本文已更新并迁移至 https://arxiv.org/abs/2305.10947

纯 16 位浮点神经网络的隐藏力量

Juyoung Yun [1] **Byungkon Kang** [2] **Francois Rameau** [2] **Zhoulai Fu** [2]

arXiv:2301.12809v2 [cs.LG] 3 May 2024

摘要

长期以来，人们一直认为，将神经网络的精度从普遍采用的 32 位精度降低会损害性能，尽管这样做能节省空间和时间。许多研究提出了各种实现半精度神经网络的技术，但没有研究纯16 位设置。本文探讨了纯 16 位神经网络在分类任务中的表现意外优于 32 位网络这一现象。我们给出了大量实验结果，表明各种 16 位神经网络的性能优于 32 位模型。此外，本文还提供了关于 16 位模型效率的理论分析，并辅以实证证据加以支持。最后，我们讨论了低精度训练确实会造成不利影响的情况。

1. 引言

如今，从自动驾驶、医疗保健和金融，到通用人工智能与工程领域，神经网络技术无处不在，对许多人而言，这已成为不争的事实。使用真实世界数据训练神经网络可能需要强大的计算能力，这促使半导体行业致力于研发尖端芯片和图形处理器解决方案，以支持低精度浮点运算。

与传统的基于中央处理器的单精度和双精度计算相比，低精度计算在速度、内存使用和能耗方面都带来了显著的性能提升。如今，许多图形处理器都支持低精度浮点运算。尤其是，广泛可用的 NVIDIA 图形处理器支持 IEEE 标准半精度，即 16 位浮点格式。

[1]美国纽约州石溪大学计算机科学系 [2]大韩民国仁川市韩国纽约州立大学计算机科学系。通讯作者：Byungkon Kang <byungkon.kang@sunykorea.ac.kr>、Zhoulai Fu <zhoulai.fu@sunykorea.ac.kr>。

然而，众所周知，仅使用低精度也会造成精度损失。在 IEEE 16 位格式中，除次正规数外，正数的范围为 5.96E-8 至 6.5E04，因此，将单精度或双精度数据直接舍入或截断为半精度，可能会导致溢出、下溢或产生非规格化数，这些都会影响数值精度。

考虑 Sigmoid 函数，$sigmoid(x) = 1/(1+e^{-x})$。当 x 较小时会发生上溢，当 x 较大时会发生下溢；两种情况都发生在 e^{-x}中，随后误差会传递到函数的输出。作为一个快速实验，可以将向量 x 设为除 $\pm\infty$ 和非数之外的所有 16 位浮点数（总计 63487 个），并将 y 设为所有向上转换后的 32 位浮点数（y= np.float32(x)），然后逐元素比较$sigmoid(x)$ 和 $sigmoid(y)$。它们的平均相对误差为 4.85E-02，平均绝对误差为 7.09E-05。

这种量级的误差可能会对通用科学计算造成严重影响，例如导致爱国者导弹系统失效（Skeel，1992年）。机器学习从业者自然会提出这样的问题：对于机器学习中的低精度模型而言，这种量级的浮点误差是否过大，以至于无法做出正确预测？

在机器学习社区，许多人认为“深度学习模型……‘非常能容忍低精度计算’”（Dean，2020）。机器学习研究人员积极探索了各种技术，旨在降低神经网络所用浮点数的精度，同时保持准确率。例如，Micikevicius 等人（2018年）提出了一种混合精度技术，将权重、激活值和梯度存储为 16 位数值，但以 32 位精度执行权重更新。Kalamkar 等人（2019年）采用了另一种混合精度技术，结合 32 位浮点数与 BFloat（Google Brain 的半精度格式），并使用一种张量修改方法，将 32 位数据流的低 16 位清零。

所有这些算法都采用混合精度，从少数可用精度中选择两种或更多精度，通常是半精度（16 位）和单精度（32 位）。据我们所知，机器学习领域很少将纯16 位神经网络作为独立解决方案使用。本研究着眼于 16 位纯神经

网络，并研究能否开箱即用，而无需参数调优或混合精度算法中常用的技术（如损失缩放）。我们的发现是肯定的。也就是说，纯 16 位神经网络即使不包含任何 32 位浮点组件，尽管其精度本质上有限，仍足以胜任机器学习的一项重要应用——分类问题。

我们的工作首先对误差容忍这一直观概念进行了形式化，并提出一个引理，从理论上保证在某些条件下，16 位“局部”能够取得与 32 位相同的分类结果。结合我们的初步观察，我们猜想 16 位和 32 位在处理分类问题时结果相近。随后，我们通过在深度神经网络（DNN）和卷积神经网络（CNN）问题上的大量实验验证了这一猜想。我们的贡献如下：

- 我们的目标是打破“普通的 16 位模型表现不佳”这一误区。我们证明，在纯 16 位下训练神经网络模型，无需采取额外措施进行“补偿”，也能取得有竞争力、甚至更高的准确率。
- 我们从理论上阐释半精度模型表现良好的原因，并提供支持我们分析的实证证据。
- 我们开展了大量实验，将各种纯16 位神经网络的性能与 32 位和混合精度网络进行比较，发现 16 位神经网络的表现可以与对应的 32 位网络相当。我们还找出了可能对 16 位模型的成功产生负面影响的因素。

2. 相关工作

人们提出了多种技术，在一定程度上保持准确率的同时降低机器学习模型的精度。与我们工作最契合的方法是降低这些模型所使用的浮点数的精度，但其他方法还包括算法修改、定点格式和硬件加速。

算法修改。这些方法旨在修改主算法的某些组件，以支持低精度计算。（De Sa 等人，2017年）的研究通过为权重和梯度指定不同的精度，提供了一种解决梯度下降中低精度相关问题的方法。此类方案借助专用硬件实现来加速这一混合精度计算过程。（Bjorck 等人，2021年）在强化学习中采用了类似的方法。这项研究提出了多种机制，以便在低精度下可靠地执行强化学习。例如，采用数值方法改进 Adam优化算法，以防止下溢。许多此类技术都是针对特定问题提出的，带有一定的临时性，但也可能适用于更广泛的场景。

定点格式。定点格式是另一种常用于表示实数的数字格式，尽管它并非标准格式。其他一些研究采用了定点格式，以利用其直观的表示方式以及对其他配置的高扩展性来实现低精度。为验证低精度在深度学习中的相关问题，(Gupta et al., 2015年) 的作者研究了降低精度时深度神经网络的行为。在这项工作中，实证结果表明，降低构成网络权重的实数的精度会使准确率平稳下降。(Chen et al., 2017年) 通过在训练期间将浮点数转换为低精度定点数，在定点数中进行神经网络训练。(Lin et al., 2016年) 也采用了浮点数的量化思想，其中作者将转换过程视为一个优化问题。优化目标是通过为每一层采用不同的位宽来缩减网络规模。在更偏向系统的方法中，(Kumar et al., 2020) 提出了编译器支持，用于将浮点数转换为低精度定点数。(Gopinath et al., 2019年) 也采用了基于编译器和语言的支持，以实现低精度数值。

