MobileNetV2: 倒残差和线性瓶颈

Mark Sandler Andrew Howard Menglong Zhu Andrey Zhmoginov Liang-Chieh Chen Google Inc.

{sandler, howarda, menglong, azhmogin, lcchen}@google.com

# 摘要

本文介绍了一种新的移动架构--MobileNetV2，它提高了移动模型在多种任务和基准测试中的性能，并跨越了不同模型大小的范围。我们还介绍了在我们称之为 SSDLite 的新框架中将这些移动模型应用于物体检测的有效方法。此外，我们还演示了如何通过 DeepLabv3 的简化形式（我们称之为 Mobile DeepLabv3）来构建移动语义分割模型。

基于一种倒置残差结构，其中的捷径连接位于薄瓶颈层之间。中间扩展层使用轻量级深度卷积来过滤作为非线性来源的特征。此外，我们还发现，为了保持表征能力，必须消除窄层中的非线性。我们证明这可以提高性能，并提供了导致这种设计的直觉。

最后，我们的方法允许将输入/输出域与转换的表现力解耦，这为进一步分析提供了一个方便的框架。我们在 ImageNet [1] 分类、COCO 物体检测 [2] 和 VOC 图像分割 [3] 上测量了我们的性能。我们评估了准确性和以乘加（MAdd）衡量的操作次数之间的权衡，以及实际延迟和参数数量。

# 导言

神经网络给机器智能的许多领域带来了革命性的变化，使具有挑战性的图像识别任务具有超人的准确性。然而，提高准确性的动力往往是有代价的：现代最先进的网络需要大量的计算资源，超出了许多移动和嵌入式应用的能力。

arXiv:1801.04381v4 [cs.CV] 21 Mar 2019

本文介绍了一种专为移动和资源受限环境定制的新型神经网络架构。我们的网络在保持相同精度的同时，大幅减少了操作次数和所需内存，从而推动了移动定制计算机视觉模型的技术发展。

我们的主要贡献在于一个新颖的层模块：具有线性瓶颈的反转残差。该模块将低维压缩表示作为输入，首先将其扩展到高维，并通过轻量级深度卷积进行过滤。随后，通过线性卷积将特征投影回低维表示。正式实现作为 TensorFlow-Slim 模型库的一部分，见 [4]。

该模块可以在任何现代框架中使用标准操作高效地实现，并使我们的模型在使用标准基准测试的多个性能点上超越了最先进的技术。此外，这种卷积模块特别适用于移动设计，因为它从不将大型中间张量完全实体化，从而大大减少了推理过程中所需的内存占用。这就减少了许多嵌入式硬件设计中对主存储器访问的需求，这些设计提供了少量非常快速的软件控制高速缓冲存储器。

# 相关工作

调整深度神经架构，在准确性和性能之间达到最佳平衡，是过去几年中活跃的研究领域。众多团队进行的人工架构搜索和训练算法改进，使 AlexNet [5]、VGGNet [6]、GoogLeNet [7]、ResNet [8]等早期设计有了显著改进。以及 ResNet [8]。最近，在算法架构探索方面取得了很多进展，包括超参数优化 [9、10、11] 以及各种网络剪枝方法 [12、13、14、15、16、17] 和连接性学习 [18、19]。还有大量工作致力于改变内部卷积块的连接结构，如 ShuffleNet [20] 或引入稀疏性 [21] 等 [22]。

最近，[23, 24, 25, 26]开辟了一个新方向，将遗传算法和强化学习等优化方法引入建筑搜索。然而，这样做的一个缺点是，最终生成的网络非常复杂。在本文中，我们的目标是更好地直观了解神经网络是如何运行的，并以此指导尽可能简单的网络设计。我们的方法应被视为对 [23] 中描述的方法及相关工作的补充。在这方面，我们的方法与 [20, 22] 所采用的方法类似，可以进一步提高性能，同时让人们了解其内部运作。我们的网络设计基于 MobileNetV1 [27]。它保持了其简洁性，不需要任何特殊的操作员，同时显著提高了准确性，在移动应用的多种图像分类和检测任务方面达到了最先进的水平。

# 序言、讨论和直觉

## 深度可分离卷积

深度可分离卷积是许多高效神经网络架构的关键构件 [27, 28, 20]，我们在本研究中也采用了这种架构。其基本思想是用一个因式分解版本取代完整的卷积算子，将卷积分成两个独立的层。第一层称为深度卷积，它通过对每个输入通道应用一个卷积滤波器来执行轻量级滤波。第二层是 1 × 1 卷积，称为点式卷积，负责通过计算输入通道的线性组合来构建新特征。

标准卷积采用 *hi* × *wi* × *di* 输入张量 *Li*，并应用卷积核 K∈R*k*×*k*×*di*×*dj* 生成 *hi* × *wi* × *dj* 输出张量 *Lj*。标准卷积层的计算成本为 *hi* · *wi* · *di* · *dj* · *k* · *k*.。

深度可分离卷积是标准卷积层的直接替代品。根据经验，它们的效果几乎与普通卷积一样好，但成本却很低：

*hi* · *wi* · *di*(*k*2 + *dj*) (1)

是深度卷积和 1 × 1 点向卷积之和。与传统层相比，深度可分离卷积的计算量减少了近 *k*2 倍。MobileNetV2 使用 k = 3（3×3 深度可分离卷积），因此计算成本是标准卷积的 8 到 9 倍，而精度却只有很小的下降[27]。

## 线性瓶颈

考虑一个由 n 个层 *Li*组成的深度神经网络，每个层都有一个尺寸为 *hi* × *wi* × *di*的激活张量。在本节中，我们将讨论这些激活张量的基本特性，并将其视为具有 di 维度的 *hi* × *wi* "像素 "的容器。非正式地讲，对于一组输入的真实图像，我们认为层激活的集合（对于任意层 *Li*）构成了一个 "感兴趣的流形"。长期以来，人们一直认为神经网络中的兴趣流形可以嵌入低维子空间。换句话说，当我们观察深度卷积层的所有单个 d 通道像素时，这些值中编码的信息实际上位于某个流形中，而该流形又可以嵌入到一个低维子空间中。

乍一看，只要降低层的维度，从而降低操作空间的维度，就能捕捉和利用这一事实。MobileNetV1 [27] 成功地利用了这一点，通过一个宽度乘数参数在计算量和精确度之间进行了有效权衡，并将其纳入了其他网络的高效模型设计中 [20]。根据这一直觉，宽度乘数方法可以降低激活空间的维度，直到感兴趣的流形横跨整个空间。然而，当我们回想起深度卷积神经网

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | Output/dim=2 | Output/dim=3 | Output/dim=5 | Output/dim=15 | Output/dim=30 |

络实际上具有非线性坐标变换(如 ReLU)时，这种直觉就不攻自破了。例如，ReLU 应用于一维空间中的一条直线时，会产生一条 "射线"，而在 R*n* 空间中，一般会产生一条具有 n 个关节的片断线性曲线。

