探索 MobileNetV3
Searching for MobileNetV3

Andrew Howard[1] Mark Sandler[1] Grace Chu[1] Liang-Chieh Chen[1] Bo Chen[1] Mingxing Tan[2]
Weijun Wang[1] Yukun Zhu[1] Ruoming Pang[2] Vijay Vasudevan[2] Quoc V. Le[2] Hartwig Adam[1]
Google AI, Google Brain
{howarda,sandler,cxy,lcchen,bochen,tanmingxing,weijunw, yukun, rpang, vrv, qvl, hadam}@google.com

摘要

我们在结合互补搜索技术和新颖架构设计的基础上，推出了下一代 MobileNets。MobileNetV3 结合了硬件感知网络架构搜索（NAS）和 NetAdapt 算法，并通过新颖的架构设计加以改进，从而适应手机 CPU。本文开始探讨自动搜索算法和网络设计如何协同工作，以利用互补方法改善整体技术水平。通过这一过程，我们创建了两个新的 MobileNet 模型并发布：MobileNetV3-Large 和 MobileNetV3-Small，分别针对高和低资源用例。然后，我们对这些模型进行了调整，并将其应用于物体检测和语义分割任务。对于语义分割（或任何密集像素预测）任务，我们提出了一种新的高效分割解码器 Lite Reduced Atrous Spatial Pyramid Pooling (LR-ASPP)。我们在移动分类、检测和分割方面取得了新的成果。与 MobileNetV2 相比，MobileNetV3-Large 在 ImageNet 分类上的准确率提高了 3.2%，同时延迟降低了 20%。MobileNetV3-Small 的准确率比 MobileNetV2 模型高出 6.6%，延迟也相当。在 COCO 检测方面，MobileNetV3-Large 的检测速度比 MobileNetV2 快 25%，准确率基本相同。在城市景观分割方面，MobileNetV3-Large LRASPP 比 MobileNetV2 R-ASPP 快 34%，准确率却相差无几。

1. 引言

高效的神经网络在移动应用中正变得无处不在，带来了全新的设备体验。神经网络也是个人隐私的重要保障，用户无需将数据发送到服务器进行评估，就能获得神经网络的优势。神经网络效率的提高不仅能通过更高的精度和更低的延迟改善用户体验，还能通过降低功耗来延长电池寿命。

本文介绍了我们开发 MobileNetV3 大型和小型模型的方法，目的是提供下一代高精度高效神经网络模型，为设备上的计算机视觉提供支持。新的网络推动了技术的进步，并展示了如何将自动搜索与新的架构进步相结合，从而建立有效的模型。

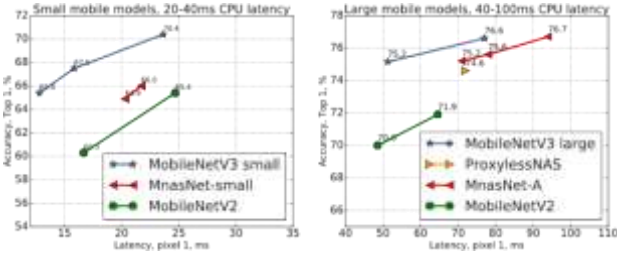

图 1.像素 1 延迟与 ImageNet Top-1 准确性之间的权衡。所有模型均使用输入分辨率 224。V3 large 和 V3 small 使用乘数 0.75、1 和 1.25 来显示最佳边界。所有延迟都是在同一设备的单个大型内核上使用 TFLite[1] 测得的。MobileNetV3-Small 和 Large 是我们提出的下一代移动模型。

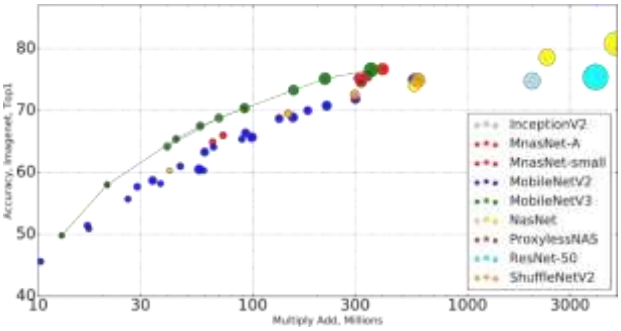

图 2.图名为准确率 vs 乘加操作数 vs 模型大小，该图为 MAdds 与 top-1 精度之间的权衡。这样就可以比较针对不同硬件或软件框架的模型。所有 MobileNetV3 输入分辨率均为 224，使用乘数 0.35、0.5、0.75、1 和 1.25。其他分辨率请参见第 6 节。以彩色显示效果最佳。

本文的目标是开发出最佳的移动计算机视觉架构，以优化移动设备上的精度和延迟权衡。为此，我们引入了（1）互补搜索技术；（2）适用于移动环境的新型高效非线性版本；（3）新型高效网络设计；（4）新型高效分割解码器。我们通过全面的实验证明了每种技术的功效和价值，并对各种使用情况和移动电话进行了评估。

本文的结构如下。第 2 节首先讨论了相关工作。第 3 节回顾了用于移动模型的高效构件。第 4 节回顾了架构搜索以及 MnasNet 和 NetAdapt 算法的互补性。第 5 节介绍了改进联合搜索模型效率的新型架构设计。第 6 节介绍了分类、检测和分割方面的大量实验，以展示功效并了解不同元素的贡献。第 7 节包含结论和未来工作。

2. 相关工作

近年来，设计深度神经网络架构以实现精度和效率之间的最佳权衡一直是一个活跃的研究领域。新颖的手工结构和算法神经架构搜索都在推动这一领域的发展方面发挥了重要作用。 SqueezeNet[22]广泛使用 1x1 卷积与挤压和扩展模块，主要侧重于减少参数数量。最近的研究将重点从减少参数转向减少操作次数（MAdds）和实际测量的延迟。MobileNetV1[19] 采用深度可分离卷积，大大提高了计算效率。在此基础上，MobileNetV2[39] 引入了具有反转残差和线性瓶颈的资源效率块。ShuffleNet[49] 利用分组卷积和信道洗牌操作进一步降低了 MAdds。CondenseNet[21]在训练阶段学习群卷积，以保持层间有用的密集连接，从而实现特征重用。ShiftNet[46]提出了与点卷积交错的移位操作，以取代昂贵的空间卷积。