虽然整数化量化与实数 定点数并不完全相同，但也可以视为一种特殊的定点格式，并已成为低精度神经网络设计中的一种可行选择。在（Wu et al., 2018）中，浮点权重经过量化后被转换为有符号整数表示。作者发现，采用整数化量化技术会产生类似正则化的效果，从而提高准确率。与这一方法类似，Das et al., 2018 的研究也提出使用整数运算来实现高准确率。这类整数运算能够有效地将浮点数转换为所谓的动态定点（DFP）格式。Jacob et al., 2017 的研究同样旨在将权重量化为整数。不过，权重仅在前向传播期间保持为整数，在反向传播时则变为浮点数，以处理细微的更新。另一项研究（Courbariaux et al., 2015）采用极端量化方法，将权重二值化为 -1 和 1。权重在前向传播和反向传播期间保持为二进制数，但在权重更新阶段变为浮点数。（Xiao et al., 2022）采用训练后量化方法，以减少大规模语言模型的内存占用。

（K¨oster 等人，2017年）提出了一种自适应数值格式，兼具浮点数和定点数的优势。该格式通过使用一个共享指数来实现，该指数会

在训练过程中更新。

混合精度。另一方面，混合精度 方法会以两种不同的精度保留标准浮点数。第一个在实践中取得成功的降精度浮点机制由（Micikevicius 等人，2018年）提出。在这项工作中，作者设计了一种混合精度训练方案：保留一组主全精度浮点权重，作为半精度对应权重的“原始副本”。虽然该技术确实缩短了运行时间，但其混合精度特性限制了所能实现的性能提升。（Wang 等人，2019年）提出了一种 4 位浮点架构，并辅以少量 8 位精度。除这种格式外，作者还设计了一种两阶段舍入过程，以抵消低精度格式导致的准确率下降。

硬件支持。最后，我们想指出，由于各自采用了不同的浮点/定点格式，许多此类工作无论是隐式还是显式地都需要硬件支持。大多数方法通常使用现场可编程门阵列和浮点运算单元来实现这些格式，但也有其他工作，例如（Sharma 等人，2017年），提出了专为处理浮点数位格式而设计的新型架构。前面提到的一些工作，例如（Wang 等人，2019年），也暗示了利用硬件辅助的可能性。

与这些先前的工作不同，我们明确聚焦于 IEEE 浮点格式，它是通用计算设备实际采用的标准。更具体地说，我们研究在没有外部支持的情况下，使用纯 16 位 IEEE 浮点格式训练神经网络的优缺点。

3. 背景

一般记号 实数和整数分别记为 $\mathbf{R}$ 和 $\mathbf{Z}$。给定向量 $x = (x_0, ..., x_{n-1}) \in \mathbf{R}^n$，其无穷范数或最大范数记为 $|x|_\infty$，定义为向量中的最大元素，即 $|x|_\infty \overset{\text{定义为}}{=} \max_i |x_i|$。

3.1. 浮点数表示

我们用 $\mathbf{F}_{16}$ 表示不包括 $\pm\infty$和 NaN（非数）的 16 位浮点数集合。根据 IEEE 754标准（IEEE计算机学会，2008），每个$x \in \mathbf{F}_{16}$ 都可以表示为

$$x = (-1)^s \times g_0.g_1...g_{10\ (2)} \times 2^e \tag{1}$$

其中 $s \in \{0, 1\}$、$g_i \in \{0, 1\}$（$0 \le i \le 10$），且 $e \in \mathbf{Z}$。我们分别将 s、$g_0.g_1....g_{10}$ 和 e 称为符号位、有效数和指数。它们满足 $g_0 \neq 0$ 和 $-14 \le e \le 15$。$g_0 = 0$ 且 $e = -14$ 的情况称为次正规数。

我们用 $\mathbf{F}_{32}$ 表示不包括 $\pm\infty$ 和非数的 32 位（单精度）浮点数集合。以下性质成立：

$$\mathbf{F}_{16} \subset \mathbf{F}_{32} \subset \mathbf{R} \tag{2}$$

也就是说，16 位或 32 位浮点数都是实数，并且 16 位浮点数可以精确表示为 32 位浮点数（通过补零实现）。表 3.1 列出了 16 位和 32 位浮点数的范围和精度。

表 *1.* 16 位和 32 位浮点格式的一些特性。

Type	Size	范围	机器 epsilon
Half	16 位	6.55E±4	4.88E-04
单精度	32 位	3.4E±38	5.96E-08

3.2. 浮点误差

表示无法写成式 1 的实数时，必须进行舍入。实数 在精度 下的舍入误差 指的是 $|x - \circ_p(x)|$，其中舍入操作，

$\circ_p : \mathbf{R} \to \mathbf{F}_p$，定义了与 x最接近的浮点数。也就是说，$\circ_p(x) \overset{\text{def}}{=} \operatorname{argmin}_{y \in \mathbf{F}_p} |x - y|$。[1] 舍入误差通常很小，量级约为机器精度（表 3.1），但它可能逐渐传播并变得更加显著。例如，浮点代码 sin(0.1) 会经过三次近似。首先，0.1 会舍入为某个精度 p下的浮点数 $\circ_p(0.1)$。随后，舍入误差会通过浮点代码 sin传播。最后，如果计算结果无法精确表示，输出也会再次舍入。

浮点误差通常用绝对误差或相对误差来衡量。本文关注分类问题，其中输出数值是介于 0 和 1 之间的概率。因此，我们使用绝对误差 $|x - y|$ 来量化两个浮点数 x 和 y之间的差异。

4. 理论

假设 M_{16} 和 M_{32} 是由相同的神经网络架构和超参数训练得到的 16 位和 32 位深度学习模型。为简化记法，我们将 M_{16}和 M_{32} 视为分类器或函数：给定输入 x（例如，一张图像），它们会返回最后一层的概率向量。

设 x 为任意选取的输入。假设 $M_{32}(x)$ 返回 $(p_0, \cdots, p_{N-1})$，而 $M_{16}(x)$ 返回 $(p'_0, \cdots, p'_{N-1})$。显然，由分类器对输入 x 得出的分类结果为

[1]为简单起见，该舍入操作的定义忽略了需要打破平局的情况。

表 2。 “浮点误差”和“误差容限”列分别表示 $\delta(M_{32},\ M_{16},\ x)$ 和 $\Gamma(M32,\ x)$ 的统计数据，其中 x 遍历 MNIST 数据集中的图像。

	浮点误差				误差容限			
训练轮数	最小值	Max	Mean	方差	Min	Max	Mean	方差
10	0.00E+00	1.68E-01	2.76E-03	5.62E-05	9.29E-05	1.00E+00	7.66E-01	7.58E-02
20	7.84E-15	2.07E-01	2.88E-03	8.53E-05	3.65E-05	1.00E+00	8.24E-01	6.31E-02
50	0.00E+00	3.82E-01	3.69E-03	1.97E-04	2.74E-06	1.00E+00	8.79E-01	4.72E-02
100	0.00E+00	5.64E-01	4.12E-03	3.27E-04	3.24E-04	1.00E+00	9.16E-01	3.42E-02
200	0.00E+00	6.75E-01	4.23E-03	4.76E-04	1.93E-04	1.00E+00	9.47E-01	2.19E-02
500	0.00E+00	9.35E-01	3.76E-03	6.18E-04	1.44E-03	1.00E+00	9.79E-01	7.72E-03
1000	0.00E+00	9.95E-01	3.14E-03	5.77E-04	1.91E-03	1.00E+00	9.91E-01	2.98E-03

分类器 M 给出的结果是

$$\text{pred}(M,x) \stackrel{\text{def}}{=} \text{argmax}_i\{p_i | p_i \in M(x)\}. \tag{3}$$

由于浮点误差， M_{32} 和 M_{16} 在 x 上的分类结果可能不同。为了量化这种差异，我们将浮点误差 定义如下。

定义 4.1。 给定 16 位分类器 M_{16}、32 位M_{32} 和一个输入 x，分类器之间的浮点误差 定义为

$$\delta(M_{32}, M_{16}, x) \stackrel{\text{def}}{=} |M_{32}(x) - M_{16}(x)|_\infty \tag{4}$$