图 1：嵌入高维空间的低维流形的 ReLU 变换示例。在这些示例中，使用随机矩阵 T 将初始螺旋嵌入 n 维空间，然后进行 ReLU 变换，再使用 T -1 将其投影回二维空间。在上述 n = 2、3 的例子中，流形的某些点会相互塌陷，导致信息丢失；而对于 n = 15 到 30 的例子，变换是高度非凸的。

不难看出，一般情况下，如果层变换 ReLU(Bx) 的结果具有非零体积 S，则映射到内部 S 的点是通过输入的线性变换 B 得到的，这表明与全维输出相对应的输入空间部分仅限于线性变换。换句话说，深度网络只有在输出域的非零体积部分才具有线性分类器的能力。更正式的说明请参阅补充材料。图示

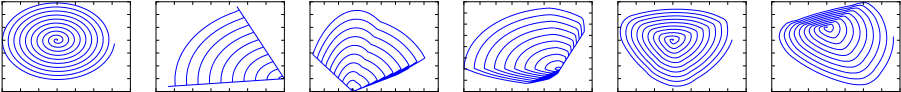
描述已自动生成

图 2：可分离卷积块的演变。对角线阴影纹理表示不包含非线性的层。最后一层（浅色）表示下一个区块的开始。注：2d 和 2c 叠加后是等效图块。最佳彩色效果。

上图翻译：(a) 规则 (b) 可分离 (c) 可分离，带线性瓶颈 (d) 瓶颈，带扩展层

另一方面，当 ReLU 崩溃通道时，它不可避免地会丢失该通道中的信息。但是，如果我们有很多通道，而激活流形中存在一种结构，那么其他通道中的信息可能仍然会被保留下来。在补充材料中，我们展示了如果输入流形可以嵌入到激活空间的一个明显低维的子空间中，那么 ReLU 变换在保留信息的同时，也为可表达函数集引入了所需的复杂性。

图示

描述已自动生成总之，我们强调了两个特性，这两个特性表明，所关注的流形应位于高维激活空间的低维子空间中：

图 3：残差块 [8, 30] 与反转残差的区别。对角线处的图层不使用非线性。我们用每个区块的厚度来表示其通道的相对数量。请注意典型残差是如何连接通道数较多的图层，而倒残差则是如何连接通道数较少的图层。以彩色显示效果最佳。

1. 如果相关流形在经过 ReLU 变换后仍保持非零体积，则对应于线性变换。
2. ReLU 能够保留输入流形的完整信息，但前提是输入流形位于输入空间的低维子空间中。

这两点启示为我们提供了优化现有神经架构的经验提示：假设感兴趣的流形是低维的，我们可以通过在卷积块中插入线性瓶颈层来捕捉这一点。实验证据表明，使用线性层是至关重要的，因为它可以防止非线性因素破坏过多信息。在第 6 节中，我们通过实验证明，在瓶颈层中使用非线性层确实会使性能降低几个百分点，从而进一步验证了我们的假设。我们注意到，[29]中也有类似的报告称非线性对性能有帮助，该报告将非线性从传统残差块的输入中移除，从而提高了 CIFAR 数据集的性能。

在本文的其余部分，我们将使用瓶颈卷积。我们将输入瓶颈大小与内部大小的比值称为扩展比。

## 反向残差

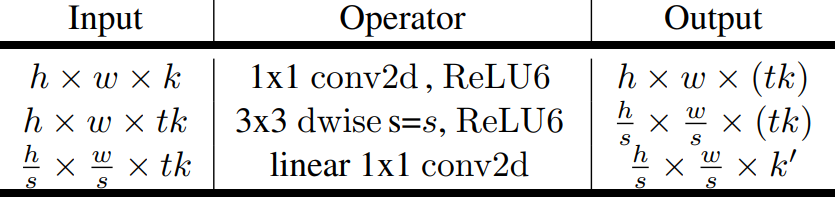
瓶颈块看起来与残差块类似，每个块包含一个输入，然后是几个瓶颈，接着是扩展[8]。然而，直觉告诉我们，瓶颈区块实际上包含了所有必要信息，而扩展层仅仅是张量非线性变换的一个实现细节，因此我们在瓶颈区块之间直接使用了快捷方式。 图 3 以示意图的形式展示了设计中的不同之处。插入捷径的动机与经典的残差连接类似：我们希望提高梯度在乘法器层间传播的能力。不过，倒置设计的内存效率要高得多（详见第 5 节），而且在我们的实验中效果略好。

表 1：从 k 个信道到 k` 个信道的瓶颈残差块转换，跨距为 s，扩展因子为 t。

**瓶颈卷积的运行时间和参数数量** 基本实现结构如表 1 所示。对于大小为 h × w、扩展因子为 t、内核大小为 k、输入通道为 d `、输出通道为 d`` 的数据块，所需的乘加总数为 h · w · d` · t(d` + k2 + d``)。与(1)相比，这个表达式多了一个项，因为我们确实多了一个 1 × 1 的卷积，但我们网络的性质允许我们使用更小的输入和输出维度。在表 3 中，我们比较了 MobileNetV1、MobileNetV2 和 ShuffleNet 每种分辨率所需的尺寸。

## 信息流解释

我们的架构有一个有趣的特性，那就是它将构件（瓶颈层）的输入/输出域与层转换（将输入转换为输出的非线性函数）自然分开。前者可视为网络各层的容量，而后者可视为网络的表现力。这与传统的卷积块（包括规则卷积块和可分离卷积块）形成鲜明对比，在传统卷积块中，表现力和容量纠缠在一起，是输出层深度的函数。

特别是，在我们的案例中，当内层深度为 0 时，由于快捷连接的存在，底层卷积是标识函数。当扩展比小于 1 时，这是一个经典的残差卷积块 [8，30]。不过，就我们的目的而言，我们证明大于 1 的扩展比是最有用的。

这种解释使我们能够将网络的表现力与其容量分开研究，我们认为有必要进一步探讨这种分离，以便更好地了解网络的特性。

# 4. 模型架构

现在，我们将详细介绍我们的架构。如上一节所述，基本构件是一个瓶颈深度分离的残差卷积。该模块的详细结构如表 1 所示。MobileNetV2 的架构包含有 32 个滤波器的初始完全卷积层，以及表 2 中描述的 19 个残差瓶颈层。我们使用 ReLU6 作为非线性，因为它在使用低精度计算时具有鲁棒性[27]。我们始终使用现代网络标准的内核大小 3×3，并在训练过程中使用 dropout 和批量归一化。

除第一层外，我们在整个网络中使用恒定的扩展率。在实验中，我们发现 5 到 10 之间的扩展率会产生几乎相同的性能曲线，规模较小的网络在扩展率稍小的情况下性能更好，而规模较大的网络在扩展率较大的情况下性能稍好。