为了实现架构设计过程的自动化，强化学习（RL）被首次引入，用于搜索具有竞争精度的高效架构[53, 54, 3, 27, 35]。一个完全可配置的搜索空间会以指数级增长，非常难处理。因此，早期的架构搜索工作主要集中在单元级结构搜索，并且在所有层中重复使用相同的单元。最近，[43] 探索了一个块级分层搜索空间，允许在网络的不同分辨率块中使用不同的层结构。为了降低搜索的计算成本，[28, 5, 45]采用了可微分结构搜索框架和基于梯度的优化方法。文献[48, 15, 12]侧重于将现有网络调整到受限的移动平台，提出了更高效的自动网络简化算法。

量化[23, 25, 47, 41, 51, 52, 37]是另一种重要的补充方法，通过降低运算精度来提高网络效率。最后，知识提炼 [4, 17] 提供了另一种补充方法，在大型 "教师 "网络的指导下生成小型精确的 "学生 "网络。

3. 高效 mobile 构件

移动模型建立在越来越高效的构建模块上。MobileNetV1 [19] 引入了深度可分离卷积，作为传统卷积层的有效替代。深度可分离卷积通过将空间过滤从特征生成机制中分离出来，有效地对传统卷积进行了因子化处理。深度可分离卷积由两个独立的层定义：用于空间过滤的轻量级深度卷积和用于特征生成的重量级 1x1 点式卷积。

MobileNetV2 [39]引入了线性瓶颈和倒残差结构，目的是利用问题的低秩特性，建立更高效的层结构。这种结构如图 3 所示，由 1x1 扩展卷积和深度卷积以及 1x1 投影层构成。当且仅当输入和输出具有相同数量的通道时，它们之间才会通过残差连接。这种结构保持了输入和输出的紧凑表示，同时在内部扩展到更高维度的特征空间，以提高非线性每通道变换的表现力。

MnasNet [43] 以 MobileNetV2 结构为基础，在瓶颈结构中引入了基于挤压和激励的轻量级注意力模块。请注意，与 [20] 中提出的基于 ResNet 的模块相比，挤压和激励模块集成的位置不同。如图 4 所示，该模块被置于扩展中的深度滤波器之后，以便将注意力应用到最大的表示上。

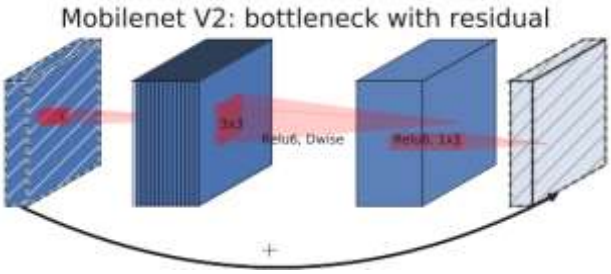

图 3.MobileNetV2 [39] 层（反向残差和线性瓶颈）。每个区块都由狭小的输入和输出（瓶颈）组成，不存在非线性，然后扩展到更高维度的空间并投影到输出。残差连接瓶颈（而不是扩展）。

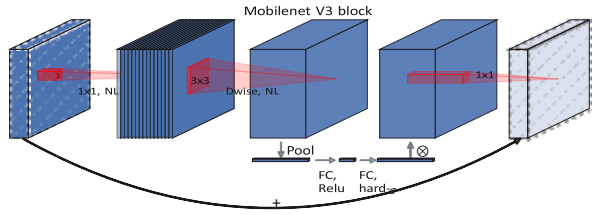

图 4.有残差的瓶颈网络 MobileNetV2 + Squeeze-and-Excite [20]。与 [20] 不同的是，我们在残差层中应用了挤压和激励。我们根据不同的层使用不同的非线性，详见第 5.2 节。

对于 MobileNetV3，我们使用这些层的组合作为构建模块，以建立最有效的模型。我们还利用改进的 swish 非线性[36, 13, 16]对图层进行了升级。挤压和激励以及 "唰 "非线性都使用 sigmoid，计算效率较低，在定点运算中保持精度也很困难，因此我们使用硬 sigmoid [2, 11]来代替它，这在第 5.2 节中讨论过。

4. 搜索最优配置（网络搜索）

网络搜索已被证明是发现和优化网络架构的一个非常强大的工具[53, 43, 5, 48]。对于 MobileNetV3，我们使用平台感知 NAS，通过优化每个网络块来搜索全局网络结构。然后，我们使用 NetAdapt 算法搜索每层的过滤器数量。这些技术是互补的，可以结合使用，从而有效地为给定的硬件平台找到优化模型。

4.1. 面向区块搜索的平台感知 NAS

与 [43] 类似，我们采用了平台感知神经架构方法来寻找全局网络结构。由于我们使用了相同的基于 RNN 的控制器和相同的因式分层搜索空间，我们发现目标延迟在 80ms 左右的大型移动模型的结果与 [43] 相似。因此，我们只需重复使用相同的 MnasNet-A1 [43] 作为初始大型移动模型，然后在其基础上应用 NetAdapt [48] 和其他优化。

然而，我们发现最初的奖励设计并没有针对小型移动模型进行优化。具体来说，它使用多目标奖励 $ACC(m) \times [LAT(m)/TAR]^w$ 来近似帕累托最优解，方法是根据目标延迟 TAR 来平衡每个模型 m 的模型准确度 ACC(m) 和延迟 LAT(m)。我们观察到，对于小模型，准确度随延迟时间的变化更为显著；因此，我们需要一个较小的权重系数 $w = -0.15$（与 [43] 中原来的 $w = -0.07$ 相比），以补偿不同延迟时间下较大的准确度变化。有了这个新的权重系数 w，我们就可以从头开始新的架构搜索，找到初始种子模型，然后应用 NetAdapt 和其他优化方法，得到最终的 MobileNetV3-Small 模型。

