技术报告：原生国产自主可控类脑脉冲大模型 SpikingBrain-瞬悉 1.0

潘昱锜[1,2,3]，冯宇鹏[1]，庄景豪[1]，丁思宇[1]，刘泽昊[1,4]，孙博涵[1]，俞雨宏[1,4]，
徐涵[1,5]，邱雪睿[1,6]，邓岸林[1]，胡安杰[1,7]，周芃[8]，
姚满[1,2,3]，吴冀彬[4]，杨建[9]，孙国梁[9]，徐波[1,2]*，李国齐[1,2,3]*

1：中国科学院自动化研究所； 2：中国科学院自动化所类脑通用智能大模型北京市重点实验室；
3：脑认知与类脑智能全国重点实验室； 4：香港理工大学； 5：北京智源人工智能研究院；
6：北京中关村学院； 7：北京航天航空大学； 8：陆兮科技； 9：沐曦集成电路有限公司

2025 年 9 月 5 日

摘要

Transformer 的训练开销随序列长度呈平方级增长, 推理显存占用随序列长度线性增加, 这造成了基于 Transformer 的大模型资源消耗巨大、长序列处理能力受限。为应对这些问题，本项目借鉴大脑结构功能与信息处理机制，开发 7B 和 76B 规模的高效类脑脉冲大模型，整个训练与推理全流程在国产算力（沐曦科技[1]曦云 C550）集群上进行。本报告涉及的核心技术点为：（1）在模型层面，基于脉冲神经元构建了具有新型线性或混合线性复杂度的基础模型架构；（2）在算法层面，构建了与现有大模型兼容的通用模型转换技术和高效训练范式，并配套开发了专用的脉冲化编码框架；（3）在工程层面，在国产 GPU 集群开发了兼容的大模型训练框架、Triton/CUDA 算子库、模型并行策略以及集群通信原语。测试评估亮点包括：（1）两款模型的长序列训练效率显著提升，能以极低的数据量实现与众多开源 Transformer 模型相媲美的通用语言建模性能（约为主流大模型的 2%）；（2）推理阶段模型的低计算/存储复杂度，结合脉冲事件驱动特性，1M 长度下的 TTFT（生成第一个 Token 所需时间）加速可达 26.5 倍，4M 长度上加速超过 100 倍，长序列处理上展现出数量级的效率和速度提升；（3）面向国产算力集群的训练框架算子加速和通信适配，能保持百卡规模训练的数周稳定运行，7B 模型训练 MFU 超过 23.4%；（4）将压缩到 1B 的类脑模型部署到 CPU 手机端推理框架上，在 64k-128k-256k 长度下较 Llama3.2 的 1B 模型解码速度分别提升 4.04×-7.52×-15.39×；（5）细粒度动态阈值脉冲化策略结合粗粒度的 MoE 方案，使得网络整体稀疏性超过 69.15%，为低功耗类脑大模型运行提供有力支撑。本次尝试为国产算力平台上的高效类脑脉冲大模型的研发扩展提供了有益探索和实践，也将启发下一代类脑芯片的设计。

1 引言

自 2017 年 Transformer 架构 [1] 提出以来，依托 GPU 集群的大规模计算能力，人工智能迈入大模型时代并取得巨大成功 [2, 3, 4, 5]。当前大模型的规模在依赖“数据-算力-算

* 通讯作者

[1] https://www.metax-tech.com

法”的 Scaling Law[6] 驱动下变得越来越大。我们将此路径称为“基于外生复杂性的通用智能模型”：通过增加网络规模、算力资源和数据量提升模型智能水平，但模型的基本计算单元—“神经元”–本身是点神经元模型：一个简单的乘加单元后接非线性函数。我们知道 Transformer 架构面临其固有缺点：训练计算开销随序列长度呈平方级增长，推理时的显存占用也随序列长度线性增加，这构成了资源消耗的主要瓶颈，导致处理超长序列的能力受限。例如，当前国内外主流大模型支持 64K 或 128K 长度的序列训练，支持 1M 以下长度推理。

人脑是目前唯一已知的通用智能系统，人脑包含约 860 亿神经元和约 1000 万亿突触数量、具有丰富的神经元种类、不同神经元又具有丰富的内部结构，但功耗仅 20W 左右。鉴此，我们相信还有另一条路径，称为“基于内生复杂性的通用智能模型”：即找到一条融合神经元丰富动力学特性构建具有生物合理性和计算高效性的神经网络新路径，其将充分利用生物神经网络在神经元和神经环路上的结构和功能特性。在该思路下，探索脑科学与 AI 基础模型架构之间的桥梁、构建新一代非 Transformer 的类脑基础模型架构，或将为中国引领下一代人工智能的发展方向、实现国产自主可控类脑大模型生态提供基础积累。