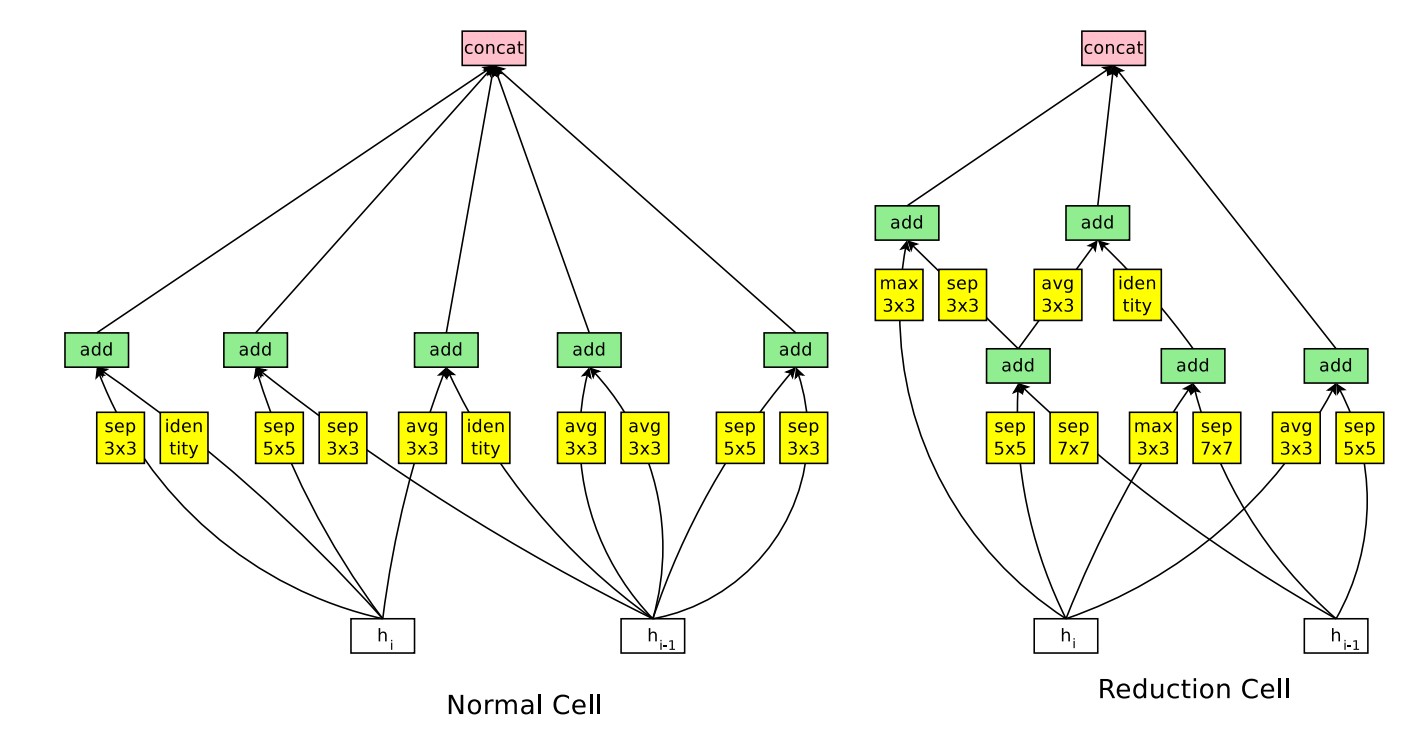
在所有主要实验中，我们对输入张量的大小使用了 6 的扩展因子。例如，瓶颈层接收 64 个通道的输入张量并生成 128 个通道的张量，那么中间扩展层就是 64 - 6 = 384 个通道。

**权衡超参数** 与文献[27]一样，我们将输入图像的分辨率和宽度乘数作为可调超参数，根据所需的精度/性能权衡进行调整，从而使我们的架构适应不同的性能点。我们的主要网络（宽度乘数 1，224 × 224）的计算成本为 3 亿次乘法加法，使用 340 万个参数。我们对 96 到 224 的输入分辨率和 0.35 到 1.4 的宽度乘数进行了性能权衡。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | Operator | *t* | *c* | *n* | *s* |
| 2242 × 3 | conv2d | - | 32 | 1 | 2 |
| 1122 × 32 | bottleneck | 1 | 16 | 1 | 1 |
| 1122 × 16 | bottleneck | 6 | 24 | 2 | 2 |
| 562 × 24 | bottleneck | 6 | 32 | 3 | 2 |
| 282 × 32 | bottleneck | 6 | 64 | 4 | 2 |
| 142 × 64 | bottleneck | 6 | 96 | 3 | 1 |
| 142 × 96 | bottleneck | 6 | 160 | 3 | 2 |
| 72 × 160 | bottleneck | 6 | 320 | 1 | 1 |
| 72 × 320 | conv2d 1x1 | - | 1280 | 1 | 1 |
| 72 × 1280 | avgpool 7x7 | - | - | 1 | - |
| 1 × 1 × 1280 | conv2d 1x1 | - | k | - |  |

表 2：MobileNetV2：每行描述 1 个或多个相同（模数跨距）层的序列，重复 n 次。同一序列中的所有层都有相同数量的 c 个输出通道。每个序列的第一层具有跨距 s，其他层均使用跨距 1。所有空间卷积都使用 3 × 3 内核。扩展因子 t 始终应用于输入大小，如表 1 所述。

图 4：不同架构的卷积块比较。ShuffleNet 使用组卷积 [20] 和洗牌，它还使用传统的残差方法，即内部块比输出块窄。ShuffleNet 和 NasNet 插图分别来自相关论文。



[

a) NasNet

(

23

]