4.2. 逐层搜索的 NetAdapt

我们在架构搜索中采用的第二种技术是 NetAdapt [48]。这种方法与平台感知 NAS 相辅相成：它允许以顺序方式对单个层进行微调，而不是试图推断粗略的全局架构。详情请参考原始论文。简而言之，该技术的流程如下：

1. 从平台感知 NAS 找到的种子网络架构开始。
2. 每个步骤：
   a) 生成一组新建议。每个建议都代表对架构的修改，与上一步相比，至少能减少 δ 的时延。
   b) 对于每个建议，我们使用上一步的预训练模型，并填充新的建议架构，酌情截断和随机初始化缺失权重。对每个建议进行 T 步微调，以获得对准确度的粗略估计。
   c) 根据某些指标选出的最佳方案。
3. 迭代上一步，直到达到目标时延。

在 [48] 中，衡量标准是使准确度变化最小。我们修改了这一算法，使延迟变化与准确度

变化之间的比率最小化。也就是说，对于在每个 NetAdapt 步骤中生成的所有建议，我们选择一个最大化的：$\frac{\Delta Acc}{|\Delta latency|}$，其中 Δlatency 满足 2(a) 中的约束条件。直觉告诉我们，由于我们的建议是离散的，因此我们更倾向于使权衡曲线斜率最大化的建议。

这一过程不断重复，直到延迟达到目标，然后我们从头开始重新训练新架构。我们在 MobilenetV2 中使用了与 [48] 相同的建议生成器。具体来说，我们允许以下两种提议：

1. 缩小任何膨胀层的尺寸；
2. 减少瓶颈大小相同的所有区块中的瓶颈--以保持剩余连接。

在实验中，我们使用了 T = 10000，结果发现，虽然它提高了建议的初始微调精度，但并没有改变从头开始训练时的最终精度。我们设置 $\delta = 0.01|L|$，其中 L 是种子模型的延迟。

5. 网络改进

除了网络搜索，我们还为模型引入了几个新的组件，以进一步改进最终模型。我们重新设计了网络开头和结尾的计算密集层。我们还引入了一种新的非线性特性--h-swish，它是最近推出的 swish 非线性特性的改进版本，计算速度更快，对量化也更友好。

5.1. 重新设计高成本层（昂贵层）

通过架构搜索找到模型后，我们发现最后几层和前面几层的成本都比其他层高。我们建议对架构进行一些修改，以减少这些慢层的延迟，同时保持准确性。这些修改超出了当前搜索空间的范围。

第一项修改是重新设计网络最后几层的交互方式，以便更高效地生成最终特征。目前基于 MobileNetV2 倒置瓶颈结构和变体的模型使用 1x1 卷积作为最后一层，以便扩展到更高维度的特征空间。这一层对于获得丰富的预测特征至关重要。但是，这样做的代价是额外的延迟。

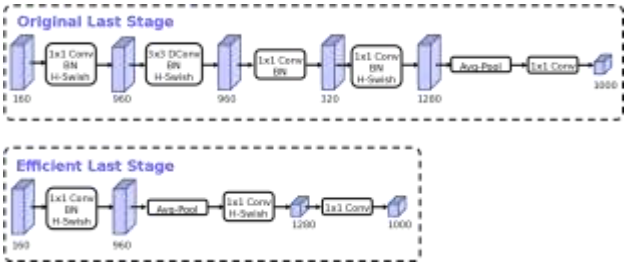

图 5.原始最后阶段与高效最后阶段的比较。这种更高效的最后阶段能够在不降低精度的情况下，在网络末端放弃三个高成本的层。

为了减少延迟并保留高维特征，我们将这一层移到了最后的平均汇集层。现在，这组最终特征是以 1x1 空间分辨率而不是 7x7 空间分辨率计算的。这一设计选择的结果是，在计算和延迟方面，特征的计算几乎是免费的。

一旦特征生成层的成本降低，就不再需要之前的瓶颈投影层来减少计算量。有了这一观察结果，我们就可以去掉前一个瓶颈层中的投影层和过滤层，从而进一步降低计算复杂度。原始和优化后的最后阶段如图 5 所示。高效的最后阶段将延迟时间减少了 7 毫秒，相当于运行时间的 11%，并将操作次数减少了 3000 万 MAdds，而精度几乎没有损失。第 6 节包含详细结果。

另一个高成本的层是初始滤波器组。目前的移动模型倾向于在完整的 3x3 卷积中使用 32 个滤波器来建立用于边缘检测的初始滤波器组。这些滤波器往往互为镜像。我们尝试减少滤波器的数量，并使用不同的非线性来减少冗余。我们最终决定在这一层使用硬喇非线性，因为它的性能与测试过的其他非线性一样好。我们能够将滤波器数量减少到 16 个，同时保持与使用 ReLU 或 Swish 的 32 个滤波器相同的精度。这额外节省了 2 毫秒和 1 千万 MAdds。

5.2. 非线性因素

在 [36, 13, 16] 中，引入了一种名为 Swish 的非线性，当它被用作 ReLU 的直接替代品时，能显著提高神经网络的准确性。该非线性的定义是

$$\text{swish}x = x \cdot \sigma(x)$$

虽然这种非线性提高了精确度，但在嵌入式环境中，这种非线性的代价并不为零，因为在移动设备上计算 sigmoid 函数的成本要高得多。我们通过两种方法来解决这个问题。

