# 面向硬件感知高效块设计的计算强度平衡卷积

**Shinkook Choi** [1] **Junkyeong Choi** [1]

arXiv:2304.04249v1 [cs.LG] 8 Apr 2023

**摘要**

随着深度学习的发展，边缘设备和轻量级神经网络变得越来越重要。为了降低人工智能加速器的延迟，不仅需要减少浮点运算次数，还需要提升硬件性能。针对空间尺寸较小的卷积中，总体算术强度受权重算术强度较低所限制的问题，我们提出了算术强度平衡卷积（ABConv）。ABConv 提高了总体算术强度的上限，并显著降低了延迟，同时不会牺牲准确率。我们在不同配置下测试了 ABConv 在 Arm Ethos-U65 神经网络处理器上的延迟和硬件性能，并将其用于替换 CIFAR100 图像分类任务中的部分 MobileNetV1 和 ResNet50。

## 1 引言

过去几年，人工智能（AI）的应用显著增长，推动了 AI 技术的巨大进步。然而，随着 AI 能力不断提升，算法的复杂性也随之增加。由于大多数高性能 AI 模型过于庞大，边缘设备难以承载，因此，设计 AI 模型优化技术以应对严格的内存和功耗约束，变得比以往任何时候都更加重要。

许多研究表明，轻量级深度神经网络（DNN）旨在减少浮点运算次数（FLOPs）和参数数量，以便在边缘设备上运行（Howard 等人，2017年；Sandler 等人，2018年；Tan & Le，2019年；2021年）。然而，诸如浮点运算次数（FLOPs）和参数等计算复杂度的间接指标，并不能直接反映硬件在边缘设备上进行计算的效率（Akin 等人，2022年；Ma 等人，2018年）。因此，在设计优化的神经网络时，同时考虑神经网络的特性和目标硬件非常重要

要为硬件设计最高效的深度学习模型，算术强度（Harris，2005年）和屋顶线分析（Williams 等人，2009）可用于阐述数据复用和硬件性能。算术强度是算术运算次数与内存占用的比值。更高的算术强度意味着更高程度的数据复用，从而实现高效计算。

随着深度神经网络工作负载成为计算效率方面的主要关注点（Gouppi 等人，2017年；Park 等人，2018年；Yang 等人，2017年），采用基于算术强度的屋顶线分析来可视化深度神经网络工作负载的效率。此外，Jha 等人（Jha 和 Mittal，2020年）进一步探讨了算术强度和数据复用的概念，并根据深度神经网络所用数据的类型，将数据复用分为权重复用和激活复用。

设计网络时我们考虑的另一项硬件特性是卷积延迟的阶梯模式（Tang 等人，2021）。尽管卷积的算术复杂度呈线性增长，但卷积延迟并不会线性变化，而是以一定的步长增加；该步长因硬件而异，呈现出阶梯状的延迟模式。

在本研究中，我们强调了 Arm Ethos-U65 NPU 上出现的一个问题：由于卷积层的输入空间尺寸，当算术强度受权重数据限制时，就会产生该问题。为解决这一问题，我们提出了一种新的层，使权重数据和激活数据的算术强度达到平衡，从而提高总体算术强度。我们证明，提高权重算术强度能够改善硬件性能，并评估了其效

*表 1.* 标准卷积、分组卷积以及所提出的 ABConv 和 ABConv-exp 的数据复用特性。

| | MACs ($M_c$) | 权重大小 ($W$) | 激活值大小 ($A$) | 算术强度 | |
|---|---|---|---|---|---|
| | | | | 权重 ($M_c/W$) | 激活值 ($M_c/A$) |
| 标准卷积 | $S_o^2k^2C_{in}C_{out}$ | $k^2C_{in}C_{out}$ | $S_o^2(C_{in}+C_{out})$ | $S_o^2$ | $\frac{k^2C_{in}C_{out}}{C_{in}+C_{out}}$ |
| 分组卷积 | $\frac{S_o^2k^2C_{in}C_{out}}{g}$ | $\frac{k^2C_{in}C_{out}}{g}$ | $S_o^2(C_{in}+C_{out})$ | $S_o^2$ | $\frac{k^2C_{in}C_{out}}{g(C_{in}+C_{out})}$ |
| ABConv | $\frac{S_o^2k^2C_{in}C_{out}}{g}$ | $\frac{k^2C_{in}C_{out}}{g^2}$ | $S_o^2(C_{in}+C_{out})$ | $gS_o^2$ | $\frac{k^2C_{in}C_{out}}{g(C_{in}+C_{out})}$ |
| ABConv-exp | $S_o^2k^2C_{in}C_{out}$ | $\frac{k^2C_{in}C_{out}}{g}$ | $S_o^2\ (C_{in}+2gC_{mid}+C_{out})$ | $gS_o^2$ | $\frac{k^2C_{in}C_{out}}{C_{in}+2gC_{mid}+C_{out}}$ |