input

Dwise 3x3,

stride=s, Relu6

conv 1x1, Relu6

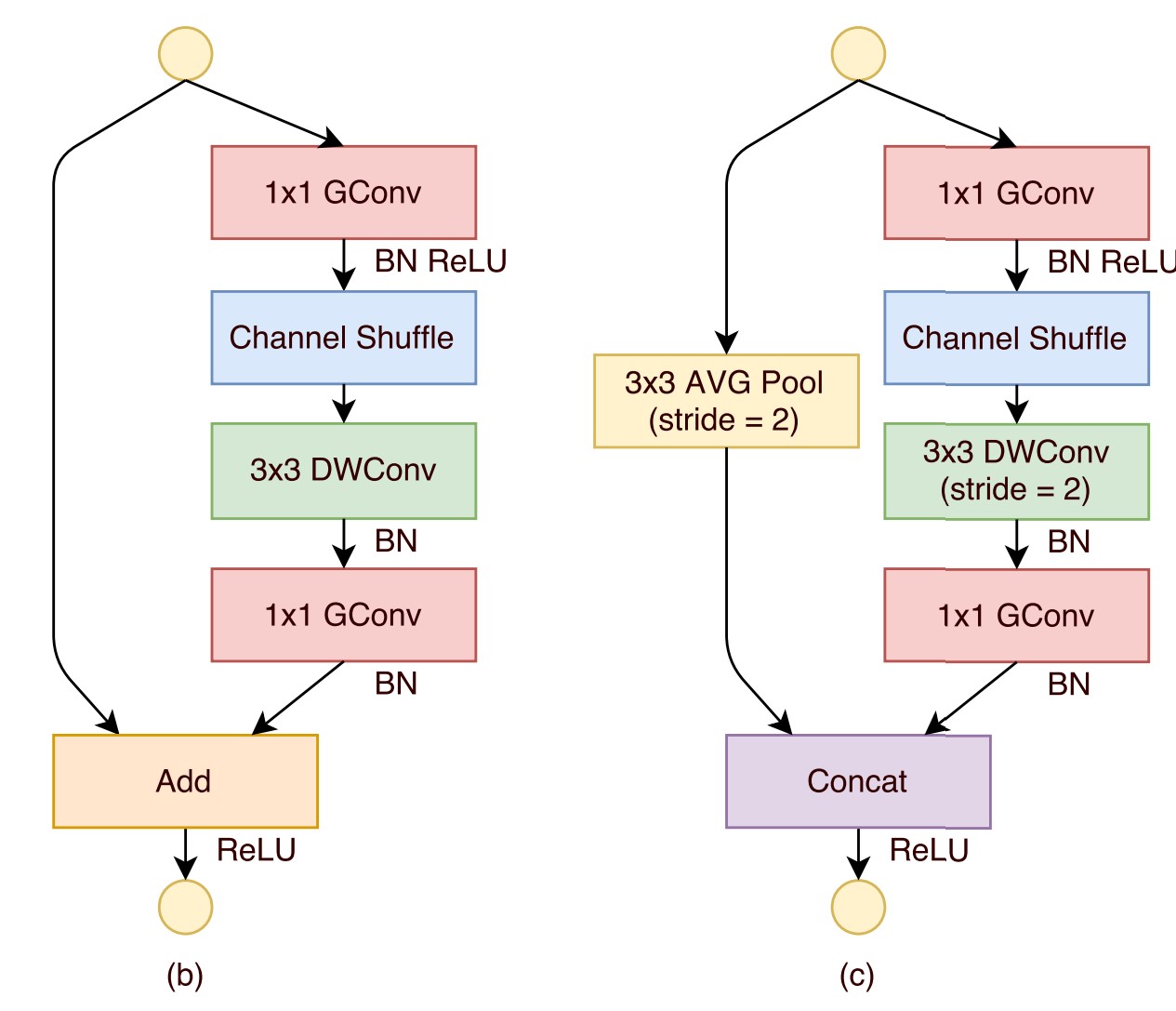
[

b) MobileNet

(

27

]



(

[

c) ShuffleNet

20

]

Conv 1x1, Relu6

Dwise 3x3, Relu6

input

conv 1x1, Linear

Add

Conv 1x1, Relu6

Dwise 3x3,

stride=2, Relu6

input

conv 1x1, Linear

Stride=1 block

Stride=2 block

(

d) Mobilenet V

2

|  |  |  |  |
| --- | --- | --- | --- |
| Size | MobileNetV1 | MobileNetV2 | ShuffleNet  (2x,g=3) |
| 112x112 | 64/1600 | 16/400 | 32/800 |
| 56x56 | 128/800 | 32/200 | 48/300 |
| 28x28 | 256/400 | 64/100 | 400/600K |
| 14x14 | 512/200 | 160/62 | 800/310 |
| 7x7 | 1024/199 | 320/32 | 1600/156 |
| 1x1 | 1024/2 | 1280/2 | 1600/3 |
| Max | 1600K | 400K | 600K |

表 3：不同架构下每种空间分辨率需要实体化的最大通道数/内存（Kb）。我们假设激活为 16 位浮点数。对于 ShuffleNet，我们使用 2x，g = 3，这与 MobileNetV1 和 MobileNetV2 的性能相匹配。对于 MobileNetV2 和 ShuffleNet 的第一层，我们可以采用第 5 节所述的技巧来减少内存需求。尽管 ShuffleNet 在其他地方使用了瓶颈，但由于非瓶颈张量之间存在捷径，因此非瓶颈张量仍然需要实体化。

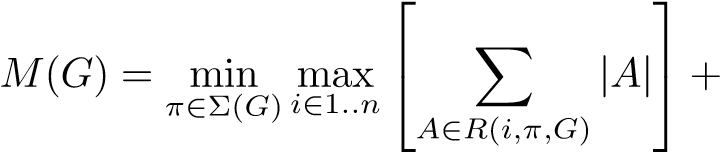
网络计算成本从 7 次乘法加法到 5.85 亿次乘法加法不等，而模型大小从 1.7 百万到 6.9 百万个参数不等。

与[27]的一个细微差别是，对于乘数小于 1 的情况，除了最后一个卷积层外，我们对所有层都应用了宽度乘数。这提高了较小模型的性能。

# 实施说明

## 图示, 示意图 描述已自动生成高效内存推理

倒置残差瓶颈层允许使用 TensorFlow[31] 或 Caffe[32] 等高效内存实现推理，建立一个有向无环计算超图 G，由代表操作的边和代表中间计算张量的节点组成。计算调度的目的是尽量减少需要存储在内存中的张量总数。在最一般的情况下，它会搜索所有可能的计算顺序 Σ(G)，并挑选出最小化计算顺序 Σ(G)的那个。

size(*πi*)*.*

其中，R(i,π,G) 是连接到 πi ...πn 节点中任意一个节点的中间张量列表，|A| 表示张量 A 的大小，size(i) 是操作 i 期间内部存储所需的内存总量。

对于只有琐碎并行结构（如残差连接）的图，只有一个非琐碎的可行计算顺序，因此可以简化计算图 G 的推理所需的内存总量和边界：

或者重述一下，内存量就是所有操作的输入和输出组合的最大总大小。在下文中，我们将展示，如果我们将瓶颈残差块视为单一操作（并将内部卷积视为一次性张量），内存总量将由瓶颈张量的大小所支配，而不是由瓶颈内部张量的大小（大得多）所支配。

图表

描述已自动生成**瓶颈残差块** 图 3b 所示的瓶颈块算子 F(x) 可以表示为三个算子的组合 F(x) = [A ◦ N ◦ B]x，其中 A 是线性变换 A : R*s*×*s*×*k* → R*s*×*s*×*n*，N 是每个信道的非线性变换：N : R*s*×*s*×*n* → R *s’*× *s’*×*n*，B 同样是输出域的线性变换：B : R*s’*× *s’*×*n* → R *s’* × *s’* ×*k’*。

图 6：非线性和各类捷径（残余）连接的影响。

左图为瓶颈层非线性的影响。

右图为残块变化的影响。

对于我们的网络，N = ReLU6 ◦ dwise ◦ ReLU6，但结果适用于任何按通道变换。假设输入域的大小为 |x|，输出域的大小为 |y|，那么计算 F(X) 所需的内存可以低至 |*s2*k| + | *s’* 2 *k ’*| + O(max(*s*2*,s*’2))。

手机屏幕截图

中度可信度描述已自动生成该算法基于以下事实：内部张量 I 可以表示为每个张量大小为 n/t 的张量的连接，因此我们的函数可以表示为

通过累加总和，我们只需要在内存中始终保留一个大小为 n/t 的中间块。使用 n = t 时，我们最终只需始终保持中间表示的一个通道。使我们能够使用这一技巧的两个限制条件是：(a) 内部变换（包括非线性和深度变换）是按通道进行的；(b) 连续的非按通道运算符的输入大小与输出大小之比很大。对于大多数传统神经网络来说，这种技巧不会产生明显的改进。