一．我们用类似于 [11, 44] 的片断线性硬相似函数：$\frac{\text{ReLU6}(x+3)}{6}$来代替 sigmoid 函数。稍有不同的是，我们使用的是 ReLU6 而不是自定义的削波常数。同样，swish 的硬版本变为

$$\text{h-swish}[x] = x\frac{\text{ReLU6}(x+3)}{6}$$

最近，[2] 也提出了类似的硬 swish 版本。图 6 显示了软版和硬版 sigmoid 和 swish 非线性的比较。我们选择常数的动机是为了简单，并且与原始的平滑版本相匹配。在实验中，我们发现所有这些函数的硬版本在精度上没有明显差异，但从部署角度来看却有多种优势。首先，ReLU6 的优化实现几乎适用于所有软件和硬件框架。其次，在量化模式下，它消除了因近似 sigmoid 的不同实现而造成的潜在数值精度损失。最后，在实际应用中，h-swish 可以作为片断函数来实现，以减少内存访问次数，从而大幅降低延迟成本。

二．应用非线性的成本会随着网络的深入而降低，因为每降低一层分辨率，每层激活内存通常就会减半。顺便提一下，我们发现只有在较深的层中使用 h-swish 才能实现大部分优势。因此，在我们的架构中，我们只在模型的后半部分使用 h-swish。具体布局请参见表 1 和表 2。

即使进行了这些优化，h-swish 仍然会带来一些延迟成本。不过，正如我们在第 6 节中所展示的，在没有优化的情况下，对精度和延迟的净影响是正的，而在使用基于片断函数的优化实现时，则会产生很大的影响。

5.3. 大型挤压激发器

在 [43] 中，挤压-激发瓶颈的大小与卷积瓶颈的大小相对应。相反，我们将它们全部替换为固定值，即扩展层中通道数的 1/4。我们发现，这样做在适度增加参数数量的同时提高了准确度，而且没有明显的延迟代价。

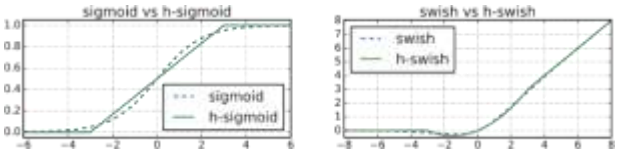

图 6.Sigmoid 和 Swish 非线性及其 "硬 "对应物。

5.4. MobileNetV3 定义

MobileNetV3 有两种型号：MobileNetV3-Large 和 MobileNetV3-Small。这些模型分别针对高资源和低资源用例。这些模型是通过

Input	Operator	exp size	#out	SE	NL	s
224²×3	conv2d	-	16	-	HS	2
112²×16	bneck, 3x3	16	16	-	RE	1
112²×16	bneck, 3x3	64	24	-	RE	2
56²×24	bneck, 3x3	72	24	-	RE	1
56²×24	bneck, 5x5	72	40	√	RE	2
28²×40	bneck, 5x5	120	40	√	RE	1
28²×40	bneck, 5x5	120	40	√	RE	1
28²×40	bneck, 3x3	240	80	-	HS	2
14²×80	bneck, 3x3	200	80	-	HS	1
14²×80	bneck, 3x3	184	80	-	HS	1
14²×80	bneck, 3x3	184	80	-	HS	1
14²×80	bneck, 3x3	480	112	√	HS	1
14²×112	bneck, 3x3	672	112	√	HS	1
14²×112	bneck, 5x5	672	160	√	HS	2
7²×160	bneck, 5x5	960	160	√	HS	1
7²×160	bneck, 5x5	960	160	√	HS	1
7²×160	conv2d, 1x1	-	960	-	HS	1
7²×960	pool, 7x7	-	-	-	-	1
1²×960	conv2d 1x1,NBN	-	1280	-	HS	1
1²×1280	conv2d 1x1,NBN	-	k	-	-	1

表 1.MobileNetV3-Large 的规范。SE 表示该区块中是否存在挤压-激发（Squeeze-And-Excite）。NL 表示所使用的非线性类型。这里，HS 表示 h-swish，RE 表示 ReLU。NBN 表示无批次归一化。

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d, 3x3	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	✓	RE	2
$56^2 \times 16$	bneck, 3x3	72	24	-	RE	2
$28^2 \times 24$	bneck, 3x3	88	24	-	RE	1
$28^2 \times 24$	bneck, 5x5	96	40	✓	HS	2
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	120	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	144	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	288	96	✓	HS	2
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	conv2d, 1x1	-	576	✓	HS	1
$7^2 \times 576$	pool, 7x7	-	-	-	-	1
$1^2 \times 576$	conv2d 1x1, NBN	-	1024	-	HS	1
$1^2 \times 1024$	conv2d 1x1, NBN	-	k	-	-	1

表 2.MobileNetV3-Small 的规格。符号见表 1。

应用平台感知 NAS 和 NetAdapt 进行网络搜索，并结合本节定义的网络改进而创建的。网络的完整规格见表 1 和表 2。

6. 实验

我们展示了实验结果，以证明新的 MobileNetV3 模型的有效性。我们报告了分类、检测和分割的结果。我们还报告了各种消融研究，以阐明各种设计决策的效果。

6.1. 分类

按照标准，我们使用 ImageNet[38] 进行所有分类实验，并将准确率与各种资源使用率进行比较，如延迟和倍增 (MAdds)。

6.1.1 训练设置

我们在 4x4 TPU Pod [24] 上使用标准 tensorflow RMSPropOptimizer（动量为 0.9）同步训练设置训练模型。我们使用的初始学习率为 0.1，批量大小为 4096（每个芯片 128 幅图像），学习率衰减率为 0.01（每 3 个历元）。我们使用 0.8 的辍学率、1e-5 的 l2 权重衰减以及与 Inception [42] 相同的图像预处理。最后，我们使用衰减为 0.9999 的指数移动平均值。我们的所有卷积层都使用批量归一化层，平均衰减为 0.99。