这种差异对结果的影响程度，是我们在本文中探讨的一个重要问题。事实上，我们可以给出一个充分条件（记作 C），保证两个分类器之间不存在差异。

(C.) 如果 p_i 中最大值与次大值之间的差异大于浮点运算误差的两倍，那么两个分类器 M_{16} 和 M_{32} 在 x 上会得到相同的分类结果。

条件（C）的示意：假设对于一个有四个标签的分类问题，$M_{32}(x)$ =(0.8,0.1,0.05,0.05)。设该概率向量与 $M_{16}(x)$ 之间的最大误差为 δ。那么在最坏情况下，对于 16 位，0.8 可能降至 $0.8-\delta$，而第二大的概率变为 $0.1+\delta$。如果 $0.8-\delta > 0.1+\delta$，那么 M_{16} 和 M_{32} 在 x 上必定会得到相同的分类结果，例如，$M_{16}(x)$ =(0.7,0.15, 0.1, 0.05)。

下面，在引入误差容限的概念后，我们将对条件（C）进行形式化定义。

定义 4.2。分类器 M 相对于输入 x 的误差容限定义为最大概率与第二大概率之间的差值:

$$\Gamma(M,x) \stackrel{\text{def}}{=} p_0 - p_1, \tag{5}$$

其中 $p_0 = |M(x))|_\infty$，且 $p_1 = |M(x)\backslash p_0|_\infty$。

这里， $M_{32}(x)\backslash p_0$ 指的是 $M_{32}(x)$ 中的元素向量，但去除了 p_0。误差容限可以视为对预测稳定性的量化。我们有如下对应于条件 C 的引理。

引理 4.3。 考虑一个由一对 $(X,\ Y)$ 表征的分类问题，其中 X 是输入数据的空间， $Y = \{0,..N-1\}$ 是分类标签。假设某个学习算法在数据集 $D \subseteq X \times Y$ 上训练了一个 32 位模型M_{32}： $X \to \mathbf{F}_{32}^N$ 和一个 16 位模型 M_{16}： $X \to \mathbf{F}_{16}^N$。

我们有：若

$$\Gamma(M_{32}, x) \geq 2\delta(M_{32}, M_{16}, x) \tag{6}$$

那么 $\text{pred}(M_{32},\ x) = \text{pred}(M_{16},\ x)$。

证明。 令 $M_{32}(x)$ 为 $(p_0,\ ....p_{N-1})$。不失一般性，我们假设 p_0 是 $\{p_i\}$ $(0 \leq i \leq N-1)$ 中最大的一个。下文将 $\delta(M_{32},\ M_{16},\ x)$ 记作 δ。根据公式 6，我们有

$$\forall i \in \{1,...,N-1\}, p_0 - p_i \geq 2\delta. \tag{7}$$

设 $M_{16}(x)$ 为 $(p'_0,....p'_{N-1})$。那么对于每个 $i \in \{1,...,N-1\}$，都有

$$\begin{aligned} p'_0 &\geq p_0 - \delta && \text{By Eq. 4 and Def. of } p_0, p'_0 \\ &\geq p_i + \delta && \text{By Eq. 5 and Eq. 7} \\ &\geq p'_i && \text{By Eq. 4} \end{aligned}$$

因此， p'_0 仍然是 $M_{16}(x)$各元素中的最大值。 □

表 *3* . 比较 32 位和 16 位神经网络在 MNIST 数据集上的准确率和损失结果。

	训练准确率		测试准确率		训练损失		测试损失	
训练轮数	32 位	16 位	32 位	16 位	32 位	16 位	32 位	16 位
10	90.3%	90.0%	90.8%	91.0%	3.49E-01	3.69E-01	3.23E-01	3.39E-01
20	92.2%	92.3%	92.8%	92.8%	2.71E-01	2.94E-01	2.57E-01	2.74E-01
50	94.9%	95.3%	94.9%	94.7%	1.81E-01	2.10E-01	1.80E-01	2.04E-01
100	96.7%	96.4%	96.3%	95.8%	1.16E-01	1.49E-01	1.28E-01	1.52E-01
200	98.4%	97.3%	97.4%	97.3%	6.05E-02	9.32E-02	8.97E-02	1.10E-01
500	99.8%	99.1%	97.7%	98.1%	1.38E-02	4.00E-02	7.87E-02	8.54E-02
1000	100.0%	99.8%	97.8%	98.1%	2.97E-03	2.04E-02	9.02E-02	8.29E-02

下面我们通过一个在 MNIST 上训练的简单神经网络来说明引理 4.3。我们的 32 位实现包含三个全连接层，末尾接一个 Softmax 层。我们的 16 位实现采用相同的架构，不同之处在于，所有浮点运算都以 16 位执行。

表 4 展示了我们的误差容限 Γ 和浮点误差 δ 结果。请注意，浮点误差的均值约为 1E-3，方差为 1E-5 或 1E-4；误差容限为 1E-1，方差为 1E-2。因此，可以认为公式 6，即 $\Gamma > 2\delta$，对 MNIST 中的大多数数据成立。该表还显示，在某些边界情况下，浮点误差可能大于误差容限。因此，我们预计我们的 16 位和 32 位实现将具有接近但不同的准确率结果。

展示了训练和测试中的结果。表 4 展示了 MNIST 中 16 位和 32 位实现的准确率和损失结果对比。我们可以持续看到，16 位结果的准确率接近 32 位模型的准确率，有时甚至更高。这一结果促使我们研究，对于更复杂的神经网络，16 位模型是否与 32 位模型相似。

理论上，如果数据集中的 80% 数据满足公式 6，根据引理 4.3，32 位模型和 16 位模型至少有 80% 的分类结果相同。这里的主要挑战在于，我们无法确定公式 6 是否始终成立，也无法确定满足该公式的数据所占的比例。我们认为，对于复杂的神经网络，大多数数据都满足公式 6。这是因为分类问题的损失函数是形式为 $-\Sigma\log(p_i)$ 的交叉熵，它应当在训练过程中引导 p_i 趋近于 1，进而使误差容限 Γ 相较于较小的 δ大得多。事实上，表 4 显示，浮点误差（δ）比 Γ小近两个数量级。虽然随着训练轮数增加，这一差距可能会缩小，但两者之间的差异仍足够大，能够满足该引理的条件。

基于这一理论进展和观察结果，我们提出以下猜想。

> 在不存在涉及浮点溢出/下溢的显著误差时，用于分类问题的 16 位神经网络，其准确率将接近 32 位神经网络的准确率。

我们预计，16 位浮点数的表示范围较小（表 3.1），因此上溢或下溢可能导致浮点误差变得显著。这一猜想或许令人意外，因此我们将在下一节进行深入验证。

5. 实验

我们旨在比较深度神经网络（DNN）[2] 和卷积神经网络（CNN）模型中 16 位运算与 32位运算的性能。我们在三种卷积神经网络模型上进行实验，包括 AlexNet（Krizhevsky 等，2012）、VGG16（Simonyan 和 Zisserman，2015年）以及 ResNet-34（He 等，2016 年）。我们将考察不同的精度设置如何影响模型的计算时间和准确率。与上一节的案例研究不同，所有实验均使用 100 个训练轮数，并逐步将批量大小从 64 增加到 384，以观察其对 16 位训练的影响。本研究实验使用的所有随机种子均固定，以便进行比较。实验在 NVIDIA 的 RTX3080 笔记本电脑 GPU 上进行。