我们注意到，使用 t 向分割计算 F(X)所需的乘加运算符数与 t 无关，但在现有实现中，我们发现用几个较小的乘加运算符代替一个矩阵乘法会因缓存丢失增加而损害运行性能。我们发现，当 t 是一个介于 2 和 5 之间的小常数时，这种方法最有帮助。它大大降低了内存需求，但仍能让我图表, 散点图

描述已自动生成们利用深度学习框架提供的高度优化矩阵乘法和卷积算子所获得的大部分效率。特殊的框架级优化是否会带来进一步的运行时改进，我们拭目以待。

图 5：MobileNetV2 与 MobileNetV1、ShuffleNet 和 NAS 的性能对比曲线。对于我们的网络，所有分辨率均使用乘数 0.35、0.5、0.75 和 1.0，224 分辨率使用额外的 1.4。最佳彩色观看效果。

# 6. 实验

## ImageNet 分类

**训练设置** 我们使用 TensorFlow[31] 训练模型。我们使用标准的 RMSPropOptimi-zer，衰减和动量均设置为 0.9。我们在每一层后使用批量归一化，标准权重衰减设置为 0.00004。按照 MobileNetV1[27] 的设置，我们使用的初始学习率为 0.045，学习率衰减率为每 epoch 0.98。我们使用 16 个 GPU 异步工作者，批量大小为 96。

**成果** 我们将我们的网络与 MobileNet-V1、ShuffleNet 和 NASNet-A 模型进行了比较。表 4 显示了几个选定模型的统计数据，图 5 显示了全部性能图。

## 物体检测

我们评估并比较了 MobileNetV2 和 MobileNetV1 作为特征提取器[33]，在 COCO 数据集[2]上使用改进版单次检测器（SSD）[34]进行物体检测的性能。我们还以 YOLOv2 [35] 和原始 SSD（以 VGG-16 [6] 为基础网络）为基准进行了比较。由于我们的重点是移动/实时模型，因此我们没有与 Faster-RCNN [36] 和 RFCN [37] 等其他架构进行性能比较。

**轻型SSD**: 本文介绍了常规 SSD 的移动友好型变体。在 SSD 预测层中，我们用可分离卷积（深度卷积，然后是 1 × 1 投影）取代了所有常规卷积。这种设计与 MobileNets 的整体设计一致，而且计算效率更高。我们称这种改进版本为 SSDLite。如表 5 所示，与普通 SSD 相比，SSDLite 大大减少了参数数量和计算成本。

对于 MobileNetV1，我们沿用了 [33] 中的设置。对于 MobileNetV2，SSDLite 的第一层附在第 15 层的扩展层上（输出跨度为 16）。SSDLite 的第二层和其他层附着在最后一层（输出跨度为 32）之上。这种设置与 MobileNetV1 一致，因为所有层都附着在相同输出跨度的特征图上。

两个 MobileNet 模型都使用开源 TensorFlow 对象检测 API [38]进行训练和评估。两个模型的输入分辨率均为 320 × 320。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network | Top1 | Params | MAdds | CPU |
| MobileNetV1 | 70.6 | 4.2M | 575M | 113ms |
| ShuffleNet (1.5) | 71.5 | 3.4M | 292M | - |
| ShuffleNet (x2) | 73.7 | 5.4M | 524M | - |
| NasNet-A | 74.0 | 5.3M | 564M | 183ms |
| MobileNetV2 | 72.0 | 3.4M | 300M | 75ms |
| MobileNetV2 (1.4) | 74.7 | 6.9M | 585M | 143ms |

表 4：不同网络在 ImageNet 上的性能比较。按照操作惯例，我们计算了乘法加法的总数。在最后一列中，我们以毫秒（ms）为单位报告了 Google Pixel 1 手机（使用 TF-Lite）单个大核心的运行时间。我们没有报告 ShuffleNet 的数据，因为目前还不支持高效的分组卷积和洗牌。

|  |  |  |
| --- | --- | --- |
|  | Params | MAdds |
| SSD[34] | 14.8M | 1.25B |
| SSDLite | 2.1M | 0.35B |

表 5：配置 MobileNetV2 并对 80 个类别进行预测的 SSD 和 SSDLite 的大小和计算成本比较。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network | mAP | Params | MAdd | CPU |
| SSD300[34] | 23.2 | 36.1M | 35.2B | - |
| SSD512[34] | 26.8 | 36.1M | 99.5B | - |
| YOLOv2[35] | 21.6 | 50.7M | 17.5B | - |
| MNet V1 + SSDLite | 22.2 | 5.1M | 1.3B | 270ms |
| MNet V2 + SSDLite | 22.1 | 4.3M | 0.8B | 200ms |

表 6：MobileNetV2 + SSDLite 和其他实时检测器在 COCO 数据集物体检测任务上的性能比较。MobileNetV2 + SSDLite 以明显较少的参数和较小的计算复杂度实现了具有竞争力的准确度。所有模型均在 trainval35k 上进行训练，并在 test-dev 上进行评估。SSD/YOLOv2 数字来自 [35]。运行时间是使用 TF-Lite 引擎的内部版本，针对谷歌 Pixel 1 手机的大核心报告的。

我们对两个模型的 mAP（COCO 挑战指标）、参数数和乘加数进行了基准测试和比较。结果如表 6 所示。MobileNetV2 SSDLite 不仅是最高效的模型，也是三种模型中最准确的。值得注意的是，在 COCO 数据集上，MobileNetV2 SSDLite 的效率比 YOLOv2 高 20 倍，体积比 YOLOv2 小 10 倍。

## 语义分割

在本节中，我们将比较 MobileNetV1 和 MobileNetV2 模型作为特征提取器与 DeepLabv3 [39]，用于移动语义分割任务。DeepLabv3 采用阿特鲁斯卷积[40, 41, 42]，这是一种明确控制计算特征图分辨率的强大工具，并构建了五个并行头，包括（a）阿特鲁斯空间金字塔池化模块（ASPP）[43]，其中包含三个具有不同阿特鲁斯速率的 3 × 3 卷积；（b）1 × 1 卷积头；以及（c）图像级特征[44]。我们用输出步长表示输入图像空间分辨率与最终输出分辨率的比率，该比率通过适当应用无齿卷积来控制。对于语义分割，我们通常采用输出跨距 = 16 或 8 来获得更密集的特征图。我们在 PASCAL VOC 2012 数据集[3]上进行了实验，并使用了[45]中的额外注释图像和评估指标 mIOU。