6.1.2 测量配置

为了测量延迟，我们使用标准的 Google Pixel 手机，并通过标准的 TFLite Benchmark 工具运行所有网络。我们在所有测量中都使用单线程大核心。我们没有报告多核推理时间，因为我们发现这种设置在移动应用中并不实用。我们为 tensorflow lite 贡献了原子 h-swish 运算符，现在它已成为最新版本的默认运算符。我们在图 9 中展示了优化后的 h-swish 的影响。

6.2. 结果

如图 1 所示，我们的模型优于 MnasNet [43]、ProxylessNas [5] 和 MobileNetV2 [39]。

Network	Top-1	MAdds	Params	P-1	P-2	P-3
V3-Large 1.0	**75.2**	**219**	5.4M	**51**	**61**	**44**
V3-Large 0.75	73.3	155	4.0M	39	46	40
MnasNet-A1	75.2	315	3.9M	71	86	61
Proxyless[5]	74.6	320	4.0M	72	84	60
V2 1.0	72.0	300	3.4M	64	76	56
V3-Small 1.0	**67.4**	**56**	2.5M	**15.8**	**19.4**	**14.4**
V3-Small 0.75	65.4	44	2.0M	12.8	15.6	11.7
Mnas-small [43]	64.9	65.1	1.9M	20.3	24.2	17.2
V2 0.35	60.8	59.2	1.6M	16.6	19.6	13.9

表 3.Pixel 系列手机的浮点运算性能（P-n 表示 Pixel-n 手机）。所有延迟时间均以毫秒为单位，使用单个大型内核进行测量，批量大小为 1。准确率最高的是 ImageNet。

Network	Top-1	P-1	P-2	P-3
V3-Large 1.0	**73.8**	44	42.5	31.7
V2 1.0	70.9	52	48.3	37.0
V3-Small	**64.9**	15.5	14.9	10.7
V2 0.35	57.2	16.7	15.6	11.9

表 4.量化性能。所有延迟均以毫秒为单位。推理延迟是使用 Pixel 1/2/3 设备上的单个大型内核测得的。

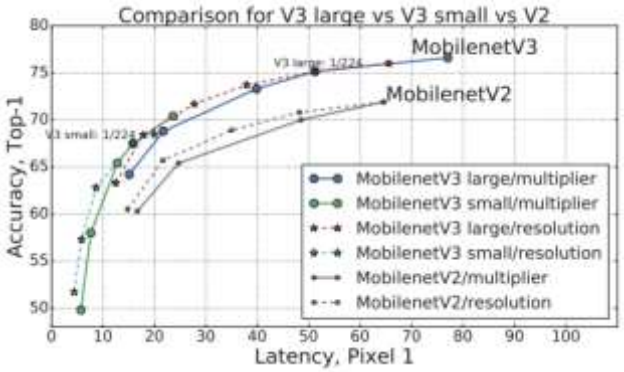

图 7.MobileNetV3 的性能与不同乘数和分辨率的函数关系。在实验中，我们使用了乘数 0.35、0.5、0.75、1.0 和 1.25，分辨率固定为 224，分辨率为 96、128、160、192、224 和 256，深度乘数固定为 1.0。彩色效果最佳。准确率最高的是 ImageNet，延迟时间以毫秒为单位。

	Top-1	P-1	P-1 (no-opt)
V3-Large 1.0	75.2	51.4	57.5
ReLU	74.5 (-.7%)	50.5 (-1%)	50.5
h-swish @16	75.4 (+.2%)	53.5 (+4%)	68.9
h-swish @112	75.0 (-.3%)	51 (-0.5%)	54.4

表 5.非线性因素对 MobileNetV3-Large 的影响。在 h-swish@N 中，N 表示启用了 h-swish 的第一层的信道数。第三列显示的是未优化 h-swish 的运行时间。准确率最高的是 ImageNet，延迟以毫秒为单位。

我们在表 3 中报告了不同 Pixel 手机的浮点性能。我们在表 4 中列出了量化结果。

图 7 显示了 MobileNetV3 性能权衡与乘数和分辨率的关系。请注意，MobileNetV3-Small 的性能比 MobileNetV3Large 高出近 3%。另一方面，分辨率提供了比乘法器更好的权衡。不过，需要注意的是，分辨率通常是由问题决定的（例如，分割和检测问题通常需要更高的分辨率），因此不能总是作为可调参数使用。

6.2.1 消融研究

非线性因素的影响 在表 5 中，我们研究了插入 h-swish 非线性因素的位置选择，以及使用优化实现比使用非线性实现的改进。可以看出，使用优化的 h-swish 实现可以节省 6 毫秒（超过运行时间的 10%）。与传统的 ReLU 相比，优化后的 h-swish 只增加了 1 毫秒。

图 8 显示了基于非线性选择和网络宽度的有效前沿。MobileNetV3 在网络中间使用 h-swish，明显优于 ReLU。值得注意的是，在整个网络中加入 h-swish 比拓宽网络的插值前沿略好。

其他组成部分的影响 图 9 显示了不同组件的引入如何沿着延迟/精度曲线移动。

6.3. 探测

我们使用 MobileNetV3 代替 SSDLite [39] 中的骨干网特征提取器，并在 COCO 数据集 [26] 上与其他骨干网进行比较。

按照 MobileNetV2 [39]，我们将 SSDLite 的第一层附加到输出跨度为 16 的最后一个特征提取层，并将 SSDLite 的第二层附加到输出跨度为 32 的最后一个特征提取层。根据检测文献，我们将这两个特征提取层分别称为 C4 和 C5。对于 MobileNetV3-Large，C4 是第 13 个瓶颈区块的扩展层。对于 MobileNetV3-Small，C4 是第 9 个瓶颈区块的扩展层。对于这两个网络，C5 是紧接池化之前的层。

此外，我们还将 C4 和 C5 之间所有特征层的通道数减少了 2 个，这是因为 MobileNetV3 的最后几层被调整为输出 1000 个类别，而这些类别在转移到 COCO 的 90 个类别时可能是多余的。

