诱导人工智能芯片模拟人类大脑：详细的性能分析

Lennart P.L. Landsmeer*量子与计算机工程系 代尔夫特理工大学荷兰代尔夫特

神经科学系 伊拉斯姆斯医学中心鹿特丹，荷兰 ORCID 0000-0003-0010-7249

Max C.W. Engelen*神经科学系 伊拉斯姆斯医学中心鹿特丹，荷兰 & *Maxeler*物联网实验室代尔夫特，荷兰 ORCID 0000-0002-5762-1276

Rene Miedema ChristosStrydis量子与计算机工程系 量子与计算机工程系 代尔夫特理工大学代尔夫特理工大学荷兰代尔夫特 荷兰代尔夫特 & &神经科学系 神经科学系 伊拉斯姆斯医学中心 伊拉斯姆斯医学中心荷兰鹿特丹 荷兰鹿特丹 ORCID 0000-0002-0447-1083 ORCID 0000-0002-0935-9322

arXiv:2301.13637v1 [cs.LG] 31 Jan 2023

摘要——**为挑战 Nvidia 的垄断地位，专用人工智能加速器芯片开始涌现，用于应对现代深度神经网络（DNNs）的推理，尤其是训练给现代计算机带来的计算挑战。研究人员已经开展了大量研究，评估这些竞争者在各种 DNN 模型类型上的性能。然而，人工智能专家清楚地认识到当前 DNNs 的局限性，并一直致力于推动第四次人工智能浪潮；可以说，这一浪潮将依赖更多生物启发模型，主要是脉冲神经网络（SNNs）。与此同时，图形处理器一直被广泛用于在计算神经科学领域模拟此类模型，但人工智能芯片尚未在这类工作负载上接受测试。本文旨在填补这一重要空白，通过评估多种前沿人工智能芯片（Graphcore IPU、GroqChip、配备张量核心的 Nvidia GPU 以及 Google TPU）来模拟一个高度具备生物学细节的脑区模型——下橄榄核（IO）。该 IO 应用通过改变计算密度、内存需求和浮点数值精度，对不同人工智能平台进行压力测试，以凸显架构权衡。我们的性能分析表明，该仿真问题与 GPU 和 TPU 架构的适配性极佳；对于包含 125,000 个细胞的网络，二者相较于中央处理器运行时间分别实现了 28 倍和 1,208 倍的加速。凭借这一速度，TPU 创下了最大规模实时下橄榄核仿真的新纪录。GroqChip 在小型网络上的性能优于这两个平台，但由于部分浮点运算采用了降低的精度，其目前尚不能用于脑模拟。**

索引词—**人工智能加速器、图形处理器、脑模拟、计算机架构**

I. 引言

迄今为止，与中央处理器相比，图形处理器在深度学习（DL）方面已经取得了显著优越的性能 [1]。最近，新型专用人工智能硬件平台开始涌现，有望进一步加速训练和推理。其主要针对的工作负载是人工构建的，具体而言是深度神经网络（DNNs）；近年来，这类网络展现出了巨大潜力。另一方面，高度符合生物学特征的模型，例如基于电导的神经元（如 Hodgkin-Huxley 神经元），并未像前者那样吸引人工智能芯片制造商和分析师的关注。这一点很奇怪，因为生物大脑——这些深度神经网络的灵感来源——正是使用建立在类似初等函数之上的方程进行建模的。此外，高细节模型被誉为下一波人工智能浪潮，旨在比其前代模型更具生物学启发性 [2], [3]。因此，神经科学家和人工智能研究人员都应当尝试使用这些人工智能加速器，并将其用于脑模拟；然而，目前尚不存在相关的性能研究。

在本研究中，我们评估了多种尖端人工智能芯片（Graphcore IPU [4], GroqChip [5], 支持 TensorRT 的 GPU [6]和 Google TPU v3 [7]），用于模拟一个具有高度生物学细节的脑区模型，即下橄榄核（IO）。具有生物学细节的脑模型（如下橄榄核）主要涉及加法、乘法、除法和指数运算，并以稀疏计算的形式组织。因此，这类模型与人工网络存在大量运算重叠。由此看来，新型人工智能芯片似乎非常适合这类模型。该 IO 应用代表了一项及时且具有现实意义的研究，并被构建为扩展 Hodgkin-Huxley 模型。通过调整 IO 模型的计算密度、内存需求和数值精度，它非常适合用于对不同人工智能平台进行压力测试，并突出展示架构权衡。评估基于将该应用编码为 TensorFlow 2 [8] 内核来执行；对于 GroqChip，这一内核必须编译为其 ONNX [9]等效形式。ONNX 是一种中间工具，用于在不同机器学习（ML）框架之间转换模型。

在对不同加速器进行分析时，使用完全相同的 TensorFlow 模型，以确保全面比较的公平性。该模型可以直接使用，也可以通过 Python 软件包 tf2onnx 转换为 ONNX。TensorFlow 是一种高级 API，用户只需进行少量到适度的干预，因此适合广泛的用户群体。图 1 展示了该设置的示意图。

尽管本研究中的所有加速器都支持芯片间通信，但本文将应用限制在单芯片性能比较上；多芯片部分留待未来

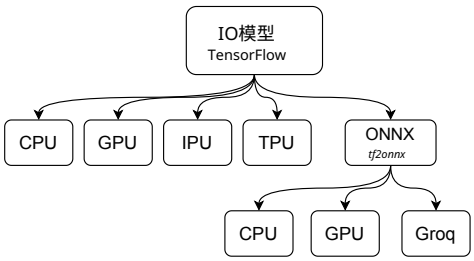

图1：本文所采用的性能分析策略概览

工作。本文的贡献如下：

- 我们深入分析了四种前沿人工智能架构，重点关注具有生物学合理性的脉冲神经网络（SNNs）。
- 我们构建了首个基于机器学习库的高效详细脑模型实现，即下橄榄核（IO）模型。
- 我们将 IO 模型部署到四种人工智能平台上，并对其性能和数值精度进行基准测试。
- 我们证明，现代机器学习库在语义上能够对科学计算中的经典问题进行建模，在保持硬件无关性的同时，带来显著的性能提升并缩短开发时间。
- 最后，这项工作首次实现了对具有真实感的、老鼠大小的 IO 模型进行实时性能模拟。

本文的组织结构如下：第二节介绍该领域的相关工作。第三节介绍作为基准测试应用的 IO 模型；第四节简要介绍待评估的四种人工智能架构，并尝试进行一些性能预测。第五节详细说明用于获得第六节所呈现结果的实验参数和平台配置，以确保实验具有可复现性。第七节对我们的发现进行总体讨论，第八节给出结论。

II. 相关工作

生物神经元模型具有不同的细节层次，涵盖从群体层面动力学、单个神经元的简化模型，到高度详细的生物物理学真实神经元 [10]。单个神经元的粗粒度模型，尤其是漏积分发放（LIF）类型模型，近年来在深度学习社区中重新受到关注（此处通常称为脉冲神经网络），被视为人工神经网络（ANN）的替代方案 [11]。相比之下，计算神经科学通常关注生物物理学准确模型，这些模型以能够深入了解潜在生物过程的方式对其进行建模。通过使用多个离散化区室对其三维结构（形态学）进行建模，可以使这些基于电导的模型更加真实。随后，通过显式计算在离散化区室内部、区室之间以及流入区室的电流，对多区室基于电导的神经元进行模拟 [12]。

由于大规模电导级脑模拟所需的计算资源十分庞大，计算神经科学很早便开始在高性能计算环境中采用通用图形处理器（GP-GPU）。基于 GPU 的大规模、具有生物学细节的脑模拟器实例包括 CoreNeuron [13], ，它支持将现有的电导级 NEURON [14] 模型移植到 GPU；以及较新的 Arbor [15], ，这是一种基于库的方法，用于实现性能可移植的大规模脑模拟。它们的成功表明，神经科学中的计算问题非常适合映射到 GP-GPU 平台，并能显著提升大规模脑模型的运行速度。然而，即使采用手工优化的 CUDA 代码 [16], ，生物规模下的 IO 应用（将在下一节详细介绍）运行速度仍比生物大脑慢一个数量级，从而阻碍了相关研究。

