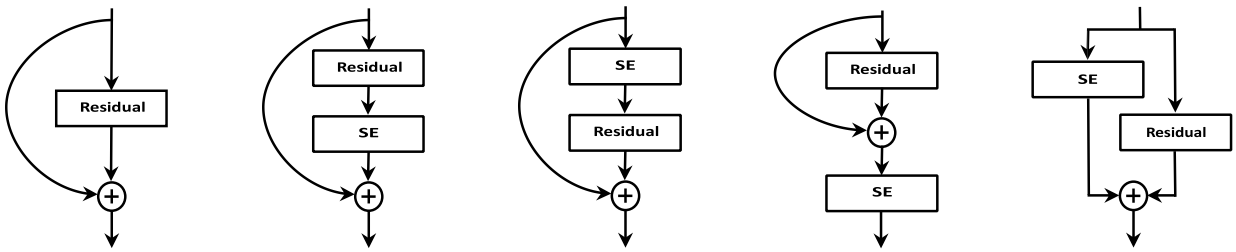


图 5. 消融研究中探讨的 SE 区块整合设计。

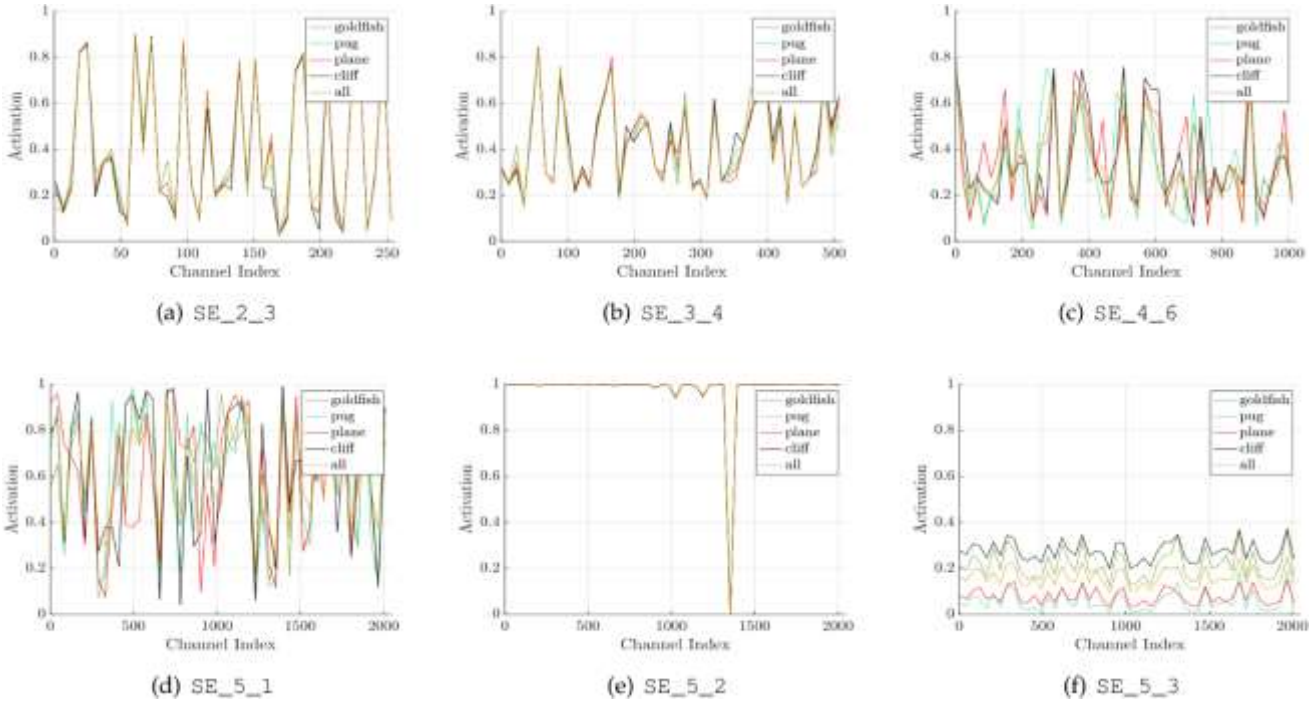


图 6：在 ImageNet 的 SE-ResNet-50 中，激发算子在不同深度引起的激活。每组激活都按照以下方案命名： SE_stageID_blockID。除了 SE_5_2 处的异常行为外，随着深度的增加，激活越来越具有类别特异性。