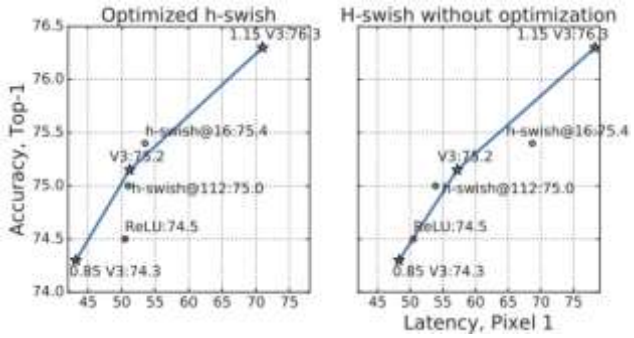

图 8.优化和非优化 h-swish 与 ReLU 对延迟的影响。曲线显示了使用深度乘法器的前沿。需要注意的是，将 h-swish 放在有 80 个或更多通道的所有层（V3），可为优化 h-swish 和非优化 h-swish 提供最佳权衡。前 1 位准确率是在 ImageNet 上的准确率，延迟以毫秒为单位。

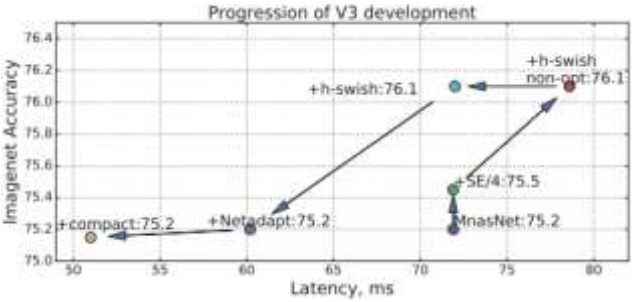

图 9.MobileNetV3 开发过程中各个组件的影响。进度通过向上和向左移动来衡量。

Backbone	mAP	Latency (ms)	Params (M)	MAdds (B)
V1	22.2	228	5.1	1.3
V2	22.1	162	4.3	0.80
MnasNet	23.0	174	4.88	0.84
V3	22.0	137	4.97	0.62
V3†	22.0	119	3.22	0.51
V2 0.35	13.7	66	0.93	0.16
V2 0.5	16.6	79	1.54	0.27
MnasNet 0.35	15.6	68	1.02	0.18
MnasNet 0.5	18.5	85	1.68	0.29
V3-Small	16.0	52	2.49	0.21
V3-Small†	16.1	43	1.77	0.16

表 6.采用不同骨干网的 SSDLite 在 COCO 测试集上的物体检测结果。†:C4 和 C5 之间区块中的通道减少了 2 倍。

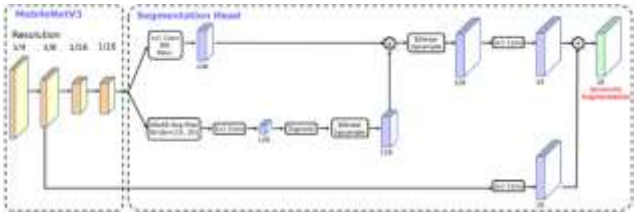

图 10 在 MobileNetV3 的基础上，所提出的分割头 Lite R-ASPP 在混合多种分辨率特征的同时，还能提供快速的语义分割结果。

COCO 测试集的结果见表 6。6.在减少信道的情况下，MobileNetV3-Large 比 MobileNetV2 快 27%，mAP 几乎相同。减少信道后的 MobileNetV3Small 也比 MobileNetV2 和 MnasNet 分别高出 2.4 和 0.5 mAP，但速度却快了 35%。对于这两种 MobileNetV3 模型，减少信道的技巧可减少约 15%的延迟，而 mAP 没有损失，这表明 Imagenet 分类和 COCO 物体检测可能偏好不同的特征提取器形状。

6.4. 语义分割

在本小节中，我们采用 MobileNetV2 [39] 和提议的 MobileNetV3 作为网络骨干来完成移动语义分割任务。此外，我们还比较了两个分割头。第一个被称为 R-ASPP 的分割头是在 [39] 中提出的。R-ASPP 是 Atrous Spatial Pyramid Pooling 模块[7, 8, 9]的简化设计，它只采用了由 1 × 1 卷积和全局平均池化操作组成的两个分支[29, 50]。在这项工作中，我们提出了另一种轻量级分割头，称为 Lite R-ASP（或 LR-ASP），如图 10 所示。Lite R-ASPP 在 R-ASPP 的基础上进行了改进，以类似于 Squeeze-and-Excitation 模块[20]的方式部署了全局平均池化。我们在 MobileNetV3 的最后一个区块中应用了 atrous 卷积[18, 40, 33, 6]，以提取更密集的特征，并进一步从低级特征中添加了一个跳过连接[30]，以捕捉更详细的信息。

我们在城市景观数据集[10]上进行了实验，并使用了度量指标 mIOU [14]，而且只利用了"精细 "注释。我们采用了与 [8, 39] 相同的训练协议。我们的所有模型都是在 ImageNet [38] 上从头开始训练的，没有经过预训练，并使用单一尺度输入进行评估。与物体检测类似，我们观察到，我们可以将网络主干最后一块的通道减少 2 倍，而不会明显降低性能。我们认为这是因为骨干网是为 1000 个类别的 ImageNet 图像分类[38]而设计的，而城市景观只有 19 个类别，这意味着骨干网中存在一些信道冗余。