就基于 TensorFlow 的电导级模型实现而言，PymoNNto [17],尝试将 Brian [18] 神经模型的应用程序编程接口引入 TensorFlow。尽管其在 GTX1080 GPU 上的速度快于 Brian模拟器，但性能并非首要目标；此外，该架构将计算定义分散在整个代码库中，从而无法利用 TensorFlow 的 JIT 编译器后端进行优化。虽然这表明 TensorFlow 确实能够为神经模型提供合适的应用程序编程接口，但目前尚不存在高效的、基于机器学习库的电导级通用 GPU 脑模拟器。

简化的脉冲神经网络模型同样已经有现成的通用 GPU 漏积分发放及类似模型实现。TensorFlow 和 PyTorch 等高级机器学习库支持以硬件无关的方式实现神经动力学，从而大幅降低了构建用于通用 GPU 仿真的脉冲神经网络模拟器的开发成本。例如，Nengo DL [19] 支持使用 TensorFlow 在 GPU 上仿真 Nengo 框架中定义的现有脉冲神经网络模型。除了在 GPU 上仿真神经网络之外，机器学习库提供的基于事件的脉冲神经网络替代梯度和自动梯度计算方面的新进展，也促成了类似脉冲神经网络深度学习库 Norse [20],snnTorch [21] 和 SpikingJelly [22]几乎同时出现。BindsNET [23] 是另一个基于 PyTorch 的高效脉冲神经网络实现，重点面向强化学习。再次强调，这些项目表明，机器学习库不仅具备运行大规模脉冲神经网络模型所需的表达能力和性能，而且与硬件专用的底层代码相比，这类系统或许还能够更快开发完成。由于这些库的目标是深度学习导向目标，因此它们都没有实现多隔室的电导级神经模型。

简化的脉冲神经网络模型也推动了专用神经形态硬件的发展，以模拟这些模型。大量出版物展示了使用这些芯片进行简化脉冲神经网络模拟的优势。关于各种芯片的简要综述，读者可参阅 [24]。尽管这些芯片对于基于脉冲神经网络的深度学习模型的低功耗推理而言令人振奋，但由于其动态特性是硬连线实现的，因此缺乏模拟电导级神经元模型的能力。

对于那些具备足够语义能力、能够承载更通用高性能计算工作负载的人工智能芯片，关于简化版和电导级脉冲神经网络的研究成果都鲜有发表。就简化版脉冲神经网络仿真而言，我们只找到一篇以人工智能芯片为目标、介绍 IPU 优化版本的预印本，

snnTorch [25]。在图像分类任务中，使用稠密三层 LIF 网络进行训练时，IPU 上的吞吐量比 A100 高 3.4 倍。随着网络规模扩大，所报告的性能优势会减小；显然，在整个应用运行期间，A100 都未得到充分利用。这表明，IPU 具有用于简单 SNN 工作负载的潜力，但其他人工智能芯片或更复杂脉冲神经网络的性能特征目前尚不明确。

目前尚无研究发表，针对采用电导级模型的人工智能芯片或其他生物学真实脑模拟场景开展工作，无论是使用高级机器学习库还是硬件专用 SDK。就作者所知，这是首项在机器学习库中实现高效电导级多室神经元的工作，也是首项针对这类工作负载对多种人工智能芯片进行基准测试的工作。

III. 下橄榄核应用

下橄榄核是位于脑干中的一个固有振荡脑区，对运动控制和学习至关重要 [26]。据估计，小鼠大脑中的神经元数量约为 10^4 个神经元 [27]，人类则在 $10^6 - 10^7$ 个神经元之间 [28]。在硬件性能评估（第六节）期间将引用这些数量。在本研究中，我们将在 TensorFlow 2 中，以首次发表于 [29] 的扩展 Hodgkin-Huxley（eHH）模型，即电导级脑模型，对下橄榄核进行建模。该模型很好地体现了逼真脑模型的计算负载；同时，它也非常适合我们的基准测试目的，因为它能够捕捉复杂的神经元动力学以及快速的神经元间通信（以缝隙连接的形式），如下文所示。

本节重新列出下橄榄核神经元的主要方程，但详细内容请参阅 [29]。此外，我们根据 [30]中描述的网络对连接关系进行建模。

1）电缆模型：

$$C_m \frac{dV^{(i)}}{dt} = - \sum_{k \in \text{Channels}} I_k^{(i)} - \sum_{i \in \text{Compartments}} I_{k,j} - \sum_{i \in \text{Gap junctions}} I_{gj,k,j} - I_{\text{app}}^{(i)} \quad (1)$$

扩展 Hodgkin-Huxley 模型将包裹神经元的膜描述为电容器。因此，可以通过对流入和流出细胞的电流进行积分来计算细胞内部电压（公式 1）。此处，I_{app} 是一个可选项，用于描述实验人员施加的外部电流。

2）通道电流：通道（CaL、h、KCa、Na、Kdr、K、CaH、Na、K）允许电流流过细胞膜。它们根据内部状态变量随时间的变化产生电流。一般而言，该电流（公式 2）由电位与通道特异性反转电位之间的差值E 乘以一个或多个内部门控变量的乘积得到，其中每个变量还可以选择性地提升为整数次幂（公式 2）。门控变量遵循常微分方程（ODE），以一定速率将其带到取决于细胞电压的特定稳态（公式 3）。后面的这些方程

列表 1轴突钠通道电流

```
m_inf  = 1/(1+tf.exp(-(V轴突+30)/5.5))
h_inf  = 1/(1+tf.exp((V轴突+60)/5.8))
tau_h  = 1.5*tf.exp(-(Vaxon+40)/33)
dh_dt  = (h_inf-h)/tau_h
I_na   = g_Na*(V轴突-V_Na)*m_inf**3*h
```

列表 2稀疏间隙连接电流

```
V_diff=tf.Gather(V_dend,g_j_src)-tf.Gather(V_dend,g_j_tgt)
I_per_gj = V_diff*g_gj*(0.2 + 0.8*tf.exp(-0.01*V_diff*V_diff))
I_gap = tf.tensor_scatter_nd_add(tf.zeros_like(V),tf.reshape(g_j_tgt,(-1,1)),I_per_gj)
```

通常是电压的高斯函数或 S形函数。对于某些运行速度较快的通道，我们将 $n(t) = n^{\infty}(V)$ 设为一种数值稳定性优化。

$$I_i = \bar{g}_i \left[\prod_k n_{i,k}(t)^{m_k} \right] (V - E_i) \quad (2)$$

$$\tau_n(V) \frac{dn}{dt} = n^{\infty}(V) - n(t) \quad (3)$$

3) 分室电流：单个下橄榄核细胞由三个独立的分室组成：轴突、胞体和树突。在不同分室之间流动的电流按电阻性方式建模为：$I_{i,j} = g_{i,j}(V_j - V_i)$

4) 间隙连接电流：缝隙连接是不同下橄榄核细胞之间的直接电连接，使电流能够在它们之间流动。它们遵循通过实验确定的 Connexin-36 蛋白质动力学：

$$I_{gj} = g_{gj} \Delta V \left[0.2 + 0.8 \exp\left(-\Delta V^2 / 100 \right) \right] \quad (4)$$

其中 ΔV 表示两个已连接细胞之间的电势差。

5) 拓扑结构：真实的下橄榄核看起来像一大片折叠的薄层，主要具有局部连接。由于近似这种结构并非本文的重点，我们假设模型神经元存在于具有环绕连接的离散三维网格上（即超环面）。这应当能够表现出与更现实模型相同的非局部内存访问模式。连接根据神经元间距离 r 按照径向对称分布进行采样：

$$p(r) \propto u(r_{max} - r) \; e^{-r^2} - ( ) ( ) ( ) e^{-r_{max}^2} \; n(r) \qquad n(r)$$