本次发布的类脑脉冲大模型探索了脉冲神经元内生复杂神经动力学 [7] 与线性注意力模型之间 [8, 9, 10, 11, 12, 13] 的机制联系，基于线性最佳逼近理论 [10] 设计了线性模型架构和基于转换的异构模型架构，通过动态阈值脉冲化解决了脉冲驱动限制下的大规模类脑模型模型性能退化问题 [14, 15, 16]，实现了国产 GPU 算力集群对类脑脉冲大模型训练和推理的全流程支持（参见图1），在高能粒子物理实验、复杂多智能体模拟、DNA 序列分析、分子动力学轨迹等超长序列科学任务建模场景中具有显著的潜在效率优势。未来团队将进一步探索神经元内生复杂动态与人工智能基础算子之间的机制联系，构建神经科学和人工智能之间的桥梁，期望通过整合生物学见解来突破现有人工智能瓶颈，进而实现低功耗、高性能、支持超长上下文窗口的类脑通用智能计算模型，为未来的类脑神经形态芯片设计 [17] 提供重要启发。

2 背景

2.1 类脑计算

类脑计算（Brain-inspired Computing）是基于神经元和神经环路的结构和功能、借鉴大脑信息处理机制，构建更通用更智能的超低功耗计算系统的技术总称。目前，在神经形态芯片上运行脉冲神经网络（Spiking Neural Network，SNN）的神经形态计算（Neuromorphic Computing，又称神经形态工程）借鉴生物神经系统的组成原理与运行机制，具有丰富的时空域神经动力学特性、多样的编码机制和事件驱动计算特性，被认为是通往更通用人工智能的新一代低功耗类脑神经网络方案之一。“神经形态工程”这一术语由加州理工学院教授 Mead 在 20 世纪 80 年代末提出，旨在使用大规模集成电路构造硅视网膜、脉冲神经元和突触等器件来模拟生物神经系统的结构和功能。经过几十年的发展，神经形态工程领域已经发展和扩展到各种应用、模型/算法和硬件等。与神经形态计算相关的研究目前已成为类脑计算领域的主流发展方向。更详细的介绍可参考本团队前期的综述论文 [18]。