表 4 展示了最能代表我们工作的结果。后续小节将对这些结果进行更详细的描述和分析。

[2]虽然深度神经网络涵盖卷积神经网络，但我们使用“深度神经网络”一词指代全连接的非卷积神经网络。

图 *1.* 使用深度神经网络进行 MNIST 分类的 Top-1 准确率（上排）和计算时间（下排）

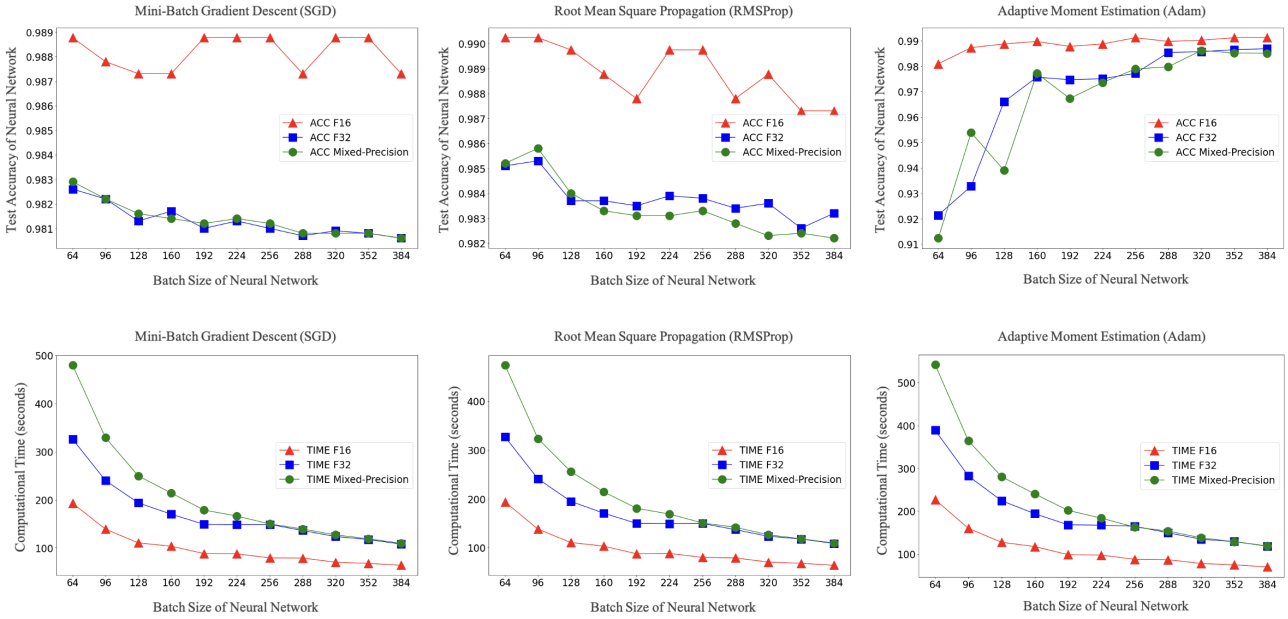


图 *2.* CIFAR-10 分类的 Top-1 准确率、Top-2 准确率和计算时间

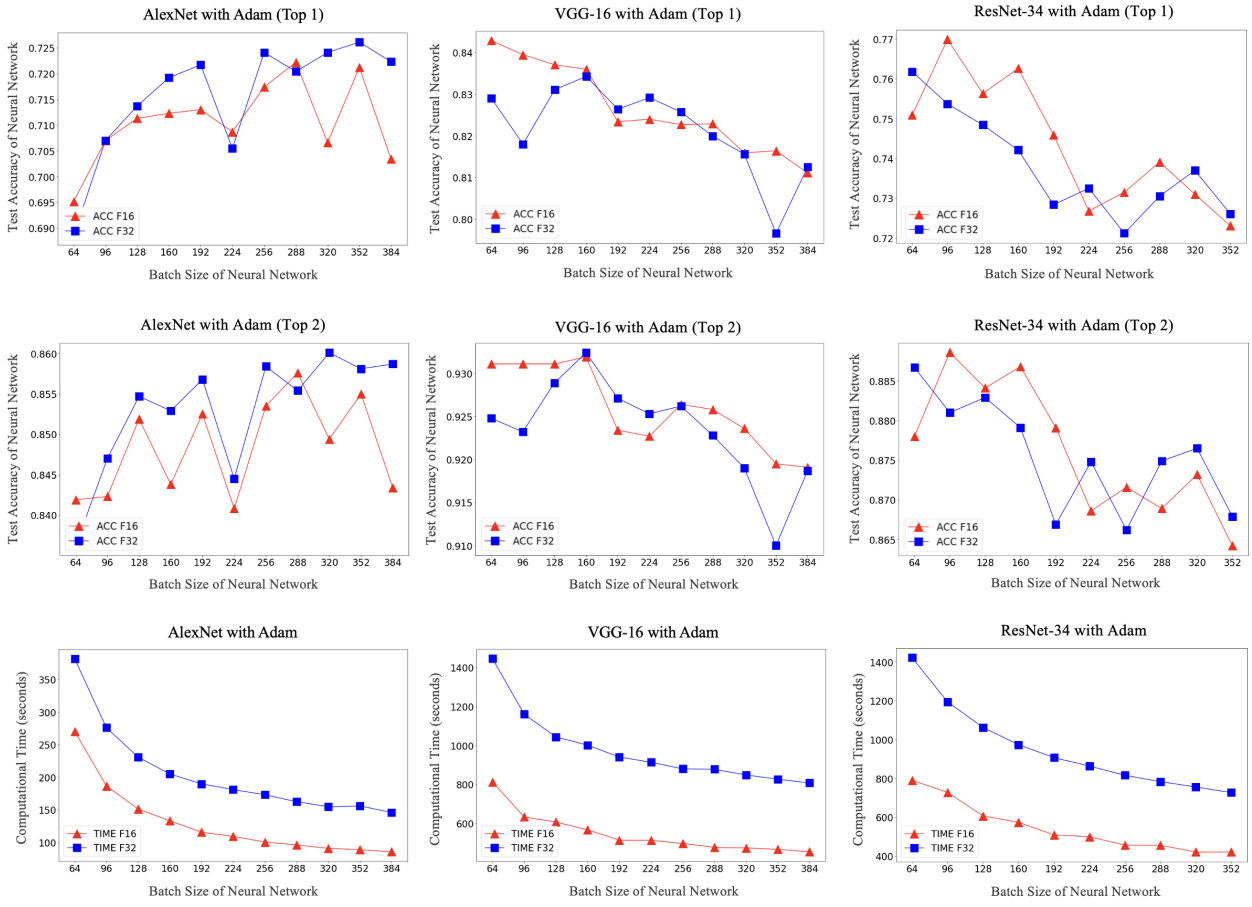


表 *4*。三种卷积神经网络的耗时和准确率表现总结。

模型	结果	FP32	FP16
AlexNet	Time	381s	270s
	准确率	68.9%	69.5%
VGG16	Time	1445s	812s
	准确率	82.9%	84.3%
ResNet-34	时间	1914s	1058s
	准确率	76.6%	76.1%