( )，其中 ( ) 是 r 周围体积壳层中的神经元密度。持续进行采样，直到平均每个神经元拥有 10 个连接。

6) TensorFlow 转译：上述方程合计为每个神经元定义了 14 个常微分方程。这组常微分方程被转译为一系列 Python 中的 TensorFlow 运算符。通过使用 TensorFlow 定义模型，而非使用特定平台的 API，我们确保所有平台拥有同等的运算优化机会。此外，TensorFlow 能够自然地转译为 ONNX 模型，而 ONNX 模型是 GroqChip 唯一可用的高级 API。实现直接转译为 TensorFlow 的方法是将所有状态存储在一个大型二维数组中，并将数学表达式直接替换为其

TensorFlow 对应版本（参见清单 1）。当某些模型参数需要由用户指定时（例如， g_i 或 I_{app}），这些参数会传递给 TensorFlow 内核；之后必须重新编译，才能再次运行。

要以高性能方式将缝隙连接转换为 TensorFlow 和 ONNX 表示形式，需要将其表达为向量操作，而不是采用更传统的基于循环的方法 [16]。每个输入输出神经元平均仅有 10 个连接，因此细胞间通信是稀疏的。从 TensorFlow 的角度来看，有效操作是两次稀疏矩阵（ SM）乘法。作为计算神经科学中的一项创新性工作，我们将其建模为 tf.gather 和 tf.tensor_scatter_nd_add 操作（见列表 2）。除了能够以更具体且更节省内存的方式描述 SM 乘法之外，自 ONNX规范opset 11 起，这些函数还可直接映射到 ONNX 算子 *Gather* 和 *ScatterND* ；相比之下，SM 乘法目前无法在 ONNX 中实现。

在每个时间步中，使用前向欧拉法对常微分方程进行积分，以生成下一个状态数组，从而得到与硬件无关的时间步进函数。对于 TensorFlow后端，会构建一个可即时编译的 TensorFlow函数，在 Δt 内执行 40 个时间步，达到 $0.025ms$ 的 $1ms$ 采样精度。对于 ONNX后端，时间步进函数会被转换为 ONNX 模型，然后使用公开的 onnx-runtime 库或 Groq编译器将其编译为可执行代码。默认情况下，这并不能实现最佳性能，因此第五节-B 将讨论硬件特定优化。

IV. 目标平台

硬件平台选自 MLCommons MLPerf 训练基准 v2.0 [31] 中性能最优的人工智能加速器。其中，Intel Habana Gaudi 对我们不可用。由于 GroqChip 已可通过学术渠道获得，因此将其纳入其中。所有人工智能芯片的概览见表 I，接下来将对此进行介绍。我们还纳入了一款现代服务器级CPU，作为后续性能和数值精度比较的基线。

*1) **Nvidia GPU** [6]*: 这些是高性能计算领域中久经验证的加速器。随着 Nvidia 图形处理器引入张量核心，它们也因人工智能能力而广为人知。张量核心能够以非常高效的方式执行矩阵乘法。当前一代张量核心最高支持 TensorFloat-32 (TF32) 精度。TF32 是一种浮点格式，具有 float 32 的动态范围，但精度为 float 16。与其交互有多种方式，例如通过 cuBLAS 和 TensorRT。

*2) **GroqChip** [5]:* 这是一种具有确定性的张量流处理器（TSP），其架构类似于经过改进的脉动阵列。芯片采用传统的二维网格布局，由核心组成，每个核心都具有专用功能。这些核心中同类型核心组成的一列称为一个功能切片。数据沿水平方向传输，并在 320 条 SIMD 式通道上执行。单条指令可以控制 16 条通道，从而形成 20 个可独立运行的超级通道，彼此互不影响。功能切片由一个向量处理器（VXM）、两个矩阵执行模块（MXM）、若干开关执行模块（SXM）和存储器模块（MEM）组成。每个功能单元（核心）都接收一组指令；例如，MEM 单元可以接收一条指令，将一个向量放入某条数据流，或将数据流中的结果存入其可用的 SRAM。一旦数据加载到数据流中，就会自动沿数据流的方向“流动”，方向可以是向东或向西。执行加法时，两个输入必须与加法指令同时到达相应的 VXM 核心。这种设计使得优化工作由生成指令的软件承担。指令可以由 Groq 编译器根据 ONNX 图输入自动生成，也可以由用户通过公开的 Groq API 手动控制；Groq API 在 Groq 指令集架构之上提供了不同层次的抽象。为支持构建大规模系统，GroqChip 配备了专用的芯片间模块，可在进行片外通信时保持确定性 [32]。本研究主要使用 VXM 和 MEM 单元。存储器模块的片上 SRAM 总容量为 220 MiB。每个超级通道都包含一个由向量算术逻辑单元组成的 4×4 网格，支持 x16 单指令多数据运算。每个算术逻辑单元的输入操作数为 32 位；不过，除加法和乘法外，其他指令均采用低精度 FP32 格式执行。

*3) **Graphcore IPU** [4]*: Graphcore 智能处理单元（IPU）旨在高效执行跨大量并行线程的细粒度操作。从设计上看，IPU 提供真正的多指令多数据（MIMD）并行性。这种独特的并行处理器设计非常适合具有不规则数据访问模式的细粒度计算。每个 IPU 包含 1,472 个芯片块，每个芯片块包含 1 个核心和 624KiB 的 SRAM 存储器。单个核心只能访问其所在芯片块中的存储器。IPU 内部通信依赖一种强大的低延迟互连网络，称为 IPU 交换网络。对于 IPU 间通信，每个芯片包含 10 个所谓的 IPU 链路。IPU 计算单元称为累加矩阵乘积（AMP）单元，支持 FP32 算术运算，旨在加速矩阵乘法和卷积。在编程模型方面，IPU 采用整体同步并行（BSP）模型 [33] ，通过该模型组织其计算和数据交换操作。这种并行计算抽象由多个连续的超步组成。一个超步包括本地计算阶段；每个进程（在 IPU 中即为芯片块）彼此隔离地运行，仅在其本地存储器上执行计算，随后进入通信阶段，在此阶段每个进程都可以交换其他芯片块所需的值。这些活动最终以屏障同步阶段结束；只有当所有进程都到达屏障后，才能开始下一个超步。正因如此，IPU 可被描述为真正的 BSP 机器。

*4) **Google TPU** [7]*: TPU（第 1 版）被设计为用于推理的脉动阵列处理器，仅支持 8/16位运算。由于仅支持矩阵乘法和

表 I：实验配置中使用的所有硬件概览

设备	片上存储器	工艺节点	晶体管数量（十亿）	基础-加速频率（MHz）	热设计功耗（W）	软件
AMD 3955WX CPU *	128 GB DDR4	7 nm	19.94	3900 - 4300	280	TF 2.11.0
GroqChip TSP	230 MB 片上存储	14 纳米	26.8	900	-	Groq SDK 0.9.1 ***
NVIDIA A100 图形处理器	80 GB HBM2e	7 nm	54.2	1275 - 1410	400	TF 2.11.0
Graphcore IPU（GC200）**	900 MB 片上存储	7 nm	59.4	1330	185	TF IPU 2.6.3+gc3.0.0
Google TPUv3	32 GiB 高带宽内存	16 纳米	（估计）11	940	450	TF 2.11.0

AMD Ryzen Threadripper PRO 3955WX（16 核）｜IPU-POD16 中的单个 M2000（配备 4 个 GC200 芯片）｜***TF2ONNX 1.13.0 和 ONNX 算子集 16*