[1]NotaInc.，韩国首尔。通讯作者：ShinkookChoi < shinkook.choi@nota.ai>。

《 $3^{rd}$ 端侧智能研讨会，MLSys 会议，美国佛罗里达州迈阿密，2023 年。版权所有 © 2023，作者所有。

我们的方法在降低推理速度方面的有效性。我们的方法还包括一种面向 Ethos-U65 神经网络处理器的硬件自适应、阶梯感知微调方法。我们的实验表明，在图像分类任务中，采用所提出层的模型比 Mo-bileNetV1 和 ResNet50具有更高的准确率和更低的延迟。据我们所知，这是首项表明卷积运算的硬件性能可能缺乏某种算术强度的研究。

## 2 动机

在深度神经网络中，算术强度是指每字节权重（滤波器系数）和激活值（输入特征图和输出特征图）所执行的乘加运算次数（MACs）。当乘加运算次数、权重大小、激活值大小分别表示为 $M_c$、 $W$ 和 $A$时，总体算术强度（$\frac{M_c}{W+A}$）在权重算术强度（$\frac{M_c}{W}$）和激活算术强度（$\frac{M_c}{A}$）上均存在上限，如公式1所示。因此，如果权重算术强度和激活算术强度中的任意一个变小，整体算术强度的上限就会降低，从而整体算术强度也会降低。

$$\frac{M_c}{W+A} < \frac{M_c}{W}, \frac{M_c}{W+A} < \frac{M_c}{A} \quad \text{if } M_c, W, A > 0. \quad (1)$$

表 1 展示了卷积的数据复用特性。权重算术强度由空间尺寸决定($S_o$)，而激活算术强度由卷积核尺寸（$k$）、输入通道数($C_{in}$)和输出通道数($C_{out}$)计算得出。

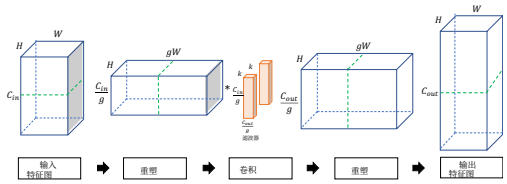
许多基于卷积层的神经网络都经过设计，随着网络深度增加，空间尺寸逐渐减小，通道尺寸逐渐增大。这种设计会导致总体算术强度降低，而网络后期的算术强度主要受权重算术强度限制。因此，有必要设计一种新的层，在扩大空间尺寸的同时提高权重的数据复用率，从而充分利用硬件。

## 3 所提出的方法

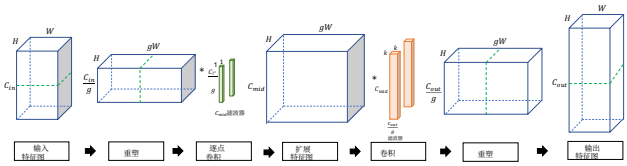
### 3.1 算术强度平衡卷积

我们提出算术强度平衡卷积（ABConv），以最大限度提高硬件利用率。当整体算术强度受限于权重算术强度时，ABConv 会扩大空间尺寸并复用更多权重数据，从而提高整体算术强度。如图所示。