为了建立移动模型，我们尝试了三种不同的设计方案：(1) 不同的特征提取器；(2) 简化 DeepLabv3 头以加快计算速度；(3) 不同的推理策略以提高性能。表 7 总结了我们的结果。我们观察到(a) 包括多尺度输入和添加左右翻转图像在内的推理策略会显著增加 MAdds，因此不适合在设备上应用；(b) 使用 output stride = 16 比 output stride = 8 更高效；(c) MobileNetV1 已经是一个功能强大的特征提取器，只需要比 ResNet-101 [8] 少约 4.9 - 5.7 倍的 MAdds（例如，mIOU：78.56 vs. ResNet-101）、mIOU: 78.56 vs 82.70，MAdds: 941.9B vs 4870.6B），（d）在 MobileNetV2 倒数第二层特征图之上构建 DeepLabv3 head 比在原始的最后一层特征图之上构建 DeepLabv3 head 更有效，因为倒数第二层特征图包含 320 个通道，而不是 1280 个通道。(e) DeepLabv3 头的计算成本很高，去掉 ASPP 模块可显著降低 MAdds，但性能略有下降。在表 7 的最后，我们确定了一个潜在的设备上应用候选方案（粗体字），该方案可实现 75.32% 的 mIOU，仅需 2.75B MAdds。

## 消融研究

**反向残余连接**。残差连接的重要性已得到广泛研究 [8, 30, 46]。本文报告的新结果是，连接瓶颈层的捷径比连接扩展层的捷径性能更好（对比见图 6b）。

**线性瓶颈的重要性。** 严格来说，线性瓶颈模型不如非线性模型强大，因为只要适当改变偏置和缩放比例，激活总是可以在线性机制下运行。然而，图 6a 所示的实验表明，线性瓶颈提高了性能，为非线性破坏低维空间信息提供了支持。

# 表格 描述已自动生成结论和今后的工作

表 7：MobileNet + DeepLabv3 在 PASCAL VOC 2012 验证集上的推理策略。MNet V2\*：倒数第二个特征图用于 DeepLabv3 头，其中包括：(1) Atrous Spatial Pyramid Pooling (ASPP) 模块；(2) 1 × 1 卷积以及图像池特征。OS：输出跨距，用于控制分割图的输出分辨率。MF：测试时的多尺度和左右翻转输入。所有模型都在 COCO 上进行了预训练。用于设备应用的潜在候选模型以粗体显示。PASCAL 图像的尺寸为 512 × 512，无绳卷积使我们可以在不增加参数数量的情况下控制输出特征分辨率。

我们描述了一个非常简单的网络架构，它使我们能够建立一系列高效的移动模型。我们的基本构建单元有几个特性，使其特别适合移动应用。它允许非常高效的记忆推理，并利用所有神经框架中的标准操作。

在 ImageNet 数据集方面，我们的架构在广泛的性能点上都提高了技术水平。

在 COCO 数据集上的物体检测任务中，我们的网络在准确性和模型复杂度方面都优于最先进的实时检测器。值得注意的是，与 YOLOv2 相比，我们的架构与 SSDLite 检测模块相结合，计算量减少了 20 倍，参数减少了 10 倍。

在理论方面：所提出的卷积块有一个独特的特性，可以将网络的表现力（由扩展层编码）与其容量（由瓶颈输入编码）分开。探索这一点是未来研究的一个重要方向。

# 致谢

我们要感谢 Matt Streeter 和 Sergey Ioffe 的有益反馈和讨论。

**8 参考文献**

1. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252, December 2015. 1
2. Tsung-Yi Lin, Michael Maire, Serge Belongie,James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C Lawrence Zitnick. Microsoft COCO:Common objects in context. In *ECCV*, 2014. 1, 7
3. Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserma. The pascal visual object classes challenge a retrospective. *IJCV*, 2014. 1, 8
4. Mobilenetv2 source code. Available from [https://github.com/tensorflow/ models/tree/master/research/slim/ nets/mobilenet.](https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet) 1
5. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Bartlett et al. [48], pages 1106–1114. 1
6. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1, 7
7. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society, 2015. 1
8. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 1, 3, 4, 8
9. James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012. 1
10. Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In Bartlett et al. [48], pages 2960–2968. 1
11. Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Md. Mostofa Ali Patwary, Prabhat, and Ryan P. Adams. Scalable bayesian optimization using deep neural networks. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2171–2180. JMLR.org, 2015. 1
12. Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In Stephen Jose Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]*, pages 164–171. Morgan Kaufmann, 1992. 2
13. Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 598– 605. Morgan Kaufmann, 1989. 2
14. Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1135–1143, 2015. 2
15. Song Han, Jeff Pool, Sharan Narang, Huizi Mao,Shijian Tang, Erich Elsen, Bryan Catanzaro, John Tran, and William J. Dally. DSD: regularizing deep neural networks with dense-sparse-dense training flow. *CoRR*, abs/1607.04381, 2016. 2
16. Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1379–1387, 2016. 2
17. Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016. 2
18. Karim Ahmed and Lorenzo Torresani. Connectivity learning in multi-branch networks. *CoRR*, abs/1709.09582, 2017. 2
19. Tom Veniat and Ludovic Denoyer. Learning timeefficient deep architectures with budgeted super networks. *CoRR*, abs/1706.00046, 2017. 2
20. Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CoRR*, abs/1707.01083, 2017. 2, 5
21. Soravit Changpinyo, Mark Sandler, and Andrey Zhmoginov. The power of sparsity in convolutional neural networks. *CoRR*, abs/1702.06257, 2017. 2
22. Min Wang, Baoyuan Liu, and Hassan Foroosh. Design of efficient convolutional layers using single intra-channel convolution, topological subdivisioning and spatial ”bottleneck” structure. *CoRR*, abs/1608.04337, 2016. 2
23. Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017. 2, 5
24. Lingxi Xie and Alan L. Yuille. Genetic CNN. *CoRR*, abs/1703.01513, 2017. 2
25. Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2902–2911. PMLR, 2017. 2
26. Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016. 2
27. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. 2, 4, 5, 6
28. Francois Chollet. Xception: Deep learning with depthwise separable convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2
29. Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. *CoRR*, abs/1610.02915, 2016. 3
30. Saining Xie, Ross B. Girshick, Piotr Dollar,´ Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016. 3, 4, 8
31. Mart´ın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro,Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane,´ Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wat-´ tenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 5, 6
32. Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 5
33. Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017. 7
34. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 7
35. Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016. 7
36. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 7
37. Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. Rfcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. 7
38. Jonathan Huang, Vivek Rathod, Derek Chow, Chen Sun, and Menglong Zhu. Tensorflow object detection api, 2017. 7
39. Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017. 7
40. Matthias Holschneider, Richard KronlandMartinet, Jean Morlet, and Ph Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets: Time-Frequency Methods and Phase Space*, pages 289–297. 1989. 7
41. Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann Le-¨ Cun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv:1312.6229*, 2013. 7
42. George Papandreou, Iasonas Kokkinos, and PierreAndre Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *CVPR*, 2015. 7
43. Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2017. 7
44. Wei Liu, Andrew Rabinovich, and Alexander C. Berg. Parsenet: Looking wider to see better. *CoRR*, abs/1506.04579, 2015. 7
45. Bharath Hariharan, Pablo Arbelaez,´ Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 8
46. Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. 8
47. Guido Montufar, Razvan Pascanu, Kyunghyun´ Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS’14, pages 2924–2932, Cambridge, MA, USA, 2014. MIT Press. 13
48. Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Leon Bottou, and Kilian Q.´ Weinberger, editors. *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 36, 2012, Lake Tahoe, Nevada, United States*, 2012. 9