基本的非线性激活函数，它不适合训练神经网络。因此，高性能计算应用——例如本文展示的应用——也不适合在该处理器上运行。然而，随着 TPUv2 的推出，Google 将重点转向支持在其 TPU 芯片上进行训练。Google 增加了一个向量处理单元（VPU），并将矩阵乘法单元改为支持 FP16 格式（ FP32，但仅具有 7 位尾数）。从本文的结果可以推断，VPU 很可能支持更高精度，但公开资料中尚未找到相关确认。这两项重大的（微）架构变更使 TPU 能够运行更广泛的应用，包括在 TPU 上训练神经模型。所有这些功能均通过 Google XLA 编译器实现，该编译器以 TensorFlow 作为输入。本文评估的 TPUv3 在功能单元数量、更高的内存速度以及优化的芯片布局方面进行了升级，但未引入任何根本性变化。

A. 性能预测

IO 应用包含两个部分，它们分别以不同方式映射到不同类型的硬件上：i）用于更新本地神经元状态的易并行计算部分；以及 ii）用于通过缝隙连接交换膜电位的 SM 计算部分。在开始实际实验之前，我们尝试根据不同人工智能芯片架构的特性进行性能预测。

易并行： 这些计算用于更新每个神经元的状态，归根结底是逐元素的向量操作。每个流式多处理器执行一个 Warp，或配备多个张量核心的 GPU 架构，非常适合这类并行化。TPU 和 GroqChip 都基于脉动阵列架构，原生支持矩阵乘法，也支持可用于这些计算的向量运算。事实上，由于神经元更新只需要一维数据，这些架构中的矩阵乘法单元（也是这些芯片的重点）实际上没有得到充分利用。IPU 拥有大量非常小的通用核心，也应当能很好地并行处理神经元状态计算；不过，其架构面向不规则数据访问模式，而这对当前任务并非必需。因此，这些高级功能带来的额外开销，无助于提升模拟中这一易并行部分的计算性能。

通信： 如前所述，间隙连接通信采用 TensorFlow 中的聚集-散射操作（本质上是 SM 操作）。对于图形处理器或 IPU 而言，由于其架构在设计上能够处理不同的执行路径，因此此类操作的处理效果更佳。

相比之下，GroqChip 和 TPU 需要以不同方式处理这些操作：一种简单做法是通过对操作数进行独热编码来强制执行密集矩阵运算，然后利用矩阵乘法硬件。如果 GroqChip 或 TPU 碰巧采用这种策略，我们预计随着 IO网络规模的增大，性能会迅速恶化，或者内存会耗尽。

1) 中央处理器 + *TensorFlow*： 对于该平台，将通过 XLA编译器 [34]进行即时编译；它会自动利用如今中央处理器上可用的大量线程。我们预计其性能良好，且结果非常准确，因为它完整支持 FP32。由于中央处理器是传统上运行脑模型的硬件，并且能够提供准确结果，因此将作为我们的基线。加速器在运行时间方面应优于此实现，尤其是在网络规模较大时。

2) 图形处理器 +*TensorFlow*： 这里使用 XLA编译器，它通过优化计算图实现单次内核启动。其中一种优化方式是“融合”计算。此外，这种融合会将中间值保存在 GPU寄存器 [35]中。用于 CUDA 的 TensorFlow 后端会使用张量核心，但会牺牲 FP32 精度。不过，只有在明确请求矩阵乘法时才会发生这种情况，而不是作为一种优化。因此，在我们的案例中，编译器只会使用 float32 CUDA 运算。

3) IPU +TensorFlow： IPU架构并不完全适合计算中易并行的部分。对于神经元间通信部分，BSP模型更加适合，因此预计性能会更好。然而，由于拓扑结构作为未知参数提供给模型，因此不能指望IPU编译器将相邻细胞分配到相邻计算瓦片上，从而导致通信性能欠佳。可用内存应当能够轻松应对较大的问题规模。

4)TPUv3 + TensorFlow： TPU支持FP32，预计能够很好地处理我们的工作负载，尤其是在未连接的情况下。由于Google在TensorFlow支持方面投入了大量精力，预计Gather-Scatter操作将根据硬件能力得到优化。由于v3模型支持FP32，我们还预计输出中的数值结果是正确的。

5) 中央处理器/图形处理器 *+ONNX*： 预期与中央处理器/图形处理器 +TensorFlow 相同。我们预计，在中央处理器情况下，XLA编译器的性能会略微优于ONNX运行时，这仅仅是因为它能够执行全程序优化。对于图形处理器，这种影响预计会大得多，并且预计 TensorFlow 将凭借其能力占据主导地位，优于 ONNX。

融合内核将是 TensorFlow 相较于 ONNX 中单内核调用的一大优势。尤其是小型 GPU内核的调用开销将损害 ONNX-GPU-运行时的性能。TensorRT 也是 ONNX 中受支持的后端，预计其性能将优于 CUDA运行时；然而，当后端切换到 TF32数值计算时，它会降低精度。

6) GroqChip + ONNX： GroqChip 是一种全新、即将推出的改进型脉动阵列处理器。其编译器接收 ONNX图，但并不局限于逐个操作执行，因为它会一次性重新编译完整的 ONNX图。因此，它有望执行与 TPU 的 XLA编译器相同的优化。作为该架构的第一个版本，目前的编译器开发仍在探索将非标准机器学习操作映射到硬件的方法。此外，GroqChip VXM 无法使用 IEEE FP32算术执行所有操作。因此，可以预期它的性能会略优于 TPU，但代价是精度有所降低。

V. 实验设置

A. 基准测试参数

每个平台都会针对固定问题（即网络）规模进行性能基准测试，同时还会通过对范围为 $[4^3, 5^3, \ldots, 20^3]$ 的小规模种群以及范围为 $[30^3, 40^3, \ldots, 100^3]$ 的较大规模种群模拟下橄榄核网络，评估其性能可扩展性；其中，三次方是立方网络拓扑生成方法所产生的结果。这些实验聚焦于每个平台的四个不同方面，具体内容如下。

1) 非连接网络：从模型中移除通信步骤（缝隙连接）后，我们得到一种计算密集型、易并行的工作负载（尽管这在生物学上并不现实）。首先，我们测量每个 AI 平台的设置时间，包括片上缓冲区分配、提前编译（AOT）或即时编译（JIT）启用函数的定义。接下来，我们模拟生物时间为 100ms 的下橄榄核网络，并从 5 次运行中取最短的实际运行时间（包括数据传输时间）。对于即时编译（JIT）目标，如果首次运行时间超出其他运行时间的标准差范围，则以首次运行时间减去后续运行时间，作为即时编译（JIT）的编译时间，从而比较提前编译（AOT）与即时编译（JIT）目标的设置时间。

2) 连接网络：通过将缝隙连接恢复到下橄榄核网络中，我们评估通信开销。运行时间的测量方式与前述相同，但预期此处的运行时间会明显长于非连接网络。

3) 数值验证：测量性能是我们的主要重点，但这不能以牺牲功能正确性为代价。在此，我们对最多包含729个神经元的连接网络进行模拟，模拟10秒的生物学时间，并将各种结果与参考 CPU输出进行数值比较。

4) 数值压力测试：在这里，我们以一种更符合生物学实际、且令神经科学家感兴趣的方式模拟下橄榄核：我们增加神经参数的变化幅度，最重要的是，加入大量外部电流输入（模拟其他脑区），以在下橄榄核动力学中诱发动作电位（尖峰）。这些快速瞬态将对人工智能硬件的数值性能进行压力测试，尤其是非 IEEE 754 硬件（张量核心和 GroqChip）。我们在规模最小的 64神经元网络上进行这项实验，然后与中央处理器进行数值精度比较。

基准测试在一个公开可用、模块化且可扩展的框架中实现，可从 GitLab 下载：https://gitlab.com/neurocomputing-lab/InferiorOliveEMC/ioperf。主要基准测试脚本会自动发现可用硬件，运行相应的基准测试并记录结果。所使用的软件版本也列于表 I 中。

B. 硬件特定优化

虽然我们最初的目标并不是编写平台特定代码，但我们发现，默认情况下，一些人工智能平台的性能并不理想。例如，大多数平台默认会在每次内核调用时复制完整的参数数组，而对于这些基本保持不变的数据而言，这并无必要。为了公平比较不同硬件平台，我们允许对硬件特定代码进行优化，具体包括在不同执行内核之间实施操作融合，或避免不必要的设备与主机间数据传输。我们与 Graphcore 和 Groq 密切合作，针对各自的芯片实施了以下具体优化：