5.1. DNN 实验

在深度神经网络实验中，我们训练了一个具有三个隐藏层的深度神经网络，以执行 MNIST 分类任务，并比较 16 位模型与 32 位模型的性能。深度神经网络中的每一层都包含 4096 个神经元，其输出经过 ReLU 激活后馈入下一层。其他研究（如 Micikevicius 等人，2018年）无论整体精度设置如何，均使用 32 位 softmax 层，以防止潜在的数值不稳定性；但我们坚持使用 16 位 softmax，以观察“纯”16 位模型相较于 32 位模型的表现。我们还在 RMSProp、Adam 和随机梯度下降之间改变优化器类型，以考察它们对性能的影响。此外，我们通过该实验确认，只要 ϵ 设置得当，不仅随机梯度下降，其他优化器（如 RMSProp 和 Adam）也可以在 16 位下使用。

本节的所有实验中，我们使用学习率 10^{-3}，并将 RMSProp 和 Adam 中的 ϵ 也固定为 10^{-3}。其他设置下的实验请参阅附录。最后，除了 32 位基线之外，我们还将本文与 (Micikevicius 等人，2018年) 提出的混合精度训练算法（由 TensorFlow 提供）进行了比较。图1显示，在保持测试准确率相近的同时，16 位深度神经网络的计算时间优于 32 位和混合精度。具体而言（见附录中的图3），与 32 位相比，16 位随机梯度下降的计算时间减少了 40.9%，测试准确率则提高了 0.6%；而在批量大小最小、为 64 时，与混合精度相比，16 位的计算时间减少了 59.8%，准确率提高了 0.59%。其他批量大小下也呈现出一致的趋势，不过幅度较小（Keskar 等人，2017年）。

在优化方面，与 32 位和混合精度相比，16 位 RMSProp 的运行时间分别缩短了 40.4% 和 59.6%，准确率分别提高了 0.3% 和 0.4%。与 32 位和混合精度相比，16 位 Adam 的运行时间分别缩短了 41.6% 和 58.1%，准确率分别提高了 6.4% 和 7.5%。在总共 33 个使用三种优化器计算的实验组中，有 31 个实验组的运行时间相比 32 位缩短了 40% 以上，同时保持了相近的准确率。16 位深度神经网络的计算时间和测试准确率均优于 32 位和混合精度。

5.2. 卷积神经网络实验

我们进行了 16 位卷积神经网络实验，以确定 16 位精度是否足以训练更复杂的图像分类问题，以及其数值结果与 32 位精度有何差异。我们使用了 CIFAR10数据集和三种卷积神经网络模型：AlexNet、VGG16 和 ResNet-34。所有实验均在 16 位环境下使用 Adam 进行。由于批量归一化（BN）层尚未作为现成模块在 16 位环境中实现，因此实验重点关注无需批量归一化层即可使用的卷积神经网络模型。有关 16 位批量归一化实现的处理方法，请参阅附录。

5.2.1. 训练时间和准确率

图 1 显示，在计算时间和测试准确率方面，16 位卷积神经网络优于 32 位和混合精度。图 2 显示，16 位运算也可应用于用于图像分类的卷积神经网络模型。我们发现，16 位卷积神经网络的准确率与 32 位相近。

AlexNet 在最小批量大小 64 下，与 32 位相比，16 位 AlexNet 的 Top-1 和 Top-2 准确率结果分别提高了 0.9% 和 0.6%，训练时间缩短了 29.1%。与 32 位版本相比，16 位 AlexNet 的运行时间缩短了 29% 以上。

VGG-16 在 Top-1 和 Top-2 准确率方面，与 32 位相比，16 位 VGG16 分别提高了 1.6% 和 0.6%，学习时间缩短了 43.7%。与 32 位版本相比，16 位 VGG16 的计算速度均提升了 40% 以上。

ResNet-34 与 32 位相比，16 位 ResNet-34 的 Top-1 和 Top-2 准确率分别下降了 0.6% 和 0.4%，计算时间缩短了 44.7%。16 位 ResNet-34 的运行时间缩短了 39% 以上。

总体而言，在 33 组实验中，与 32 位相比，16 位 top1 准确率的最大降幅为 2.6%（AlexNet，批量大小：384），最大增幅为 2.7%（ResNet-34，批量大小：192）。与 32 位相比，运行时间的最小降幅为 29.1%（AlexNet，批量大小：64），最大降幅为 45.6%（VGG16，批量大小：288）。换言之，所有实验中使用的 16 位实验组，其计算速度至少提高了 29.1%。

这项实验表明，在

使用卷积神经网络模型进行图像分类时，采用 16 位运算可以显著缩短运行时间（无论是训练还是测试），同时保持与 32 位运算相近或更高的准确率。我们的结果证明了神经网络在低精度设置下训练卷积神经网络模型的效率。

5.2.2. 模型大小

已训练模型权重的保存是当代深度学习的一个重要方面。

表5. 以 16 位和 32 位格式保存的卷积神经网络模型大小。

模型	FP16	FP32
AlexNet	41.99 MB	83.94 MB
VGG16	67.34 MB	134.61 MB
ResNet-34	171.99 MB	343.87 MB

如果具有相近准确率的模型所需的存储空间更小，那么这对于模型训练和存储都很有益。表5显示，存储的模型大小约为 32 位模型大小的一半，充分体现了半精度带来的尺寸缩减。16 位神经网络能够在保持相近准确率的同时，将模型大小缩减一半。这为 16 位模型采用更复杂、更准确的架构以获得更好的结果提供了可能。

5.3. 局限性与讨论

本小节报告了我们在使用16 位神经网络时发现的局限性。

轻量级超参数调优。 在我们的实验中，无需对16 位神经网络训练的超参数进行调优，例如学习率，但优化器设置中的 epsilon 除外。SGD（Goodfellow 等人，2016 年）优化器不包含 epsilon，因此可以直接使用。对于我们在实验中测试的其他优化器——RMSProp（Hinton 等人）和 Adam（Kingma 和 Ba，2015年）——则需要更改 epsilon，因为其默认值（在 TensorFlow 中）为 1E-7，很容易导致16 位训练出现显著的不准确。

具体而言，参数 epsilon 对应于以下权重更新中的 ϵ：

$$\text{RMSProp}: w_t = w_{t-1} - \eta \frac{g_t}{\sqrt{v_t} + \epsilon}$$

$$\text{ADAM}: w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

引入这些优化器 ϵ 以增强数值稳定性，但分母中的 ϵ = 1 E-7 会导致浮点

当 RMSProp 中的 v_t in 或 Adam 中的 $\hat{}v_t$ in 接近 0 时，会发生上溢。如前所述，对于 16 位训练，我们将其设为 ϵ = lE-3。

缺少16位批归一化。 TensorFlow 当前的批归一化层不直接支持纯16 位运算。该批归一化层可能源自混合精度相关工作：这些工作会将16 位输入值转换为32 位以进行计算，然后再向下转换为16 位。这种类型转换只会由于中间的32 位计算而产生运行时开销。

由于本研究旨在报告纯 16 位神经网络的结果，因此我们必须自行实现 16 位批归一化层。我们实现的 16 位批归一化可见附录。

批量大小。 我们的卷积神经网络实验表明，批量大小越大，16 位神经网络的准确率越低。这可能是因为在小批量中对更多样本的损失求平均时会产生精度损失。这或许只是一个小限制，因为在内存受限的环境中，神经网络通常不会使用较大的批量大小。

尽管存在这些限制，我们已经证实，经过少量微调（ϵ 在优化器方面）后，16 位神经网络的学习效果与 32 位神经网络相当。因此，我们认为，机器学习从业者在解决优化问题时可以轻松受益于 16位神经网络，因为它速度更快、内存占用更少，同时能达到与 32 位神经网络相近的准确率。我们无需像使用 32 位神经网络那样投入同等的时间和空间，就能构建出成本效益比更高的网络。

6. 结论

在这项工作中，我们证明了对于分类问题，16位浮点神经网络的准确率可以接近 32 位神经网络。我们提出了关于二者准确率相近的猜想，并通过理论和实验证实了这一猜想。我们的实验还表明，准确率可能会略有提升，这或许是降低精度带来的正则化效应所致。这些发现表明，使用 16 位精度比通常认为的要安全、高效得多：运行时间和内存消耗显著降低，而准确率几乎不受影响或完全不受影响。未来，我们计划拓展这项工作，验证其他类型的架构和问题（如生成模型和回归）是否具有类似特性。

参考文献

Bjorck, J.、Chen, X.、Sa, C. D.、Gomes, C. P. 和 Weinberger, K. Q. 低精度强化学习：以半精度运行软演员-评论家算法。载于《……论文集》

*Inte*际机器学习会议，2 021.