	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

表 16：挤压操作符对 ImageNet 的影响（错误率 %）

用实证方法来研究 SE 块所扮演的角色，目的是至少对其实际功能有一个初步的了解。

7.1 挤压效果

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，我们取消了池化操作，并在激励算子中用具有相同通道维度的相应 1 × 1 卷积来替换两个 FC 层，即 NoSqueeze，其中激励输出保持了作为输入的空间维度。与 SE 块不同的是，这些点向卷积只能将通道重映射为局部算子输出的函数。虽然在实践中，深度网络的后几层通常会拥有一个（理论上的）全局感受野，但在 NoSqueeze 变体中，全局嵌入不再能在整个网络中直接访问。表 16 比较了这两种模型与标准 ResNet-50 模型的精度和计算复杂度。我们发现，全局信息的使用对模型性能有很大影响，这突出了挤压操作的重要性。此外，与 "无挤压 "设计相比，"挤压 "模块允许以计算简便的方式使用全局信息。

7.2 激励的作用

为了更清楚地了解激发算子在 SE 块中的功能，我们将在本节中研究 SE-ResNet-50 模型中的激活示例，并考察它们在网络不同深度的不同类别和不同输入图像中的分布情况。特别是，我们希望了解不同类别的图像以及同一类别中不同图像的兴奋是如何变化的。

我们首先考虑不同类别的激励分布。具体来说，我们从 ImageNet 数据集中抽取了四个显示语义和外观多样性的类别，即金鱼、哈巴狗、飞机和悬崖（这些类别的示例图像见附录）。然后，

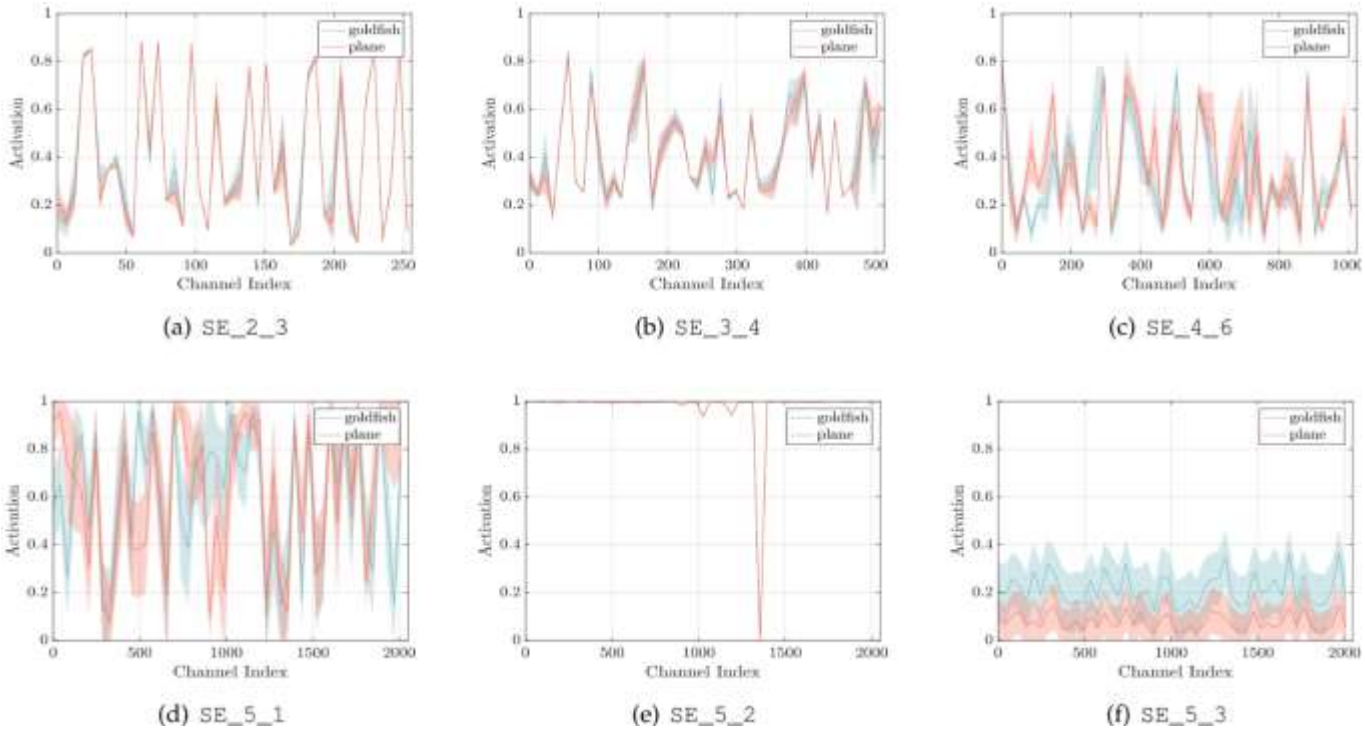


图 7 SE-ResNet-50 的不同模块在 ImageNet 的金鱼类和平面类图像样本上受到激发后产生的激活。模块名为 "SE_stageID_blockID"

我们从验证集中为每个类别抽取 50 个样本，在每个阶段的最后一个 SE 块（紧接着下采样之前）计算 50 个均匀采样通道的平均激活度，并将其分布绘制在图 6 中。作为参考，我们还绘制了所有 1000 个类别的平均激活分布图。