1) TensorFlow XLA： 运行图时，TensorFlow图执行器通常会通过相应的内核调用分别执行每项操作。XLA 提供了另一种运行 TensorFlow 模型的方式：它会将 TensorFlow图转换为一系列针对特定应用创建的内核。这些内核可以利用特定于应用的信息进行优化，例如操作融合。XLA编译器提供三个可用后端：中央处理器、图形处理器和 TPU。对于 IO 应用，我们实现了一个 TensorFlow包装函数，每次调用时最多融合 40 个时间步，以充分发挥 XLA编译器的优势。

2) ONNX： 除 GroqChip 外，所有 ONNX 实现都基于 *onnxruntime* 或 *onnxruntime-gpu*。我们启用了所有后端支持的图优化。我们使用显式 IOBindings，避免不必要的主机与设备之间的数据复制。仿真开始时，参数会被复制到设备一次。随后，状态会分配两份；每个时间步在两个缓冲区之间切换，一个作为输入状态，另一个作为输出（下一）状态。对于 TensorRT，我们保留默认设置，启用 TF32；否则，它将无法使用其 TensorCore。

3) Groq： 使用 Groq编译器 编译 ONNX图 后，二进制程序可以直接在 GroqChip 上执行。这里一种简单的方法是：针对 40 个 时间步 调用该二进制程序 40 次，并持续来回传输数据，因为 GroqChip 只有 SRAM，而 SRAM 在 编译时完全由系统管理。然而，Groq编译器 能够将输入和输出张量绑定到片上SRAM中的一个

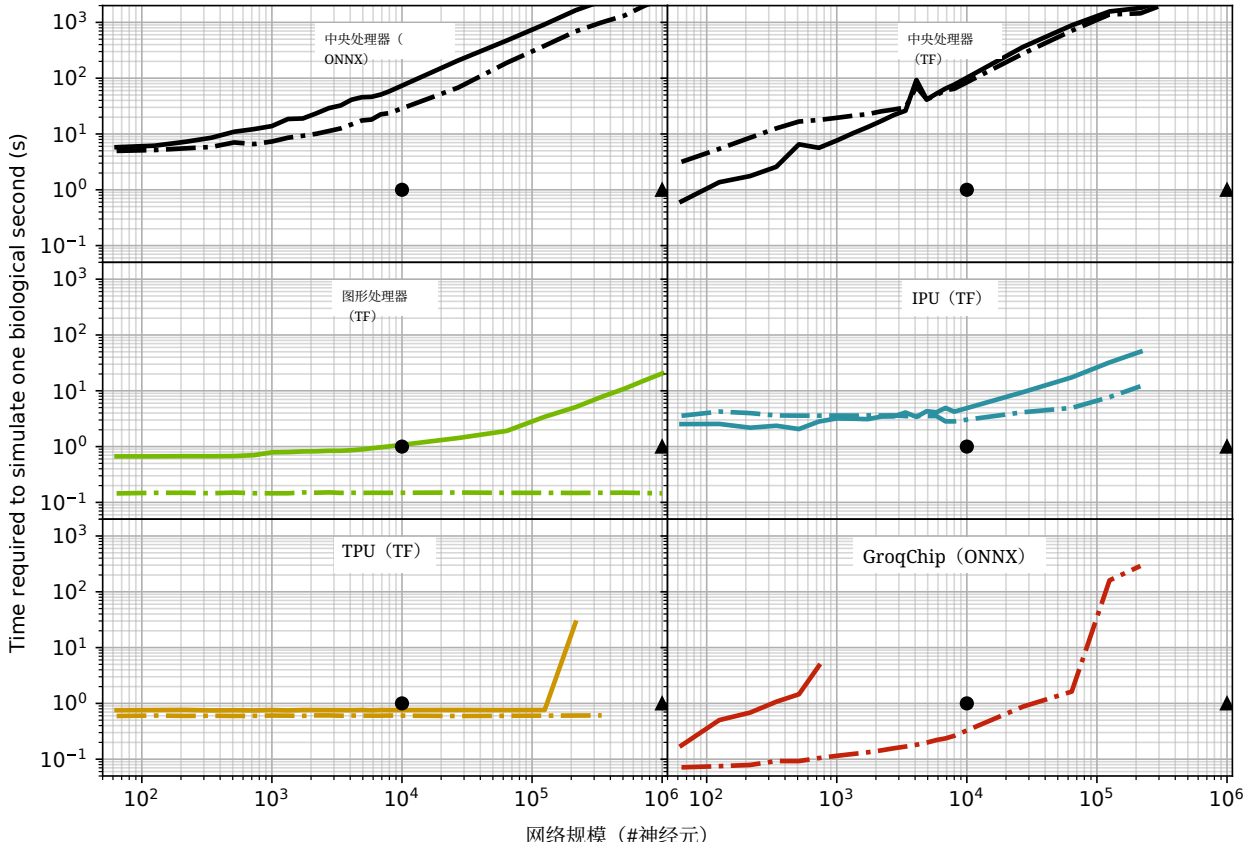

图 2：运行时性能（越低越好），中央处理器基线、图形处理器与人工智能芯片之间的比较。作为尺度参考，鼠标（•）和人类（▲）下橄榄核在所有图中均显示为实时运行。中央处理器被列出两次，以解释所观察到的 IPU 切换行为。在中央处理器上，当 XLA 优化器构建单核心的连通网络仿真时，它会构建多核心的非连通网络仿真（通过负载测试观察到），从而导致后者出现意外的缓慢仿真。同样的行为也可以在 IPU 上观察到，因为 IPU 同样使用 XLA编译器。

持久化 内存缓冲区。采用这种方法后，仍需调用二进制程序 40 次，但可以跳过 主机 与 加速器 之间持续进行的 I/O。进一步提升性能的一种方法，是将 40 个 时间步 编译成一个单独的 ONNX图，然后再使用 Groq编译器 进行转换；这样可以将 40 次调用减少为 1 次。只要编译器能够完成编译，我们就实现了所有优化。随着网络规模增大，同时处理 40 个 时间步 很快就遇到了 编译器错误。

4) Graphcore：IPU 在架构层面支持流式内存。这意味着我们可以在芯片上运行一个涵盖整个模拟过程的单一程序，并每隔 40 个时间步流式输出样本。内部未采样的 1 毫秒、40 个时间步循环使用 ipu.loops.repeat 运行，随后将记录的电压推送到一个样本大小为 200 的 IPUOutfeedQueue 中。然后，再次使用 ipu.loops.repeat 循环运行该过程，以完成所需的模拟毫秒数，并将其封装在一个 TensorFlow即时编译函数中。此外，还启用了快速数学运算优化，使用 place_ops_on_io_tiles = True 为 I/O 预留 128 个 IPU瓦片，并将程序执行限制在单个 IPU 上。

VI. 实验结果

除参考 CPU 外，为简洁起见，本文根据 TensorFlow 和 ONNX 哪者性能更佳，报告相应的结果。总体性能图如图 2 所示，下一节将对此进行详细说明。总体而言，对于 IO 应用，ONNX 端口的性能不如其 TensorFlow 对应版本。这是因为 onnx-runtime 库目前无法执行像 XLA编译器那样广泛的优化。例如，CUDA 目标会将每个计算步骤转换为一次预定义内核调用。TensorRT 后端会执行算子融合，从而生成多个串联算术运算的内核。不过，CUDA XLA 后端的性能远超 ONNX CUDA 目标，因此我们将相应结果从主要分析中删去。需要注意的是，Groq 平台仅支持对 ONNX 模型进行提前编译。

A. 编译时间

软件栈和硬件都会影响程序设置时间，如图 3 所示；图中展示的是能够适配所有人工智能芯片的最大网络（729 个细胞）。中央处理器在各方面的编译速度都是最快的，因为我们将 ONNX 操作直接转换为 CPU 优化回调。图中未包含的 TensorFlow（XLA）版本速度要慢得多