Chen, X., Hu, X., Zhou, H. 和 Xu, N. FxpNet：使用定点表示训练深度卷积神经网络。载于国际神经网络联合会议论文集，2017年。

Courbariaux, M., Bengio, Y. 和 David, J.-P. BinaryConnect：在传播过程中使用二进制权重训练深度神经网络。载于神经信息处理系统论文集，2015年。

Das, D., Mellempudi, N., Mudigere, D., Kalamkar, D., Avancha, S., Banerjee, K., Sridharan, S., Vaidyanathan, K., Kaul, B., Georganas, E., Heinecke, A., Dubey, P., Corbal, J., Shustrov, N., Dubtsov, R., Fomenko, E. 和 Pirogov, V. 使用整数运算进行卷积神经网络的混合精度训练。载于国际学习表征会议论文集，2018年。

De Sa, C., Feldman, M., R´e, C., and Olukotun, K. 理解并优化异步低精度随机梯度下降。载于国际计算机体系结构研讨会论文集，2017年。

Dean, J. 深度学习革命及其对计算机体系结构和芯片设计的影响。载于*IEEE*国际固态电路会议论文集，2020年。

Goodfellow, I., Bengio, Y., andCourville, A. 深度学习。麻省理工学院出版社，2016年。http://www.deeplearningbook.org。

Gopinath, S., Ghanathe, N., Seshadri, V., and Sharma, R. 将千字节级机器学习模型编译到微型物联网设备上。载于编程语言设计与实现会议论文集，2019年。

Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. 使用有限数值精度进行深度学习。载于国际机器学习会议论文集，2015年。

He, K., Zhang, X., Ren, S., 和 Sun, J. 深度残差网络中的恒等映射。载于欧洲计算机视觉会议论文集，2016年。

Hinton, G., Srivastava, N., 和 Swersky, K. 面向机器学习的神经网络，第 6e 讲。网址 http://www.cs.toronto.edu/~tijmen/ csc321/slides/lecture_slides_lec6.pdf。

IEEE计算机学会。《IEEE浮点算术标准》。*IEEE 754-2008*标准，第1–70页，2008年。doi: 10.1109/IEEESTD.2008.4610935。

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H. 和 Kalenichenko, D. 面向高效纯整数推理的神经网络量化与训练。载于*IEEE*计算机视觉与模式识别会议论文集，2017年。Kalamkar, D., Mudigere, D., Mellempudi, N., Das, D., Banerjee, K., Avancha, S., Vooturi, D. T., Jammala-madaka, N., Huang, J., Yuen, H., Yang, J., Park, J., Heinecke, A., Georganas, E., Srinivasan, S., Kundu, A., Smelyanskiy, M., Kaul, B. 和 Dubey, P. 关于BFLOAT16 在深度学习训练中的研究，2019年。URL https://arxiv.org/abs/1905.12322。

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., 和 Tang, P. T. P. 深度学习的大批量训练：泛化差距与尖锐极小值。载于国际学习表征会议论文集，2017年。

Kingma, D.P. 和 Ba., J. L. Adam：一种随机优化方法。载于国际学习表征会议论文集，2015年。

Krizhevsky, A.、Sutskever, I. 和 Hinton, G. E. 使用深度卷积神经网络进行 ImageNet 分类。载于神经信息处理系统会议论文集，2012年。

Kumar, A.、Seshadri, V. 和 Sharma, R. Shiftry：仅用 2kb 随机存取存储器进行循环神经网络推理。美国计算机协会《程序设计语言》会议论文集，4（面向对象编程、系统、语言与应用会议）：182:1–182:30，2020年。

K¨oster, U.、Webb, T. J.、Wang, X.、Nassar, M.、Bansal, A. K.、Constable, W. H.、Elibol, O. H.、Gray, S.、Hall, S.、Hornof, L.、Khosrowshahi, A.、Kloss, C.、Pai, R. J. 和 Rao, N. Flexpoint：一种用于高效训练深度神经网络的自适应数值格式。载于神经信息处理系统会议论文集，2017年。

Lin, D. D., Talathi, S. S., and Annapureddy, V. S. 深度卷积网络的定点量化。载于国际机器学习会议论文集，2016年。

Micikevicius, P., Narang, S., Alben, J., Diamos, G.,Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., and Wu, H. 混合精度训练。载于国际学习表征会议论文集，2018年。

Sharma, H.、Park, J.、Suda, N.、Lai, L.、Chau, B.、Kim, J. K.、Chandra, V. 和 Esmaeilzadeh, H. 位融合：用于加速深度神经网络的按位动态可组合架构。载于《国际计算机体系结构研讨会论文集》，2017年。

Simonyan, K. 和 Zisserman, A. 用于大规模图像识别的超深度卷积网络。载于《国际学习表征会议论文集》，2015年。Skeel, R. 舍入误差与爱国者导弹。《*SIAMNews*》，（4），1992年。Wang, N.、Chen, C.-Y. 和 Gopalakrishnan, K. 深度神经网络的超低精度训练。载于《神经信息处理系统论文集》，2019年。Wu, S.、Li, G.、Chen, F. 和 Shi, L. 深度神经网络中的整数训练与推理。载于《国际学习表征会议论文集》，2018年。Xiao, G.、Lin, J.、Seznec, M.、Demouth, J. 和 Han, S. SmoothQuant：面向大语言模型的准确高效训练后量化，2022年。URL：https://arxiv.org/abs/2211.10438。