N	Backbone	RF2	SH	F	mIOU	Params	MAdds	CPU (f)	CPU (h)
1	V2	-	×	256	72.84	2.11M	21.29B	3.90s	1.02s
2	V2	✓	×	256	72.56	1.15M	13.68B	3.03s	793ms
3	V2	✓	✓	256	72.97	1.02M	12.83B	2.98s	786ms
4	V2	✓	✓	128	72.74	0.98M	12.57B	2.89s	766ms
5	V3	-	×	256	72.64	3.60M	18.43B	3.55s	906ms
6	V3	✓	×	256	71.91	1.76M	11.24B	2.60s	668ms
7	V3	✓	✓	256	72.37	1.63M	10.33B	2.55s	659ms
8	V3	✓	✓	128	72.36	1.51M	9.74B	2.47s	657ms
9	V2 0.5	✓	✓	128	68.57	0.28M	4.00B	1.59s	415ms
10	V2 0.35	✓	✓	128	66.83	0.16M	2.54B	1.27s	354ms
11	V3-Small	✓	✓	128	68.38	0.47M	2.90B	1.21s	327ms

表 7.Cityscapes val set 的语义分割结果。RF2：将最后一个区块中的滤波器减少 2 倍。V2 0.5 和 V2 0.35 分别为深度乘数 = 0.5 和 0.35 的 MobileNetV2。SH：分割头，其中 × 采用 R-ASP，而 X 采用建议的 LR-ASP。F：分割头使用的滤波器数量。CPU（f）：全分辨率输入（即 1024 × 2048）时在 Pixel 3 单个大型内核上测量的 CPU 时间（浮点数）。CPU (h)：在半分辨率输入（即 512×1024）条件下测量的 CPU 时间。第 8 行和第 11 行是我们的 MobileNetV3 候选分段。

Backbone	OS	mIOU	MAdds (f)	MAdds (h)	CPU (f)	CPU (h)
V3	16	72.6	9.74B	2.48B	2.47s	657ms
V3	32	72.0	7.74B	1.98B	2.06s	534ms
V3-Small	16	69.4	2.90B	0.74B	1.21s	327ms
V3-Small	32	68.3	2.06B	0.53B	1.03s	275ms
ESPNetv2 [32]	-	66.2	-	2.7B	-	-
CCC2 [34]	-	62.0	-	3.15B	-	-
ESPNetv1 [31]	-	60.3	-	4.5B	-	-

表 8：城市景观测试集的语义分割结果城市景观测试集的语义分割结果 OS：输出跨度，即输入图像空间分辨率与骨干输出分辨率之比。当 OS = 16 时，在主干的最后一个区块中应用无距卷积。当 OS = 32 时，不使用无齿卷积。MAdds (f)：根据全分辨率输入（即 1024 × 2048）测得的乘法累加。MAdds (h)：根据半分辨率输入（即 512 × 1024）测得的乘积。CPU (f)：全分辨率输入（即 1024 × 2048）时在 Pixel 3 单个大型内核上测量的 CPU 时间（浮点运算）。CPU (h)：在半分辨率输入（即 512×1024）条件下测量的 CPU 时间。ESPNet [31, 32] 和 CCC2 [34] 采用半分辨率输入，而我们的模型直接采用全分辨率输入。

我们在表 7 中报告了城市景观验证集的结果。7.如表 7 所示，我们观察到：(1) 将网络主干最后一块的信道减少 2 倍可显著提高速度，同时保持相似的性能（第 1 行与第 2 行相比，第 5 行与第 6 行相比）；(2) 提议的分割头 LR-ASPP 比 R-ASPP [39] 稍快，同时性能有所提高（第 2 行与第 3 行相比，第 6 行与第 7 行相比）；(3) 将分割头中的过滤器从 256 个减少到 128 个可提高速度，但代价是性能稍差（第 3 行与第 4 行相比，第 7 行与第 8 行相比）；(4) 采用相同设置时，MobileNetV3 模型变体可获得相似的性能，但性能稍差（第 3 行与第 4 行相比，第 7 行与第 8 行相

比）。第 3 行与第 4 行相比，第 7 行与第 8 行相比）；(4) 采用相同设置时，MobileNetV3 模型变体的性能与 MobileNetV2 相似，但速度略快于 MobileNetV2（第 1 行与第 5 行相比，第 2 行与第 6 行相比，第 3 行与第 7 行相比，第 4 行与第 8 行相比）；(5) MobileNetV3-Small 的性能与 MobileNetV2-0.5 相似，但速度更快；(6) MobileNetV3-Small 的速度与 MobileNetV2-0.35 相似，但性能明显优于 MobileNetV2-0.35。

表 8 显示了城市景观测试集的结果。表 8 显示了我们的城市景观测试集结果。以 MobileNetV3 为网络骨干的分割模型的 MAdds 分别比 ESPNetv2 [32]、CCC2 [34] 和 ESPNetv1 [32] 快 6.4%、10.6% 和 12.3%。如果在 MobileNetV3 的最后一个区块中不采用无序卷积来提取密集特征图，性能会略微下降 0.6%，但速度却提高到 1.98B（半分辨率输入），分别是 ESPNetv2、CCC2 和 ESPNetv1 的 1.36、1.59 和 2.27 倍。此外，使用 MobileNetV3-Small 作为网络骨干的模型仍以至少 2.1% 的健康优势优于所有这些模型。

7. Conclusions and future work

在本文中，我们介绍了 MobileNetV3 大型和小型模型，展示了移动分类、检测和分割领域的最新技术水平。我们介绍了如何利用多种网络架构搜索算法以及网络设计的进步来提供下一代移动模型。我们还展示了如何以量化友好和高效的方式调整非线性（如 "唰"）并应用挤压和激发，将其作为有效工具引入移动模型领域。我们还引入了一种名为 LR-ASPP 的新型轻量级分割解码器。虽然如何更好地将自动搜索技术与人类直觉相结合仍是一个有待解决的问题，但我们很高兴能首次展示这些积极的成果，并将在今后的工作中继续完善这些方法。

致谢

感谢 Andrey Zhmoginov、Dmitry Kalenichenko、朱孟龙、Jon Shlens、张晓、Benoit Jacob、Alex Stark、Achille Brighton 和 Sergey Ioffe 提供的有益反馈和讨论。

参考文献

[1] 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 Watten-´ berg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 1