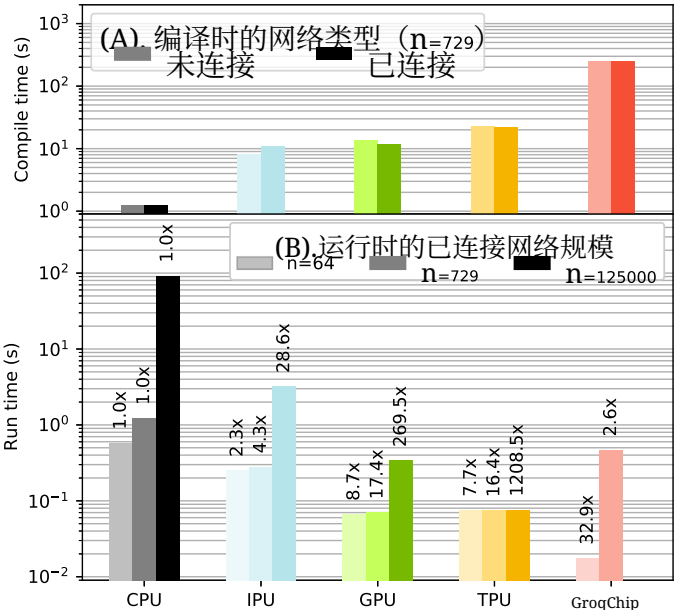

图 3：（**A**）在非连接网络和连接网络配置下，包含 729 个神经元的网络的设置（提前编译+即时编译 +内存分配）时间。即时编译时间取自 5 次性能运行中的首次运行，并计入初始设置时间（若超出一个标准差）。（**B**）在连接网络基准测试中，不同网络规模下，各种人工智能芯片相对于中央处理器基准的性能和加速比。所选规模分别为能够在 GroqChip（729）和 TPUv3（125,000）上运行的最小（64）和最大连接网络。最右侧的 GroqChip 柱缺失，表示对应模型无法完成编译。

这是由于编译器复杂度增加所致。IPU 和图形处理器的即时编译速度相近。由于显式串联了 40 个时间步，GroqChip 的提前编译器针对这一工作负载所需时间明显更长。每个程序仅包含一个时间步的 GroqChip 版本，其编译速度远快于 Graphcore 或 A100 版本，但性能会略有下降。

B. 运行时性能

1）非连接网络（易并行）：对于非连接细胞，神经动力学仅使用向量化操作来表示。正如预测的那样，这种方式非常适合图形处理器的计算范式。性能随问题规模线性扩展（水平线），表明对于所有模拟网络规模，图形处理器核心均未得到充分利用。

作为脉动阵列处理器，TPU 和 GroqChip 的架构适配性预计较差，因为芯片的大部分区域将处于闲置状态。不过，对高效向量操作的侧重仍可能带来加速。我们确实可以在图 2 中观察到这一点，尽管具体表现有所不同。与图形处理器类似，TPU 在所有问题规模下都趋于稳定，但速度慢了 4.1 倍。因此，对于 TPU 而言，内存容量并不是问题；真正限制性能的是架构设计选择导致的原始单细胞计算性能上限。相比之下，GroqChip 起初比图形处理器快 2.0 倍，但很快失去这一优势，并在 10^3 到 10^4 个单元之间开始触及其内存容量限制，随着问题规模增大，性能逐渐下降。超过 640,000 个单元的网络根本无法容纳在该芯片上。GroqView分析器证实问题在于核间内存通信，并且大多数专用核心大部分时间都未被使用。

鉴于 IPU 拥有大量核心，原本预计其表现会很好，但高度同质化的计算负载并不适合其多指令多数据设计，导致芯片利用率严重不足。就实时性能而言，只有图形处理器，其次是 GroqChip（忽略内存问题），以及勉强达到要求的 TPU 能够在 1 秒内完成。

2）连接网络（通信开销较高）：正如预测的那样，由少量缝隙连接引起的通信模式，导致图形处理器上小型网络的性能大幅下降，降幅达到 4.6 倍。对于规模更大的问题，性能下降速度不断加快；当网络包含 10^6 个细胞时，性能降低了 141 倍。人工智能芯片在此表现好得多，其中大多数芯片初始性能相比其未连接版本的降幅低于 20%。

作为例外，GroqChip 的连接网络模拟运行速度比未连接版本慢 2.5 倍；即便如此，在规模极小的连通神经网络上，其速度仍比图形处理器快 3.7 倍。不过，GroqChip（正如预期）会将 SM通信转换为密集矩阵乘法，从而充分利用其确定性执行硬件。这很快会导致矩阵乘法规模大到无法承受；当细胞数量超过 729 个时，调度器无法分配所需的指令。实际上，对于规模更大的网络，GroqChip 失去了相对于图形处理器的优势。

相比之下，TPU 的表现与未连接的情况几乎完全相同，且其性能仍不会随问题规模增大而提升。当网络规模超过 10^5, 时，*JIT*编译器 似乎会遇到性能问题。在这种情况下，我们观察到性能出现较大的随机波动：在 5 次重复运行中，运行时间有时约为 1 秒，有时则超过 400 秒。我们认为这可能源于 TPU 的内存限制，因此由于运行时间过长而不得不停止基准测试。不过，我们无法确定波动的真正原因。

对于未连接的情况，IPU 的利用率严重不足；而在小型网络中，增加通信开销实际上会提升其性能。虽然这有悖直觉，但这与我们在基于XLA的CPU后端上观察到的效果相同。在该后端上，我们发现缝隙连接会使模拟变为单核运行；由于同步开销较低，这种运行方式实际上比未连接情况下的并行多核运行更快。当网络规模达到约 10^4 个细胞时，这种情况会发生变化：间隙连接通信成为正常模拟之外的一项固定开销。到了一定规模后，这种开销会呈指数级增长，最大的模拟网络甚至无法再放入单个 IPU 中。

C. 数值验证

虽然所有人工智能芯片的性能都优于CPU基线，但同时探究不同运行结果相对于同一CPU结果在数值精度方面可能存在的下降也是明智之举。在此，我们比较了一个扩展的、持续10秒的729个细胞连接网络模拟中，以1毫秒为采样间隔获得的细胞胞体电压

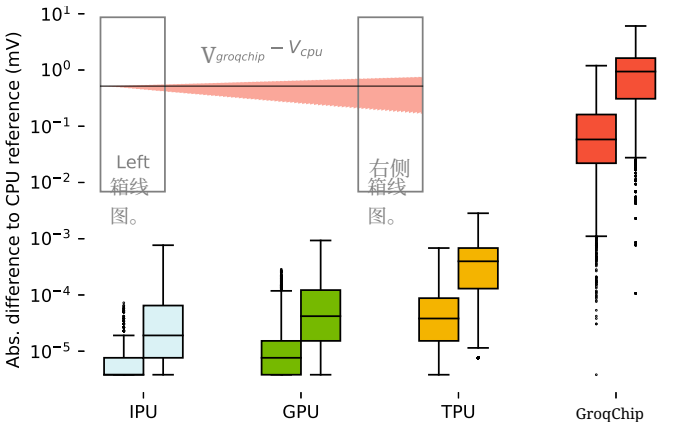

图4：数值精度验证（越低越好）。箱线图显示了相对于CPU基线的偏差，这些数据记录于数值验证模拟开始时和结束时各一个1秒时间段内，模拟总时长为10秒。偏差最大的GroqChip结果绘制在左上角，并与两个记录时间段一同显示。