# A. 瓶颈改造

在本节中，我们将研究算子 AReLU(Bx) 的特性，其中 x∈R*n* 表示 n 个信道像素，B 是 m × n 矩阵，A 是 n × m 矩阵。我们认为，如果 m≤ n，这种形式的变换只能以损失信息为代价利用非线性。相反，如果 ，这种变换可以是高度非线性的，但仍有很高的可逆概率（对于初始随机权重）。

首先，我们证明 ReLU 对于位于其图像内部的任何一点来说都是同一变换。

**例题 1** 设 S(X) = {ReLU(x)|x∈X} 。如果 S(X) 的卷非零，则 interiorS(X) ⊆ X。

**证明**: 设 S*’* = interiorReLU(S)。事实上，ReLU 的图像不包含坐标为负的点，坐标为零的点也不可能是内部点。因此，对于每个 x∈ S*’*，x = ReLU(x) 如所愿。

由此可见，对于交错线性变换和 ReLU 算子的任意组合，如果它保留了非零量，那么在这种组合中保留的输入空间 X 的那部分就是线性变换，因此很可能对深度网络的能力贡献不大。然而，这种说法并不可靠。事实上，如果输入流形可以嵌入 (n - 1) 维流形（总共有 n 个维度），那么该 Lemma 就完全正确，因为起始体积为 0。

由于 ReLU(x) 非线性是一个将整个 x ≤ 0 的射线映射为 0 的射函数，因此在神经网络中使用这种非线性会导致信息丢失。一旦 ReLU 将输入流形的一个子集折叠为更小维度的输出，接下来的网络层就无法再区分折叠后的输入样本。在下文中，我们将证明具有足够大扩展层的瓶颈网络可以抵御因 ReLU 激活函数的存在而造成的信息丢失。

**定理 2（ReLU 的可逆性）**考虑算子 ReLU(Bx)，其中 B 是 m × n 矩阵，x∈ Rn。对于某个 x0 ∈ Rn，让 y­0 = ReLU(Bx0)，那么方程 y0 = ReLU(Bx)关于 x 有唯一解，当且仅当 y0 至少有 n 个非零值，且 B 中有 n 行线性独立的行与 y0 的非零坐标相对应。

**证明：** Denote the set of non-zero coordinates of *y*0 as *T* and let *yT* and *BT* be restrictions of *y* and *B* to the subspace defined by *T*. If |*T*| *< n*, we have *yT* = *BTx*0 where *BT* is under-determined with at least one solution *x*0, thus there are infinitely many solutions. Now consider the case of |*T*| ≥ *n* and let the rank of *BT* be *n*. Suppose there is an additional solution *x*1 6= *x*0 such that *y*0 = ReLU(*Bx*1), then we have, which cannot be satisfied unless One of the corollaries of this lemma says that if *n*, we only need a small fraction of values of *Bx* to be positive for ReLU(*Bx*) to be invertible.

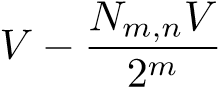
The constraints of the lemma 2 can be empirically validated for real networks and real inputs and hence we can be assured that information is indeed preserved. We further show that with respect to initialization, we can be sure that these constraints are satisfied with high probability. Note that for random initialization the conditions of lemma 2 are satisfied due to initialization symmetries. However even for trained graphs these constraints can be empirically validated by running the network over valid inputs and verifying that all or most inputs are above the threshold. On Figure 7 we show how this distribu-

tion looks for different MobileNetV2 layers. At step 0 the activation patterns concentrate around having half of the positive channel (as predicted by initialization symmetries). For fully trained network, while the standard deviation grew significantly, all but the two layers are still above the invertibility thresholds. We believe further study of this is warranted and might lead to helpful insights on network design.

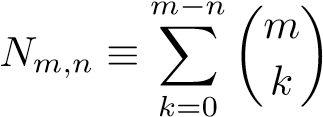
Theorem 1 *Let S be a compact n-dimensional submanifold of* R*n. Consider a family of functions fB*(*x*) = ReLU(*Bx*) *from* R*n to* R*m parameterized by m* × *n matrices B* ∈ B*. Let p*(*B*) *be a probability density on the space of all matrices* B *that satisfies:*

* *P*(*Z*) = 0 *for any measure-zero subset Z* ⊂ B*;*
* *(a symmetry condition) p*(*DB*) = *p*(*B*) *for any B* ∈ B *and any m* × *m diagonal matrix D with all diagonal elements being either* +1 *or* −1*.*

*Then, the average n-volume of the subset of S that is collapsed by fB to a lower-dimensional manifold is*

*,*

*where V* = vol*S and*

*.*

Proof: For any *σ* = (*s*1*,...,sm*) with *sk* ∈ {−1*,*+1}, let *Qσ* = {*x* ∈ R*m*|*xisi >* 0} be a corresponding quadrant in R*m*. For any *n*-dimensional submanifold Γ ⊂ R*m*, ReLU acts as a bijection on Γ ∩ *Qσ* if *σ* has at least *n* positive values[[1]](#footnote-1) and contracts Γ ∩ *Qσ* otherwise. Also notice that the intersection of *BS* with R*m*\(∪*σQσ*) is almost surely (*n*−1)-dimensional. The average *n*-volume of *S* that is not collapsed by applying ReLU to *BS* is therefore given by:

X

E*B*[*Vσ*(*B*)]*,* (3)

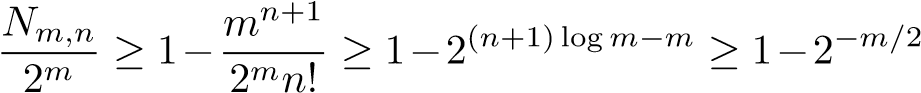
*σ*∈Σ*n*

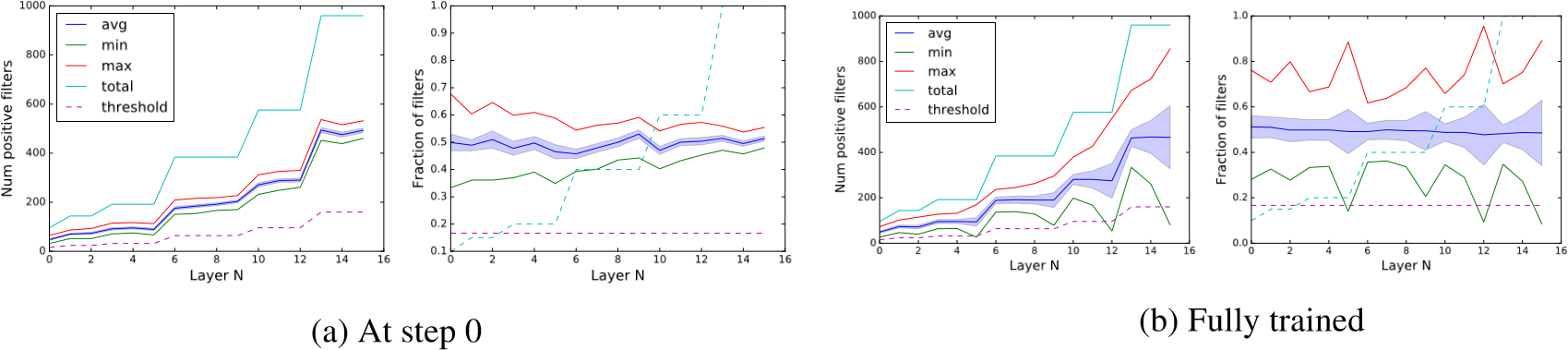
where Σ*n* = {(*s*1*,...,sm*)|P*k θ*(*sk*) ≥ *n*}, *θ* is a step function and *Vσ*(*B*) is a volume of the largest subset of *S* that is mapped by *B* to *Qσ*. Now let us calculate E*B*[*Vσ*(*B*)]. Recalling that *p*(*DB*) = *p*(*B*) for any *D* = diag(*s*1*,...,sm*) with *sk* ∈ {−1*,*+1}, this average can be rewritten as E*B*E*D*[*Vσ*(*DB*)]. Noticing that the subset of *S* mapped by *DB* to *Qσ* is also mapped by *B* to *D*−1*Qσ*, we immediately obtain