1，ABConv 由三个层组成，依次为重塑层、卷积层和重塑层




(a) ABConv




(b) ABConv-exp

图 *1*。所提出的 ABConv 和 ABConv-exp 的示意图，其输出的特征图数量与标准卷积层相同。

。第一个重塑层沿通道方向按组（$g$）切分特征图，并沿空间尺寸方向将其拼接。在卷积层中，按组划分的卷积核与空间尺寸扩大了组倍的输入特征图进行卷积。最后一个重塑层将输出特征图的尺寸恢复至原始值。由于 ABConv 使用单个卷积核对沿空间方向扩展的特征图进行卷积，因此权重算术强度提高了 $g$ 倍。

表格1 表明，ABConv 的网络容量可能会降低，因为 ABConv 的乘加运算次数和权重大小会因分组数的增加而减少。为了提高 ABConv 的网络复杂度，我们提出了 ABConv的扩展版本（ABConv-exp），在 ABConv 的第一个重塑层之后添加了一个逐点卷积层。ABConv-exp 将逐点卷积的滤波器数量（$C_{mid}$）设置为 $\frac{k^2 C_{in} C_{out}}{C_{in}+k^2 C_{out}}$，使其与标准卷积的乘加运算次数具有相同的值。

### 3.2 组选择算法

如表 1 中 ABConv 和 ABConv-exp 所示，当 $g$ 过度增大以提高权重算术强度时，激活算术强度反而会降低。因此，为了提高整体算术强度，平衡权重数据复用与激活数据复用至关重要。我们提出了一种分组选择算法，用于平衡权重和激活算术强度，从而提升硬件性能并降低延迟（算法 1）。首先，在卷积阶梯的步长（$q_{in}$， $q_{out}$）下，将输入和输出通道分别除以该步长，计算其商（$r_{in}$， $r_{out}$）和余数（$t_{in}$， $t_{out}$）。其次，如果余数均为零，则使整体最大化的最优 $g$ 值（$g_{opt}$）

**算法 1**分组选择算法

**过程** GROUP-SELECT($S_o, k, C_{in}, C_{out}, t_{in}, t_{out}, is_{\text{exp}}$)

$q_{in} = floor(C_{in}/t_{in})$
$q_{out} = floor(C_{out}/t_{out})$
$r_{in} = C_{in}/t_{in} - q_{in}$
$r_{out} = C_{out}/t_{out} - q_{out}$
**如果** $(r_{in} = 0)\&(r_{out} = 0)$ **则**
  **如果** $is_{\text{exp}} = False$ **则**
    $g_{opt} = \sqrt{\frac{C_{in}+C_{out}}{S_o^2}}$
  **else**
    $g_{opt} = \frac{-C_{in}-C_{out}+\sqrt{\frac{(C_{in}+C_{out})^2+8(C_{in}+k^2 C_{out}\ C_{mid}^2)}{S_o^2}}}{4C_{mid}}$
  **结束如果**
  $G$ = 公因数($q_{in}$, $q_{out}$)
  $g = G[argmin(|G - g_{opt}|)]$
  **如果** $g = 1$ **则**
    $\text{sw}_{rep} = False$
  **else**
    $\text{sw}_{rep} = True$
  **结束如果**
**else**
  $g = 1$, $\text{sw}_{rep} = False$
**结束如果**
**返回** $g$, $\text{sw}^{rep}$
**结束过程**

算术强度通过求解寻找平衡点的问题得到。使用 ABConv 和 ABConv-exp 时，$is_{\text{exp}}$ 分别为 $False$ 和 $True$。我们将两个商的公约数列为分组候选项。接下来，从分组候选项中选出最接近 $g_{opt}$ 的值。最后，该算法返回 $g$ 和 $\text{sw}^{rep}$，后者是一个开关，用于决定是否将卷积更改为 ABConv 或 ABConv-exp。

## 4 实验设置

### 4.1 目标设备

实验重点关注边缘设备，并使用 Arm Ethos-U65 Micro-NPU（Ethos-U65）作为神经网络处理器设备。我们使用 Arm固定虚拟平台（FVP）（Arm固定虚拟平台）对集成 Arm 神经网络处理器的系统进行了完整仿真。在实验中，Ethos-U65 设备配置了 256 个乘累加处理单元，并采用专用 SRAM 存储器模式。Ethos-U65 设备的模型被量化为 8 位整数，并通过 Vela 编译器（Arm Vela）转换。