（这是所有平台均支持的最大规模种群）；结果如图4的箱线图所示。

不出所料，支持 IEEE754浮点数值计算的平台（IPU、图形处理器、TPU）都能准确复现电压轨迹。即使启用了快速数学运算，IPU 仍最忠实于 CPU基线。图形处理器和 TPU 的偏差逐渐增大，但仍处于可由浮点指令重排序解释的范围内。GroqChip 虽支持 FP32数值存储，但包括指数计算在内的某些运算采用了较低精度。这表现为其相对于 CPU基线存在相当大的 mV 阶偏差，而该过程发生在 $10-100mV$ 量级的尺度上。对于振荡细胞而言，这一电压差异主要源于缓慢累积的相位差。TensorRT（未显示在此图中）默认使用 Nvidia 的 TF32，其精度被发现与 GroqChip 相近。

D. 数值压力测试

数值压力测试增大神经元变异，并加入会促使神经元产生尖峰的外部输入。这些快速瞬态无法使用 FP16精度进行模拟，但张量核心或 GroqChip 所采用的精度降低的 FP32 运算仍未经测试。我们再次将各种人工智能芯片的胞体电压轨迹偏差与 CPU基线进行比较。

同样，原生FP32支持的平台偏差最低：IPU的偏差为 $0.087mV$，GPU的偏差为 $0.135mV$，TPU与CPU基线相比的最大绝对差值为 $0.672mV$。这些中等程度的 mV量级差异，可以用微小的尖峰时间差异来解释；由于神经元尖峰幅度较大，这些差异会迅速导致明显的电压偏差。重要的是，所有模拟都能稳定运行；也就是说，不会导致这个混沌的IO模型模拟器崩溃。GroqChip模拟起初与数值验证测试中的情况相同，但一旦施加输入扰动，便会变得不稳定，测得的最大电压偏差达到 $8.51 \times 10^{36}mV$，这对于科学应用而言不可接受。

值得注意的是，误差在此处趋于稳定，不会像FP16模拟中观察到的那样发散至无穷大或 NaN 值。为恢复数值稳定性，我们尝试将GroqChip模拟的时间步长常数分别降低 Δt 10倍和100倍，但这并未使结果更接近CPU的结果范围。

VII. 讨论

正如本文所示，使用 TensorFlow 这样的机器学习库，在人工智能平台上执行高度符合生物学特性的 SNN 工作负载，会变得异常简单易用。诚然，通过各种硬件软件开发工具包（Software Development Kits）编写代码，或许能获得更好的性能；但如今可供选择的硬件种类繁多，不现实地要求计算科学家掌握所有硬件选项的底层细节。

如前所示，即时编译带来的额外优势，使手写 CUDA 实现的性能可与 XLA 编译的 TensorFlow 版本相当，同时也让用户能在新硬件发布时轻松迁移。我们预计，未来会有更多传统的高性能计算工作负载采用机器学习库实现，也就是基于张量的实现。

对于 Graphcore 和 Groq 等前景可期的新型加速器，我们认为，未来的性能提升主要将来自软件和编译器升级，因为当前的软件开发工具包大多针对机器学习工作负载进行了优化。例如，GroqChip 上的聚集-散射操作无需实现为稠密矩阵操作；内存利用率还可以进一步提高，也必须为迭代程序提供更好的支持。TPU 在架构上与 GroqChip 相似；显然，与将索引编码为独热向量相比，它执行聚集-散射操作的效率更高。

通过在 IPU 上有效使用混合精度，或在 Tensor Cores 或 GroqChip 上使用精度降低的 FP32运算，可以获得加速。对于 IPU，需要另行开展数值敏感性分析，以确定计算图中的哪些部分可以降为（随机舍入的）FP16。如前所示，Tensor Cores 和 GroqChip 当前形式下的精度损失使其无法用于脑模拟，但或许可以通过切换积分方案或采用其他数值优化来加以利用。

最后，本文未涉及多芯片拓扑。所有讨论的架构都支持专门开发的低延迟芯片间硬件以及多种通信协议。在许多方面，这种连贯通信是比加速速度本身更重大、也更迫切的挑战，而解决这一挑战将为大规模脉冲神经网络模拟（或训练）带来巨大收益。然而，要利用这些平台特定接口，需要针对相应 SDK 编写 IO 应用；通常无法依赖 TensorFlow 或 ONNX 框架。为此必须进行审慎且针对平台的编码，这部分工作留待未来开展。

VIII. 结论

在本研究中，我们构建了首个基于机器学习库、高效实现的大规模电导级脑

模型——下橄榄核（IO）。随后，我们对该模型在一台 16 核 AMD Ryzen Threadripper PRO 3955WX 中央处理器、一块 Nvidia A100 图形处理器以及不同人工智能芯片（ Graphcore IPU M2000、GroqChip 和 Google TPU v3）上的仿真性能进行了基准测试。我们发现，所有加速器相比中央处理器实现都能显著提速。对于这一特定问题，图形处理器和 TPU 似乎最适合进行仿真，其中 TPU 创下了实时下橄榄核仿真的新纪录。对于小型网络，GroqChip 的性能优于其他加速器，但大型网络无法装入片上指令存储器。更广泛地说，我们推测，现代机器学习库具备对科学计算中的经典问题进行建模的语义能力。因此，这些问题能够与机器学习驱动的新型人工智能芯片架构实现极佳的映射；除了带来显著的性能优势外，这些架构还可缩短开发时间。例如，在 TPU 上运行的下橄榄核应用版本，其性能远超手写且经过手工优化的 CUDA 实现，而开发成本仅为后者的一小部分。确切的硬件权衡将因应用而异，而硬件无关的模型描述也能显著助益于硬件选择。

致谢

如果无法使用专用硬件，本研究将不可能完成：RTX6000 由 NVIDIA 硬件资助计划捐赠，Google 为 TPU 访问提供了免费的云积分，Graphcore 则通过 Paperspace 和 Gcore 云，借助其 Academia 计划提供了 POD16 机器的访问权限。此外，我们还要感谢 Graphcore 员工协助优化 IPU 代码，并感谢 Mario Negrello 博士提供神经科学方面的见解。

参考文献

[1] A. Reuther、P. Michaleas、M. Jones、V. Gadepally、S. Samsi 和 J. Kepner，“机器学习加速器的调研与基准测试”，载于 *2019 IEEE* 高性能极限计算会议（ *HPEC*）。IEEE，2019 年，第 1–9 页。[2] N. Kasabov，“第 6 章——面向脑启发人工智能的演化型脉冲连接主义系统”，载于《神经网络与脑计算时代的人工智能》，R. Kozma、C. Alippi、Y. Choe 和 F. C. Morabito 编。Academic Press，2019 年，第 111–138 页。 [在线]。可访问：https://www.sciencedirect.com/science/article/pii/B[3] 9780128154809000062 W. Maass，“脉冲神经元网络：神经网络模型的第三代”，《神经网络》，第 10 卷，第 9 期，第 1659–1671 页，1997 年。 [在线]。可访问：https://www.sciencedirect.com/science/article/pii/S[4] 0893608097000117 Z. Jia、B. Tillman、M. Maggioni 和 D. P. Scarpazza，“通过微基准测试剖析 Graphcore IPU 架构”，2019 年。 [在线]。可访问：https://arxiv.org/abs/1912 03413 。D. Abts、J. Ross、J. Sparling、M. Wong-VanHaren、M. Baker、T. Hawkins、A. Bell、J. Thompson、T. Kahsai、G. Kimmell、J. Hwang、R. Leslie-Hurd、M. Bye、E. R. Creswick、M. Boyd、M. Venigalla、E. Laforge、J. Purdy、P. Kamath、D. Maheshwari、M. Beidler、G. Rosseel、O. Ahmad、G. Gagarin、R. Czekalski、A. Rane、S. Parmar、J. Werner、J. Sproch、A. Macias 和 B. Kurtz，“快速思考：用于加速深度学习工作负载的张量流处理器（TSP）”，载于第 *47* 届 *ACM/IEEE* 计算机体系结构国际年度研讨会论文集，ISCA ’20 系列。IEEE Press，2020 年，第 145–158 页。 [ ] 10 1109 45697 2020 00023 在线。可访问：https://doi.org/ 。 /ISCA 。 。[6] R. Krashinsky、O. Giroux、S. Jones、N. Stam 和 S. Ramaswamy，“Nvidia Ampere 架构深入解析”。 [在线]。可访问：https: // developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/