Figure 7: Distribution of activation patterns. The *x*-axis is the layer index, and we show minimum/maximum/average number of positive channels after each convolution with ReLU. *y*-axis is either absolute or relative number of channels. The “threshold” line indicates the ReLU invertibility threshold - that is the number of positive dimensions is higher than the input space. In our case this is 1*/*6 fraction of the channels. Note how at the beginning of the training on Figure 7a the distribution is much more tightly concentrated around the mean. After the training has finished (Figure 7b), the average hasn’t changed but the standard deviation grew dramatically. Best viewed in color.

P*σ*0 *Vσ*[diag(*σ*0)*B*] = P*σ*0 *Vσ*0[*B*] = vol*S* and therefore E*B*[*Vσ*(*B*)] = 2−*m* vol*S*. Substituting this and  into Eq. 3 concludes the proof.

Notice that for sufficiently large expansion layers with, the fraction of collapsed space *Nm,n/*2*m* can be bounded by:



and therefore ReLU(*Bx*) performs a nonlinear transformation while preserving information with high probability.

We discussed how bottlenecks can prevent manifold collapse, but increasing the size of the bottleneck expansion may also make it possible for the network to represent more complex functions. Following the main results of [47], one can show, for example, that for any integer *L* ≥ 1 and *p >* 1 there exist a network of *L* ReLU layers, each containing *n* neurons and a bottleneck expansion of size *pn* such that it maps *pnL* input volumes (linearly isomorphic to [0*,*1]*n*) to the same output region [0*,*1]*n*. Any complex possibly nonlinear function attached to the network output would thus effectively compute function values for *pnL* input linear regions.

# B. Semantic segmentation visualization results

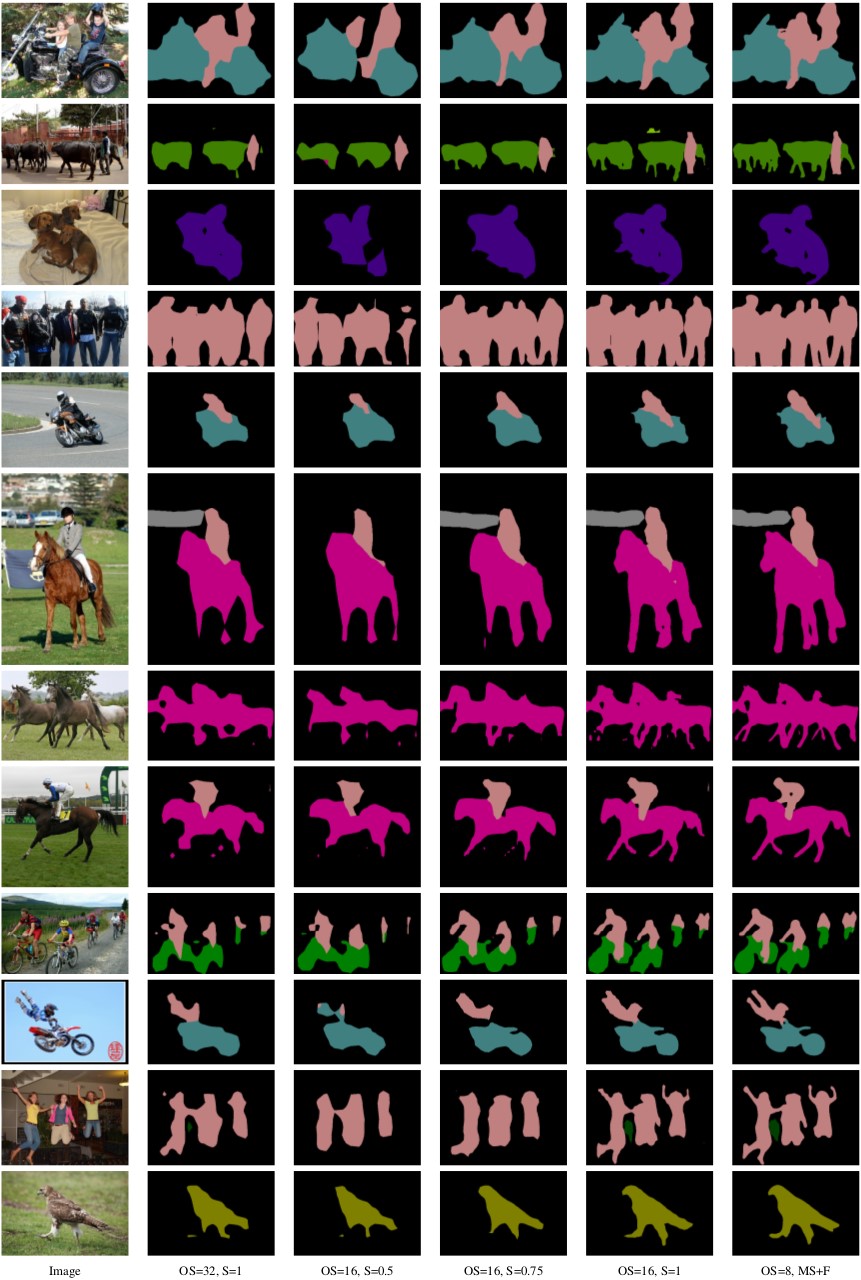


Figure 8: MobileNetv2 semantic segmentation visualization results on PASCAL VOC 2012 *val* set. OS: *output stride*. S: single scale input. MS+F: Multi-scale inputs with scales = {0*.*5*,*0*.*75*,*1*,*1*.*25*,*1*.*5*,*1*.*75} and left-right flipped inputs. Employing *output stride* = 16 and single input scale = 1 attains a good trade-off between FLOPS and accuracy.

1. unless at least one of the positive coordinates for all *x*∈ Γ∩*Qσ* is fixed, which would not be the case for almost all *B* and Γ = *BS* [↑](#footnote-ref-1)