A. 额外实验

A.1. Top-1 准确率和时间结果

图*3*。 MNIST 分类的 Top-1 准确率和计算时间

DNN Top-1 Test Accuracy

Batch Size	Optimizers											
	Mini-Batch Gradient Descent (SGD)				Root Mean Square Propagation (RMSProp)				Adaptive Moment Estimation (Adam)			
	Float16	Float32	Mixed Precision	Relative Error (16-32)	Float16	Float32	Mixed Precision	Relative Error (16-32)	Float16	Float32	Mixed Precision	Relative Error (16-32)
64	98.8%	98.2%	98.2%	6.31E-03	99.0%	98.6%	98.5%	3.85E-03	98.0%	92.1%	91.2%	6.45E-02
96	98.7%	98.2%	98.2%	5.59E-03	98.9%	98.5%	98.4%	4.05E-03	98.7%	93.2%	95.3%	5.85E-02
128	98.7%	98.1%	98.1%	6.11E-03	98.9%	98.3%	98.3%	6.61E-03	98.8%	96.6%	93.9%	2.34E-02
160	98.7%	98.1%	98.1%	5.70E-03	98.8%	98.3%	98.3%	5.18E-03	98.9%	97.5%	97.7%	1.43E-02
192	98.8%	98.1%	98.1%	7.84E-03	98.7%	98.3%	98.2%	4.47E-03	98.7%	97.4%	96.7%	1.34E-02
224	98.8%	98.1%	98.1%	7.54E-03	98.9%	98.2%	98.4%	7.32E-03	98.8%	97.5%	97.3%	1.40E-02
256	98.8%	98.1%	98.1%	7.84E-03	98.9%	98.3%	98.3%	6.61E-03	99.1%	97.7%	97.8%	1.44E-02
288	98.7%	98.0%	98.0%	6.72E-03	98.7%	98.3%	98.2%	4.57E-03	98.9%	98.5%	97.9%	4.36E-03
320	98.8%	98.0%	98.0%	8.05E-03	98.8%	98.2%	98.2%	6.61E-03	99.0%	98.5%	98.6%	4.56E-03
352	98.8%	98.0%	98.0%	8.15E-03	98.8%	98.2%	98.3%	6.31E-03	99.1%	98.6%	98.5%	4.76E-03
384	98.7%	98.0%	98.0%	6.83E-03	98.7%	98.2%	98.2%	5.19E-03	99.1%	98.6%	98.5%	4.45E-03

DNN Training Time (Seconds)

Batch Size	Optimizers											
	Mini-Batch Gradient Descent (SGD)				Root Mean Square Propagation (RMSProp)				Adaptive Moment Estimation (Adam)			
	Float16	Float32	Mixed Precision	Relative Error (16-32)	Float16	Float32	Mixed Precision	Relative Error (16-32)	Float16	Float32	Mixed Precision	Relative Error (16-32)
64	192.5	325.9	479.6	4.09E-01	194.0	326.0	480.6	4.04E-01	226.8	388.9	541.7	4.16E-01
96	138.8	239.9	329.1	4.21E-01	138.2	239.6	323.2	4.23E-01	159.7	281.9	364.3	4.33E-01
128	110.1	193.4	249.5	4.30E-01	110.7	193.9	254.4	4.28E-01	127.2	224.3	280.3	4.32E-01
160	103.3	169.9	213.9	3.91E-01	103.6	169.5	214.1	3.88E-01	117.2	193.9	240.2	3.95E-01
192	88.0	148.8	178.5	4.08E-01	87.8	148.2	180.9	4.07E-01	98.4	168.0	201.7	4.14E-01
224	87.4	148.2	165.9	4.10E-01	88.0	148.3	165.9	4.06E-01	97.2	167.2	183.7	4.18E-01
256	79.0	148.6	149.5	4.68E-01	80.2	149.0	148.3	4.61E-01	87.6	164.7	162.8	4.68E-01
288	78.8	136.0	139.1	4.20E-01	78.5	136.6	139.3	4.24E-01	86.7	149.6	153.6	4.20E-01
320	69.9	123.4	126.8	4.33E-01	70.3	122.9	125.8	4.27E-01	77.8	134.5	137.8	4.21E-01
352	67.6	117.2	118.7	4.22E-01	68.2	117.4	117.3	4.18E-01	74.6	129.0	128.5	4.21E-01
384	63.6	107.6	109.3	4.08E-01	63.7	108.2	108.7	4.10E-01	69.7	118.1	118.8	4.09E-01

A.2. 使用16位批归一化

正如正文中所述，我们在第5节提供的卷积神经网络结果基于不包含批归一化（BN）层的架构。图4展示了不含 BN 层的16 位和32 位卷积神经网络模型的性能。排除这些层的主要原因是，目前没有现成的16 位批归一化层实现。

作为补救措施，我们手动实现了这些方法，并展示了结果。我们选择将这一结果从正文移至此处，是因为无法充分确认该实现是否确实完全采用 16 位计算。也就是说，尽管我们在 Python 层面采取了一切可能的预防措施，以确保遵循 16 位计算，但我们并不清楚库层面（例如，libcuda 或 cublas）是否会发生任何此前未曾预料到的升位转换。因此，图5中的结果来自据我们所知采用 16 位计算的神经网络。

从图中可以看出，采用 16 位批归一化层后，仍呈现出与正文中所示相似的模式。除 AlexNet 外，16 位网络的表现与 32 位模型相近，甚至更优。

图 *4.* 不使用批归一化层时 CIFAR-10 分类的 Top-1 和 Top-2 准确率及计算时间

CNN Top-1 Test Accuracy

Batch Size	Convolutional Neural Network Models								
	AlexNet			VGG-16			ResNet-34		
	Float16	Float32	Relative Error (16-32)	Float16	Float32	Relative Error (16-32)	Float16	Float32	Relative Error (16-32)
64	69.5%	68.8%	9.58E-03	84.2%	82.9%	1.67E-02	76.0%	76.5%	6.39E-03
96	70.7%	70.7%	0.00E+00	83.9%	81.8%	2.62E-02	75.0%	76.1%	1.43E-02
128	71.1%	71.3%	3.36E-03	83.7%	83.1%	7.21E-03	76.9%	75.3%	2.14E-02
160	71.2%	71.9%	9.59E-03	83.6%	83.4%	2.03E-03	75.6%	74.8%	1.05E-02
192	71.3%	72.1%	1.20E-02	82.3%	82.6%	3.63E-03	76.2%	74.2%	2.76E-02
224	70.8%	70.5%	4.53E-03	82.4%	82.9%	6.27E-03	74.5%	72.8%	2.40E-02
256	71.7%	72.4%	9.25E-03	82.2%	82.5%	3.63E-03	72.6%	73.2%	7.78E-03
288	72.2%	72.0%	2.49E-03	82.2%	81.9%	3.65E-03	73.1%	72.1%	1.42E-02
320	70.6%	72.4%	2.41E-02	81.5%	81.5%	3.67E-04	73.9%	73.0%	1.17E-02
352	72.1%	72.6%	6.74E-03	81.6%	79.6%	2.49E-02	73.1%	73.7%	8.14E-03
384	70.3%	72.2%	2.61E-02	81.1%	81.2%	1.72E-03	72.3%	72.6%	4.13E-03

CNN Top-2 Test Accuracy

Batch Size	Convolutional Neural Network Models								
	AlexNet			VGG-16			ResNet-34		
	Float16	Float32	Relative Error (16-32)	Float16	Float32	Relative Error (16-32)	Float16	Float32	Relative Error (16-32)
64	84.1%	83.6%	6.81E-03	93.1%	92.4%	6.81E-03	88.7%	89.1%	4.15E-03
96	84.2%	84.7%	5.54E-03	93.1%	92.3%	8.55E-03	87.8%	88.6%	9.81E-03
128	85.1%	85.4%	3.27E-03	93.1%	92.8%	2.36E-03	88.8%	88.1%	8.62E-03
160	84.3%	85.2%	1.06E-02	93.1%	93.2%	5.36E-04	88.4%	88.2%	1.35E-03
192	85.2%	85.6%	5.01E-03	92.3%	92.7%	3.99E-03	88.6%	87.9%	8.75E-03
224	84.0%	84.4%	4.38E-03	92.2%	92.5%	2.80E-03	87.9%	86.6%	1.40E-02
256	85.3%	85.8%	5.70E-03	92.6%	92.6%	2.15E-04	86.8%	87.4%	7.08E-03
288	85.7%	85.5%	2.57E-03	92.5%	92.2%	3.25E-03	87.1%	86.6%	6.23E-03
320	84.9%	86.0%	1.24E-02	92.3%	91.9%	5.00E-03	86.8%	87.4%	6.85E-03
352	85.5%	85.8%	3.61E-03	91.9%	91.0%	1.04E-02	87.3%	87.6%	3.76E-03
384	84.3%	85.8%	1.78E-02	91.9%	91.8%	4.35E-04	86.4%	86.7%	4.26E-03