### 4.2 神经网络

由于高效模型在边缘设备上的瓶颈层通常由逐点卷积（$k = 1$）组成，因此我们在本研究中首先通过替换这些层来验证所提出的方法。我们选择MobileNetV1 和 ResNet50 作为基线模型，并将组选择算法应用于基线模型的每个逐点卷积层。如果该算法返回$\text{sw}^{rep}$ 作为 $True$，且 $g$ 大于 1，我们就将该逐点卷积层替换为 ABConv

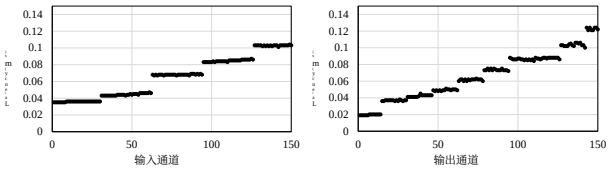



图 *2.* 当 $S_o$ 为 32 时的阶梯式延迟图。

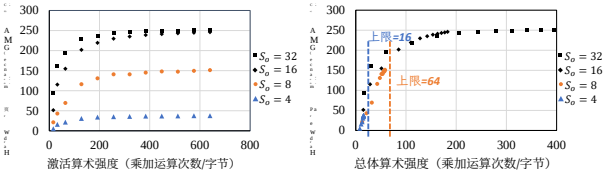



图*3*。逐点卷积的屋顶线分析。纵轴表示每秒十亿次乘加运算。

或 ABConv-exp。实验评估在 CIFAR100 数据集上进行，以聚焦于较小的空间尺寸（$S_o = 32$）。

我们使用预训练的基线模型，并以 128 的批次大小训练变体模型 200 个周期。我们使用标准的随机梯度下降优化器，动量设为 0.9。初始学习率设为 0.001；当验证集的准确率在 0.0001 的范围内不再提升时，就降低学习率。我们测量批次大小为 1 时的推理延迟。

## 5 结果

### 5.1 硬件性能

为了正确设置我们的分组选择算法，该方法应了解设备在每个卷积参数上的步长。如图所示2，Ethos-U65 上的卷积运算在输入通道（$t_{in}$）中的步长为 32，在输出通道（$t_{out}$）中的步长为 16。

在评估 ABConv 之前，我们研究了逐点卷积的低效问题，尤其是在空间尺寸较小时。图 3 展示了 Ethos-U65 的 Roofline 图，分别对应激活数据的算术强度和总体算术强度。每个点代表一个逐点卷积，其输入/输出通道数和空间尺寸各不相同。我们观察到，当空间尺寸较小时（例如 4 和 8），卷积的硬件性能会受到某个上限的限制。这是因为总体算术强度受权重数据算术强度的限制，即使激活数据具有足够高的算术强度也是如此。空间尺寸较小的图无法超过某些上限，这些上限等于空间尺寸的平方。因此，无论卷积对激活数据的复用效果如何，都无法突破总体算术强度的上限

表2。当 $S_o^2 \times C_{in} \times C_{out}$ 的配置为 $4^2 \times 1024 \times 1024$ 时，逐点卷积与所提出方法的对比

| | 组策略 | $g$ | MACs | 参数 | 算术强度 | | | 延迟 ($\mu s$) |
|---|---|---|---|---|---|---|---|---|
| | | | | | 权重 | 激活值 | 整体 | |
| 逐点卷积 | | | 16,777,216 | 1,048,576 | 16 | 512 | 15.5 | 452.1 |
| ABConv | | 32 | 524,288 | 1,024 | 512 | 16 | 15.5 | 7.1 |
| | ✓ | 4 | 4,194,304 | 65,536 | 64 | 128 | 42.7 | 36.1 |
| ABConv-exp | | 32 | 16,777,216 | 32,768 | 512 | 30 | 28.4 | 110.1 |
| | ✓ | 8 | 16,777,216 | 131,072 | 128 | 102 | 56.9 | 86.1 |