关于激励操作的作用，我们有以下三点看法。首先，在网络的前几层，如 SE 2 3，不同类别的分布非常相似。这表明，在早期阶段，不同类别可能共享特征通道的重要性。第二个观察结果是，随着深度的增加，每个通道的价值变得更加针对不同类别，因为不同类别对特征的区分价值表现出不同的偏好，例如 SE 4 6 和 SE 5 1。这些观察结果与之前的研究结果[81]、[82]一致，即较早层的特征通常更具一般性（例如，在分类任务的背景下与类别无关），而较晚层的特征则表现出更高的特异性[83]。

接下来，我们在网络的最后阶段观察到了一些不同的现象。SE 5 2 显示出一种有趣的饱和状态趋势，在这种状态下，大部分激活都接近于 1。当所有激活值都为 1 时，SE 块就会还原为同一算子。在 SE 5 3 的网络末端（紧接着是分类器之前的全局池化先验），不同类别出现了类似的模式，直至规模发生适度变化（可由分类器调整）。这表明，SE 5 2 和 SE 5 3 在为网络提供重新校准方面的重要性低于之前的区块。这一发现与第 4 节中的实证调查结果显示的结果一致，即通过在最后阶段移除 SE 块，可以显著减少额外的参数数量，而性能损失微乎其微。

最后，我们在图 7 中展示了两个样本类别（金鱼和飞机）中同一类别图像实例激活的平均值和标准偏差。我们观察到了与类间可视化一致的趋势，表明 SE 块的动态行为在类和类内实例中都有所不同。特别是在网络的后几层，单个类别内的表征具有相当大的多样性，网络学会利用特征重新校准来提高其辨别性能[84]。总之，SE 模块会产生针对特定实例的响应，但其功能是在架构的不同层支持模型日益增长的针对特定类别的需求。

8 结论

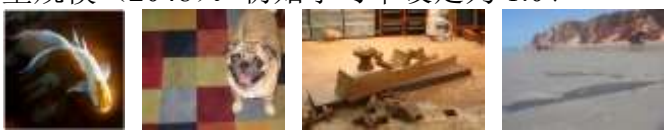
在本文中，我们提出了 SE 块，这是一个架构单元，旨在通过使网络能够执行动态信道特征重新校准来提高网络的表征能力。大量实验表明，SENets 在多个数据集和任务中都取得了最先进的性能。此外，SE 块还揭示了以前的架构无法充分模拟信道特征依赖性的问题。我们希望这一见解能在其他需要强鉴别特征的任务中发挥作用。最后，SE 块产生的特征重要性值可能会用于其他任务，如模型压缩的网络剪枝。

致谢

作者感谢 Momenta 公司的李超和王光远在训练系统优化和 CIFAR 数据集实验中做出的贡献。我们还要感谢 Andrew Zisserman、Aravindh Mahendran 和 Andrea Vedaldi 在讨论中给予的帮助。这项工作部分得到国家自然科学基金资助（61632003、61620106003、61672502、61571439）、中国国家重点研发计划（2017YFB1002701）和澳门 FDCT 资助（068/2015/A2）的支持。塞缪尔-阿尔巴尼（Samuel Albanie）得到了英国工程与物理学研究中心（EPSRC）AIMS CDT EP/L015897/1 的资助。

附录： Senet-154 的详细情况

SENet-154 是在 64×4d ResNeXt-152 的改进版中加入 SE 块而构建的，ResNeXt-101[19]采用了 ResNet-152 [13]的块堆叠策略，扩展了原来的 ResNeXt-101[19]。除了使用 SE 块之外，该模型在设计和训练上的其他不同之处如下： (a) 每个瓶颈构建块的第一个 1 × 1 卷积通道的数量减半，以降低模型的计算成本，同时将性能降低到最低程度。(b) 第一个 7 × 7 卷积层被替换为三个连续的 3 × 3 卷积层。(c) 1 × 1 下采样投影与步长-2 卷积被 3 × 3 步长-2 卷积取代，以保留信息。(d) 在分类层之前插入一个剔除层（剔除率为 0.2），以减少过拟合。(e) 在训练过程中使用了 Labelsmoothing 正则化（如文献 [20] 所介绍）。(f) 所有 BN 层的参数在最后几个训练历元都被冻结，以确保训练和测试的一致性。(g) 训练由 8 台服务器（64 个 GPU）并行执行，以实现较大的批量规模（2048）。初始学习率设定为 1.0、



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

图 8. 7.2 节所述实验中使用的 ImageNet 四类图像样本。

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. Dollar, 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, “Inceptionv4, 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-toend 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, "Multilevel 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. Dollar, and C. L. Zitnick, "Microsoft COCO: Common objects in´ context," in *ECCV*, 2014.

[76] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollar, and K. He, "De-´ tectron," 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.