[7] N. Jouppi、D. Yoon、G. Kurian、S. Li、N. Patil、J. Laudon、C. Young 和 D. Patterson，“用于训练深度神经网络的领域专用超级计算机”，《美国计算机学会通讯》，第 63 卷，78，2020 年 06 月，第 67 页。–。[8] M. Abadi、A. Agarwal、P. Barham、E. Brevdo、Z. Chen、C. Citro、G. S. Corrado、A. Davis、J. Dean、M. Devin、S. Ghemawat、I. Goodfellow、A. Harp、G. Irving、M. Isard、Y. Jia、R. Jozefowicz、L. Kaiser、M. Kudlur、J. Levenberg、D. Man´e、R. Monga、S. Moore、D. Murray、C. Olah、M. Schuster、J. Shlens、B. Steiner、I. Sutskever、K. Talwar、P. Tucker、V. Vanhoucke、V. Vasudevan、F. Vi´egas、O. Vinyals、P. Warden、M. Wattenberg、M. Wicke、Y. Yu 和 X. Zheng，“TensorFlow：异构系统上的大规模机器学习”，2015 年，软件可从 tensorflow.org 获取。 [在线]。可访问：https://www.tensorflow.org/ [9] J. Bai、F. Lu、K. Zhang 等，“Onnx：开放神经网络交换”，https://github.com/onnx/onnx，2019 年。[10] W. Gerstner、H. Sprekeler 和 G. Deco，“神经科学中的理论与模拟”，《科学》，第 338 卷，第 6103 期，第 60–65 页，2012 年。[11] A. Tavanaei、M. Ghodrati、S. R. Kheradpisheh、T. Masquelier 和 A. Maida，“脉冲神经网络中的深度学习”，《神经网络》，第 111 卷，第 47–63 页，2019 年。[12] E. M. Izhikevich，《神经科学中的动力系统》。麻省理工学院出版社，2007 年。[13] P. Kumbhar、M. Hines、J. Fouriaux、A. Ovcharenko、J. King、F. Delalon- dre 和 F. Sch¨urmann，“CoreNeuron：神经元模拟器的优化计算引擎”，《神经信息学前沿》，第 13 卷，第 63 页，2019 年。[14] N. T. Carnevale 和 M. L. Hines，《NEURON》一书。剑桥大学出版社，2006 年。[15] N. Abi Akar、B. Cumming、V. Karakasis、A. K¨usters、W. Klijn、A. Peyser 和 S. Yates，“Arbor——面向当代高性能计算架构的形态学细节神经网络模拟库”，载于《2019 年第 27 届 Euromicro 并行、分布式与网络处理国际会议（PDP）》。IEEE，2019 年，第 274–282 页。[16] M. A. v. d. Vlag、G. Smaragdos、Z. Al-Ars 和 C. Strydis，“探索多GPU部署中的复杂脑模拟工作负载”，《ACM体系结构与代码优化汇刊》（TACO），第 16 卷，第 4 期，第 1–25 页，2019 年。[17] M. Vieth、T. M. St¨ober 和 J. Triesch，“PymoNNto：用于设计脑启发神经网络的灵活模块化工具箱”，《神经信息学前沿》，第 15 卷，第 715131 页，2021 年。[18] D. F. Goodman 和 R. Brette，“Brian模拟器”，《神经科学前沿》，第 26 页，2009 年。[19] D. Rasmussen，“Nengodl：结合深度学习与神经形态建模方法”，《神经信息学》，第 17 卷，628，2019 年，第 4 期，第 611 页。–。[20] C. Pehle 和 J. E. Pedersen，“Norse—— 用于脉冲神经网络的深度学习库”，2021 年 1 月，文档：https://norse.ai/docs/。 [ ] 10 5281 4422025 在线。可访问：https://doi.org/ 。 /zenodo。[21] J. K. Eshraghian、M. Ward、E. Neftci、X. Wang、G. Lenz、G. Dwivedi、M. Bennamoun、D. S. Jeong 和 W. D. Lu，“借鉴深度学习经验训练脉冲神经网络”，arXiv 预印本 arXiv:2109.12894，2021 年。[22] W. Fang、Y. Chen、J. Ding、D. Chen、Z. Yu、H. Zhou、T. Masque- lier、Y. Tian 及其他贡献者，“SpikingJelly”，https://github.com/fangwei123456/spikingjelly，2020 年。[23] H. Hazan、D. J. Saunders、H. Khan、D. Patel、D. T. Sanghavi、H. T. Siegelmann 和 R. Kozma，“BindsNET：Python 中面向机器学习的脉冲神经网络库”，《神经信息学前沿》，第 12 卷，第 89 页，2018 年。 [在线]。可访问：https://www.frontiersin.org/article/ 10 3389 2018 00089 。/fninf。[24] R. A. Nawrocki、R. M. Voyles 和 S. E. Shaheen，“神经形态架构与实现综述”，《IEEE电子器件汇刊》，第 63 卷，第 10 期，第 3819–3829 页，2016 年。[25] P.-S. V. Sun、A. Titterton、A. Gopiani、T. Santos、A. Basu、W. D. Lu 和 J. K. Eshraghian，“智能处理单元加速神经形态学习”，arXiv 预印本 arXiv:2211.10725，2022 年。[26] J. De Gruijl、L. Bosman、C. I. De Zeeuw 和 M. De Jeu，“下橄榄核：全面解析”，《小脑与小脑疾病手册》，第 3 卷，第 1013–58 页，2013 年。[27] Y. Yu、Y. Fu 和 C. Watson，“C57BL/6J小鼠的下橄榄核：化学构筑学研究”，《解剖学记录》，第 297 卷，第 2 期，第 289–300 页，2014 年。[28] V. Braitenberg 和 A. Sch¨uz，《皮层：神经元连接性的统计学与几何学》。施普林格科学与商业媒体，2013 年。

[29] J. R. De Gruijl、P. Bazzigaluppi、M. T. de Jeu 和 C. I. De Zeeuw，“网络环境中的攀爬纤维爆发大小与橄榄核亚阈值振荡”，*PLoS Computational Biology*，第 8 卷，第 12 期，第 e1002814 页，2012 年。[30] M. Negrello、P. Warnaar、V. Romano、C. B. Owens、S. Lindeman、E. Iavarone、J. K. Spanke、L. W. Bosman 和 C. I. De Zeeuw，“下橄榄核的准周期节律”，*PLoS Computational Biology*，第 15 卷，第 5 期，第 e1006475 页，2019 年。[31] “MLPerf Training v2.0”。 [在线]。可访问：https://mlcommons.org/en/ training-normal-20/ [32] D. Abts、G. Kimmell、A. Ling、J. Kim、M. Boyd、A. Bitar、S. Parmar、I. Ahmed、R. DiCecco、D. Han、J. Thompson、M. Bye、J. Hwang、J. Fowers、P. Lillian、A. Murthy、E. Mehtabuddin、C. Tekur、T. Sohmers、K. Kang、S. Maresh 和 J. Ross，“面向大规模机器学习的软件定义张量流式多处理器”，载于第 *49* 届年度国际计算机体系结构研讨会论文集，ISCA ’22 系列。美国纽约州纽约：美国计算机协会，2022 年，第 567–580 页。 [在线]。可访问：https://doi.org/ 10 1145 3470496 3527405 。/。L. G. Valiant，“并行计算的桥接模型”，《美国计算机学会通讯》，第 33 卷，第 8 期，第 103–111 页，1990 年 8 月。[33] [在线]。可访问：https://doi.org/ 10 1145 79173 79181 。/。“XLA：面向机器学习的优化编译器”。[34] [在线]。可访问：https://www.tensorflow.org/xla[35]“借助 XLA 突破 GPU 性能极限”。[在线]。可访问：https://blog.tensorflow.org/2018/11/ pushing-limits-of-gpu-performance-with-xla.html