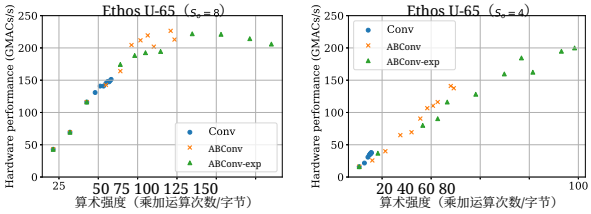



图4。 在 $S_o$ 为4和8时，Ethos-U65 上的屋顶线分析。Y 轴表示每秒十亿次乘加运算。

，导致空间尺寸较小时硬件性能低下。

### 5.2 ABConv

表2比较了卷积层、分组卷积、ABConv 和 ABConv-exp 在使用和不使用分组选择算法时的表现。通过提高权重算术强度，ABConv 和 ABConv-exp 的算术强度高于卷积层和分组卷积，延迟则更低。不使用分组选择算法时，权重数据的算术强度会随着最大值 $g$ 增至 32 而提高；由于最大值 $g$，整体算术强度会受限于降低后的激活算术强度，从而下降。在不使用分组选择算法的 ABConv 中，延迟最低，但这是因为较大的 $g$值导致乘加运算次数减少。

x 图4展示了当输入通道数和输出通道数从128开始以128为步长增加至1280时，ABConv和ABConv-exp在Ethos-U65上的屋顶线分析结果。结果还表明，ABConv和ABConv-exp的算术强度高于卷积层。因此，所提出的两种方法的硬件性能均优于卷积层。由于输入尺寸越小，总体算术强度的下降幅度越大，所提出的卷积层在硬件性能方面的提升也更为显著。

图5 展示了在配置与图 4相同的情况下，Ethos-U65 上的延迟与乘加运算次数之间的关系。由于所提出的 ABConv 和 ABConv-exp 在硬件利用率方面性能出色，因此延迟较低。与卷积层相比，AB-

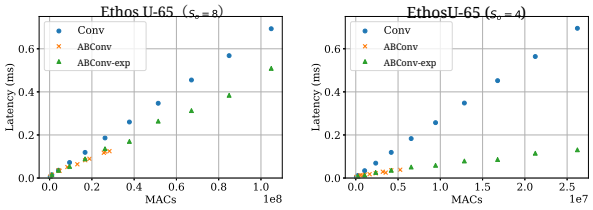



图5。 在 $S_o$ 为4和8时，Ethos-U65上的延迟与乘加运算次数对比。

表3。各模型在CIFAR100上的性能比较

| 基线 | Type | 准确率 (%) | MACs (M) | 参数 (M) | 延迟 (ms) |
|---|---|---|---|---|---|
| MobileNetV1 | 原文 | 66.73 | 46.5 | 3.3 | 1.43 |
| | ABConv | 67.18 | 39.9 | 2.4 | 0.99 |
| | ABConv-exp | 66.95 | 46.8 | 2.4 | 1.02 |
| ResNet50 | 原始 | 78.94 | 1298.0 | 23.8 | 12.35 |
| | ABConv | 78.73 | 1239.3 | 21.0 | 11.56 |
| | ABConv-exp | 78.69 | 1304.8 | 21.5 | 11.91 |

Conv 的乘加运算次数和延迟都较低，而 ABConv-exp 的乘加运算次数相同，但延迟更低。

为在基线网络中引入高效层，我们用 ABConv 替换现有的逐点卷积层，以提升硬件性能。然而，由于不同通道组之间缺乏通信，将所有逐点卷积替换为 ABConv 会降低准确率。因此，有必要交替使用逐点卷积和 ABConv，以维持准确率。

如表 3所示，我们通过将每个逐点卷积替换为 ABConv 和 ABConv-exp，对比了两个基线模型的性能。基于 MobileNetV1 的模型采用所提出的层后，准确率提高且延迟降低。我们观察到，ResNet50 中经过修改的模型也降低了延迟，同时准确率仅有轻微下降。

## 6 结论

本文报告，随着权重复用减少，总体算术强度必然受限于权重算术强度。我们提出 ABConv 作为一种新型层，用于平衡权重数据和激活数据的算术强度，从而最大化整体算术强度。为验证 ABConv，我们将 MobileNetV1 和 ResNet50 中的逐点卷积替换为 ABConv。我们发现，在 CIFAR100 上使用 ABConv 的模型能够降低 Arm Ethos-U65 神经网络处理器上的延迟，同时准确率没有下降。未来研究可以对标准卷积（$k > 1$）和所提出的方法进行对比实验，以进一步验证该方法。