CNN Training Time

Batch Size	Convolutional Neural Network Models								
	AlexNet			VGG-16			ResNet-34		
	Float16	Float32	Relative Error (16-32)	Float16	Float32	Relative Error (16-32)	Float16	Float32	Relative Error (16-32)
64	270.1	381.4	2.91E-01	812.4	1445.3	4.37E-01	1057.6	1914.4	4.47E-01
96	186.5	276.0	3.24E-01	632.9	1160.2	4.54E-01	789.8	1422.2	4.44E-01
128	151.3	230.8	3.44E-01	607.2	1043.6	4.18E-01	727.6	1194.0	3.90E-01
160	133.1	205.1	3.51E-01	567.0	1001.4	4.33E-01	606.4	1062.6	4.29E-01
192	115.7	189.6	3.89E-01	513.7	940.6	4.53E-01	573.5	973.7	4.11E-01
224	109.2	181.2	3.97E-01	513.4	913.5	4.38E-01	508.8	906.9	4.38E-01
256	100.5	173.0	4.19E-01	497.1	879.9	4.35E-01	499.1	863.5	4.21E-01
288	96.1	162.5	4.08E-01	476.9	877.5	4.56E-01	455.9	815.9	4.41E-01
320	90.9	154.7	4.12E-01	474.0	848.3	4.41E-01	455.2	783.1	4.18E-01
352	88.7	155.9	4.30E-01	466.7	826.9	4.35E-01	420.2	755.7	4.43E-01
384	85.6	145.9	4.13E-01	454.2	807.4	4.37E-01	420.5	726.6	4.21E-01

图5. 使用 16 位批归一化层的结果

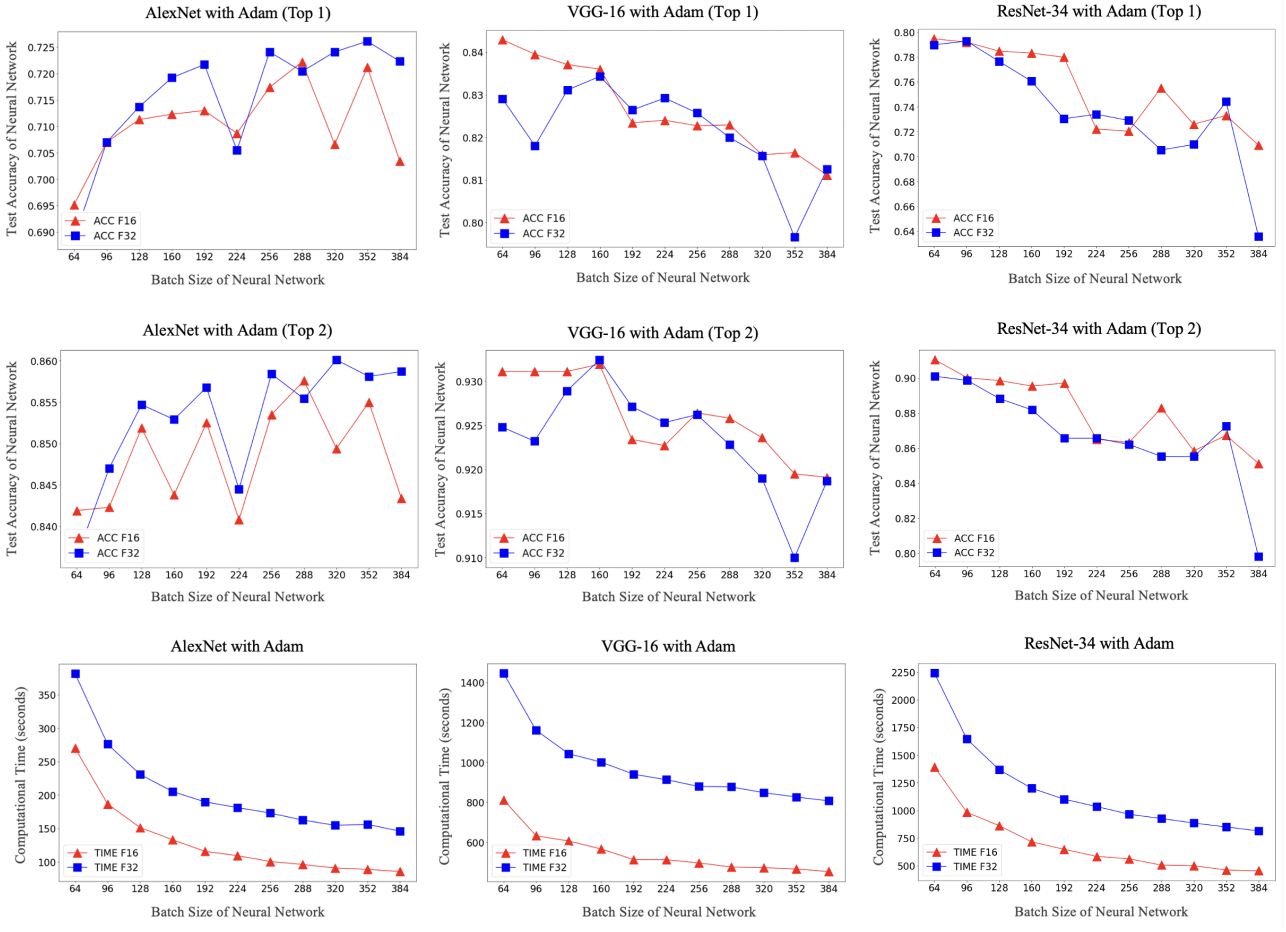


图 6. ResNet-34 使用批归一化层在 CIFAR-10 上进行分类的 Top-1 和 Top-2 准确率及计算时间

Batch Size	ResNet-34								
	Top 2 Test Accuracy			Top 2 Test Accuracy			Training Time		
	Float16	Float32	Relative Error (16-32)	Float16	Float32	Relative Error (16-32)	Float16	Float32	Relative Error (16-32)
64	79.4%	78.9%	6.20E-03	91.0%	90.0%	1.04E-02	1393.8	2241.1	3.78E-01
96	79.2%	79.2%	1.13E-03	90.0%	89.8%	1.66E-03	982.5	1644.9	4.02E-01
128	78.4%	77.6%	1.05E-02	89.8%	88.8%	1.15E-02	861.1	1366.7	3.69E-01
160	78.3%	76.0%	2.97E-02	89.5%	88.1%	1.54E-02	715.2	1199.7	4.03E-01
192	77.9%	73.0%	6.77E-02	89.7%	86.5%	3.62E-02	647.9	1101.1	4.11E-01
224	72.2%	73.4%	1.60E-02	86.5%	86.5%	6.93E-04	584.2	1034.5	4.35E-01
256	72.0%	72.9%	1.19E-02	86.3%	86.2%	1.39E-03	562.1	965.1	4.17E-01
288	75.5%	70.5%	7.08E-02	88.2%	85.5%	3.22E-02	505.3	926.0	4.54E-01
320	72.5%	70.9%	2.28E-02	85.8%	85.5%	3.85E-03	499.8	885.4	4.35E-01
352	73.3%	74.4%	1.47E-02	86.7%	87.2%	6.07E-03	460.4	849.8	4.58E-01
384	70.9%	63.5%	1.15E-01	85.1%	79.8%	6.65E-02	454.5	814.1	4.41E-01