[2] R. Avenash and P. Vishawanth. Semantic segmentation of satellite images using a modified cnn with hard-swish activation function. In *VISIGRAPP*, 2019. 2, 4

[3] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *CoRR*, abs/1611.02167, 2016. 2

[4] Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-ˇ Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 535–541, New York, NY, USA, 2006. ACM. 2

[5] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *CoRR*, abs/1812.00332, 2018. 2, 3, 6

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 7

[7] 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

[8] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous

convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017. 7, 8

[9] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 7

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*,2016. 7

[11] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *CoRR*, abs/1511.00363, 2015. 2, 4

[12] Xiaoliang Dai, Peizhao Zhang, Bichen Wu, Hongxu Yin, Fei Sun, Yanghan Wang, Marat Dukhan, Yunqing Hu, Yiming Wu, Yangqing Jia, Peter Vajda, Matt Uyttendaele, and Niraj K. Jha. Chamnet: Towards efficient network design through platform-aware model adaptation. *CoRR*, abs/1812.08934, 2018. 2

[13] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoidweighted linear units for neural network function approximation in reinforcement learning. *CoRR*, abs/1702.03118, 2017. 2, 4

[14] 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. 7

[15] Yihui He and Song Han. AMC: automated deep compression and acceleration with reinforcement learning. In *ECCV*, 2018. 2

[16] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016. 2, 4

[17] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. 2

[18] Matthias Holschneider, Richard Kronland-Martinet, 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. Springer Berlin Heidelberg, 1989. 7

[19] 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

[20] J. Hu, L. Shen, and G. Sun. Squeeze-and-Excitation Networks. *ArXiv e-prints*, Sept. 2017. 2, 3, 7

[21] Gao Huang, Shichen Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Condensenet: An efficient densenet using learned group convolutions. *CoRR*, abs/1711.09224, 2017. 2

[22] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016. 2

[23] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[24] Norman P. Jouppi, Cliff Young, Nishant Patil, David A. Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierreluc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, Richard C. Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit. *CoRR*, abs/1704.04760, 2017. 5

[25] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *CoRR*, abs/1806.08342, 2018. 2

[26] 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. 7

[27] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan

Huang, and Kevin Murphy. Progressive neural architecture search. *CoRR*, abs/1712.00559, 2017. 2

[28] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. *CoRR*, abs/1806.09055,2018. 2

[29] Wei Liu, Andrew Rabinovich, and Alexander C. Berg. Parsenet: Looking wider to see better. *CoRR*, abs/1506.04579, 2015. 7

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 7

[31] Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda G. Shapiro, and Hannaneh Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part X*, pages 561–580, 2018. 8

[32] Sachin Mehta, Mohammad Rastegari, Linda G. Shapiro, and Hannaneh Hajishirzi. Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. *CoRR*, abs/1811.11431, 2018. 8

[33] George Papandreou, Iasonas Kokkinos, and Pierre-Andre Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *CVPR*, 2015. 7

[34] Hyojin Park, Youngjoon Yoo, Geonseok Seo, Dongyoon Han, Sangdoo Yun, and Nojun Kwak. Concentratedcomprehensive convolutions for lightweight semantic segmentation. *CoRR*, abs/1812.04920, 2018. 8

[35] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *CoRR*, abs/1802.03268, 2018. 2

[36] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017. 2, 4

[37] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. *CoRR*, abs/1603.05279, 2016. 2

[38] 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, Dec. 2015. 5, 8

[39] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018. 2, 3, 6, 7, 8

[40] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Math-¨ ieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv:1312.6229*, 2013. 7

[41] Daniel Soudry, Itay Hubara, and Ron Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *NIPS*, pages 963–971, 2014. 2

[42] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. 5

[43] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. *CoRR*, abs/1807.11626, 2018. 2, 3, 5, 6

[44] SPSE the Society for Imaging Science, Technology, Society of Photo-optical Instrumentation Engineers, and Technical Association of the Graphic Arts. *Curves and Surfaces in Computer Vision and Graphics*. Number v. 1610 in Proceedings of SPIE–the International Society for Optical Engineering. SPIE, 1992. 4

[45] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. *CoRR*, abs/1812.03443, 2018. 2

[46] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter H. Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. *CoRR*, abs/1711.08141,2

[47] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. *CoRR*, abs/1512.06473, 2015. 2

[48] Tien-Ju Yang, Andrew G. Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *ECCV*, 2018. 2, 3

[49] 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

[50] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 7

[51] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *CoRR*, abs/1702.03044,2

[52] Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, abs/1606.06160, 2016. 2

[53] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016. 2, 3

[54] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017. 2

A. 不同分辨率和乘法器的性能表

我们在表 9 中给出了详细的表格，其中包括乘法加法、精确度、参数数量和延迟。

Network	Accuracy	Madds (M)	Params (M)	P-1 (ms)
large 224/1.25	76.6	356	7.5	77.0
large 224/1.0	75.2	217	5.4	51.2
large 224/0.75	73.3	155	4.0	39.8
large 224/0.5	68.8	69	2.6	21.7
large 224/0.35	64.2	40	2.2	15.1
large 256/1.0	76.0	282	5.4	65.6
large 192/1.0	73.7	160	5.4	38.0
large 160/1.0	71.7	112	5.4	27.8
large 128/1.0	68.4	73	5.4	17.8
large 96/1.0	63.3	43	5.4	12.5
small 224/1.25	70.4	91	3.6	23.6
small 224/1.0	67.5	57	2.5	15.8
small 224/0.75	65.4	44	2.0	12.8
small 224/0.5	58.0	21	1.6	7.7
small 224/0.35	49.8	12	1.4	5.7
small 256/1.0	68.5	74	2.5	20.0
small 160/1.0	62.8	30	2.5	8.6
small 128/1.0	57.3	20	2.5	5.8
small 96/1.0	51.7	12	2.5	4.4

表 9.大型和小型 V3 型号的浮点性能。P-1 相当于 Pixel 1 上的大型单核性能。