## 参考文献

Akin、B.、Gupta、S.、Long、Y.、Spiridonov、A.、Wang、Z.、White、M.、Xu、H.、Zhou、P. 和 Zhou 、 Y. 在边缘张量处理单元上搜索用于设备端机器学习的高效神经网络架构。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第2667–2676页，2022年。

Arm Ethos-U65。Arm Ethos-U65微型神经网络处理器描述。https://www.arm.com/products/silicon-ip-cpu/ethos/ethos-u65。

Arm固定虚拟平台。Arm固定虚拟平台说明。https://developer.arm.com/Tools%20and%20Software/Fixed%20Virtual%20Platforms。Arm Vela。Arm Vela 编译器说明。https://developer.arm.com/documentation/101888/0500/NPU-software-overview/NPU-software-tooling/The-Vela-compiler。

Harris，M. 将计算概念映射到图形处理器。发表于*ACM SIGGRAPH 2005*课程，第50–页。2005年。

Howard，A. G.、Zhu，M.、Chen，B.、Kalenichenko，D.、Wang，W.、Weyand，T.、Andreetto，M. 和 Adam，H. MobileNets：面向移动视觉应用的高效卷积神经网络。*arXiv* 预印本 *arXiv:1704.04861*，2017年。Jha，N. K. 和 Mittal，S. 通过考虑数据类型对深度神经网络中的数据复用进行建模。*IEEE*计算机汇刊，70(9):1526–1538，2020年。Jouppi，N. P.、Young，C.、Patil，N.、Patterson，D.、Agrawal，G.、Bajwa，R.、Bates，S.、Bhatia，S.、Boden，N.、Borchers，A. 等。张量处理单元在数据中心的性能分析。发表于第*44*届年度国际计算机体系结构研讨会论文集，第1–12页，2017年。Ma，N.、Zhang，X.、Zheng，H.-T. 和 Sun，J. ShuffleNet V2：高效卷积神经网络架构设计的实用指南。发表于欧洲计算机视觉会议（*ECCV*）论文集，第116–131页，2018年。Park，J.、Naumov，M.、Basu，P.、Deng，S.、Kalaiah，A.、Khudia，D.、Law，J.、Malani，P.、Malevich，A.、Nadathur，S. 等。Facebook 数据中心的深度学习推理：特征分析、性能优化和硬件影响。*arXiv* 预印本 *arXiv:1811.09886*，2018年。

Sandler，M.、Howard，A.、Zhu，M.、Zhmoginov，A. 和 Chen，L.-C. MobileNetV2：倒置残差和线性瓶颈。发表于*IEEE*会议论文集，主题为计算机视觉与模式识别，第 4510–4520 页，2018年。

Tan, M. 和 Le, Q. Efficientnet：重新思考卷积神经网络的模型缩放。在国际机器学习会议上，第 6105–6114 页。PMLR，2019 年。

Tan, M. 和 Le, Q. Efficientnetv2：更小的模型与更快的训练。载于国际机器学习会议，第 10096–10106 页。PMLR，2021 年。

Tang, X.、Han, S.、Zhang, L. L.、Cao,T.和 Liu, Y. 为连接神经网络设计与现实世界性能：神经网络的行为研究。机器学习与系统论文集，3:21–37，2021年。

Williams, S.、Waterman, A.和Patterson, D.Roofline：一种针对多核架构的富有洞察力的可视化性能模型。*ACM*通讯，52(4):65–76，2009年。

Yang, T.-J.、Chen,Y.-H.、Emer, J. 和 Sze, V. 一种估算深度神经网络能耗的方法。载于*2017*年 第*51*届阿西洛马会议：信号、系统与计算机，第1916–1920页。IEEE，2017年。

# 附录

## A 消融研究

表 4 展示了在 CIFAR100上，混合使用逐点卷积和所提出方法的不同模型之间模型准确率与延迟的权衡。 $P$、 $A$和 $E$ 分别表示交替使用逐点卷积、ABConv 和 ABConv-exp的模型。正如我们所预期的，仅使用 ABConv 的模型具有最快的延迟，但由于不同通道组之间缺乏通信，其准确率有所下降。我们观察到，混合使用 $P$、 $A$和 $E$ 的模型能够在通道之间建立连接并提高准确率。

## B NVIDIA JETSONNANO

为了在不同设备上验证所提出的 ABConv，我们选择 NVIDIA Jetson Nano 作为边缘GPU设备，并进行了与 Ethos U-65 NPU相同的实验。如图 6所示，JetsonNano 上的卷积运算在输入通道上的步长为 8，在输出通道上的步长为 32。图 7 和 8 表明，在 Jetson Nano 上，具有相同乘加运算次数时，ABConv和 ABConv-exp 也具有更高的算术强度，并能降低延迟。表 5 显示，将MobileNetV1 和 ResNet50 中的每个逐点卷积替换为 ABConv 和 ABConv-exp 后，在准确率相近的情况下可降低延迟。

表 *4*。Ethos-U65 上 MobileNetV1 的架构研究

| 模型 | 乘加运算次数（M） | 延迟 (毫秒) | 准确率 (%) |
|---|---|---|---|
| $P-P-P-P-P-P-P$ | 46.5 | 1.43 | 66.73 |
| $A-A-A-A-A-A-A$ | 23.6 | 0.42 | 64.72 |
| $A-P-A-P-A-P$ | 36.8 | 0.95 | 66.69 |
| $A-P-P-A-P-P$ | 39.9 | 0.99 | 67.18 |
| $E-E-E-E-E-E$ | 46.8 | 0.60 | 65.26 |
| $E-P-E-P-E-P$ | 46.8 | 1.02 | 66.95 |
| $E-P-P-E-P-P$ | 46.8 | 1.07 | 66.92 |

表 *5*。Jetson Nano 上各模型的 CIFAR100 性能比较

| 基线 | Type | 准确率 (%) | MACs (M) | 参数 (M) | 延迟 (ms) |
|---|---|---|---|---|---|
| MobileNetV1 | 原版 | 66.73 | 46.5 | 3.3 | 2.94 |
| | ABConv | 67.18 | 39.9 | 2.4 | 2.79 |
| | ABConv-exp | 66.95 | 46.8 | 2.4 | 2.94 |
| ResNet50 | 原文 | 78.94 | 1298.0 | 23.8 | 23.09 |
| | ABConv | 78.73 | 1239.3 | 21.0 | 20.97 |
| | ABConv-exp | 78.69 | 1304.8 | 21.5 | 21.69 |

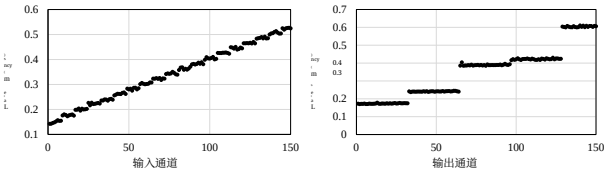



图 *6*。当 $S_o$ 为 32 时，NVIDIA Jetson Nano 上的阶梯式延迟图。

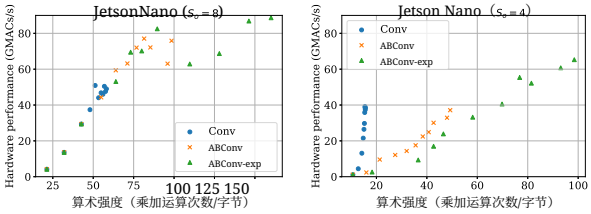



图 *7*。当 $S_o$ 为 4 和 8 时，Jetson Nano 上的 Roofline 分析。纵轴表示每秒十亿次乘加运算。

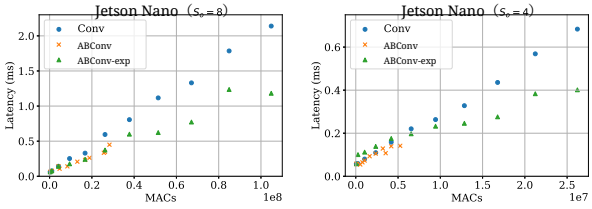



图 *8*。 在 $S_o$ 为4 和 8 时 Jetson Nano 上的延迟与乘加运算次数。