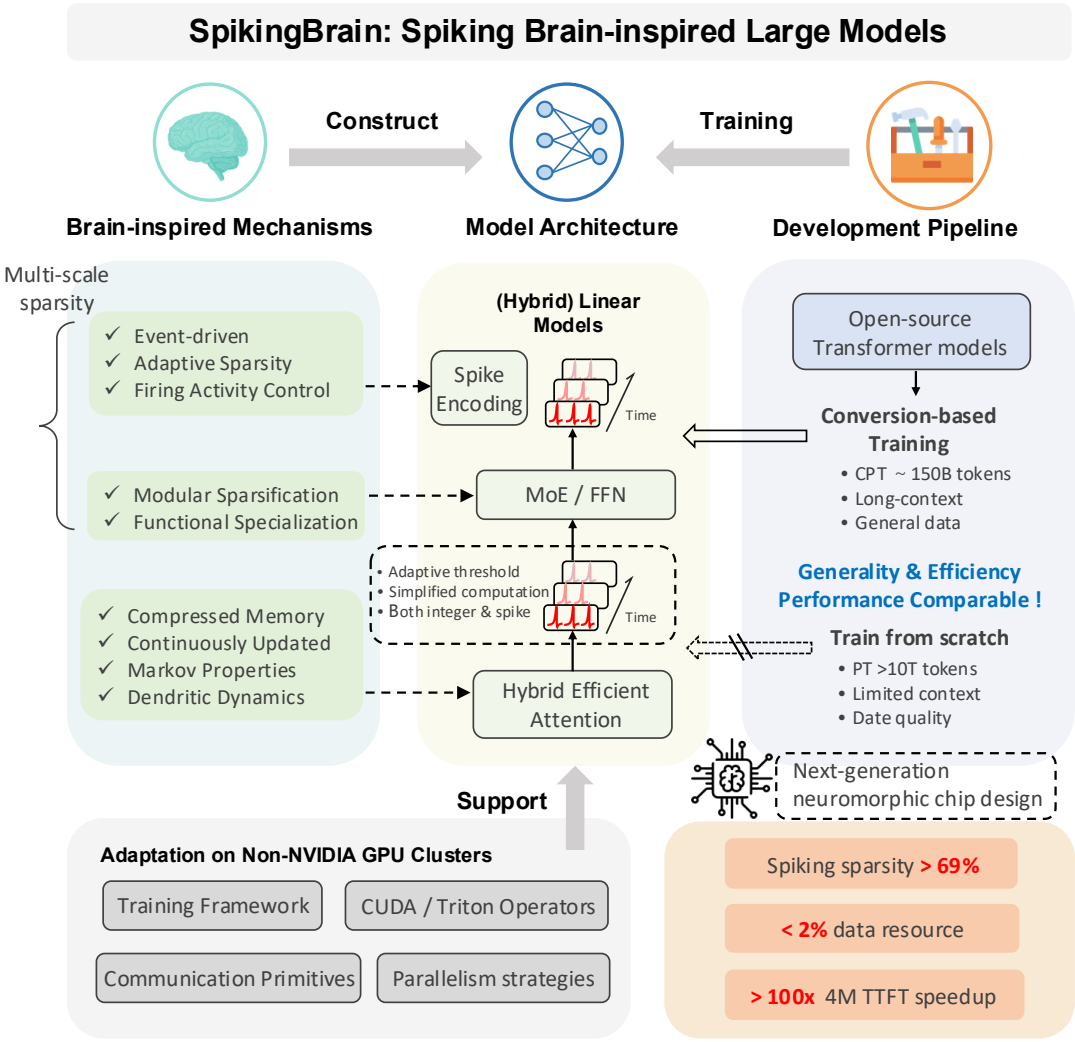

图 1: **SpikingBrain 概览**。受大脑机制启发，SpikingBrain 在架构中集成了混合高效注意力、MoE 模块和脉冲编码，并由兼容开源模型生态的通用转换流水线支持。这使得模型能够在使用不足 2% 数据的条件下持续预训练，同时实现与主流开源模型相当的性能。我们进一步针对非 NVIDIA（MetaX）集群适配了训练框架、算子、并行策略与通信原语，确保大规模训练与推理的稳定性。在 4M-token 序列场景下，SpikingBrain 在 TTFT 上实现了超过 100 倍的加速，而脉冲机制在微观层面带来了超过 69% 的稀疏度。结合宏观层面的 MoE 稀疏性，这些进展为下一代类脑芯片的设计提供了有价值的参考。

2.2 脉冲神经网络

脉冲神经网络具有二值通信、稀疏激活、事件驱动、超低功耗等特性，但也因其复杂的时域动态和离散不可导的脉冲发放过程而难以训练。进入深度学习时代后，脉冲神经网络领域面临的核心问题是，如何在 GPU 上进行大规模训练。为此，领域发展出了两条技术路线：第一，采用梯度替代法解决二值脉冲的不可微分问题，进而进行脉冲深度网络的直接训练；第二，采用基于转换的方法，将传统人工神经网络转换为脉冲神经网络。两条技术路线各有优劣，前者灵活但训练成本高，后者代价低但应用场景受限。近三年来，结合网络架构设计的快速发展，替代梯度直接训练法能够支持大规模脉冲神经网络的训练，在端侧视觉任务，例如分类、检测、分割中，已经能够在具有低功耗优势的基础上取得与传统人工神经网络相媲美的性能。本团队在前期的工作中指出了二值脉冲发放存在空间表达和时间动态上的根本性机制缺陷，提出了同步整数训练-异步脉冲推理的解决方法，该方案同时具有性能提升、功耗降低、高效训练、易于规模化、更适合神经形态芯片部署等优势 [15]。

2.3 神经元内生复杂动力学

深度神经网络的基础计算单元为神经元，其建模过程可以理解为寻找生物合理性和计算有效性的折衷。生物合理性的目标是保留生物神经系统的结构和功能，计算有效性的目标则是能够在大规模计算平台上进行高效训练计算。具体地，生物神经元是神经系统的基本构成单元，其主要功能是将一连串的突触输入转化为有意义的输出动作电位流。生物神经元的典型结构主要包括四部分：树突、突触、胞体和轴突。生物神经元的内部时空动态可以简要地描述为：树突收集来自其他神经元的突触输入信号，并将它们传递给胞体；这些输入信号更新了神经元内部的膜电位，当膜电位超过一定阈值时，胞体产生输出脉冲（即动作电位）；该脉冲沿轴突传播而不衰减，并通过轴突末梢的突触将信号传递到下游神经元。现在广泛使用的人工神经元为点神经元，被建模为简单的乘加单元后接非线性函数。在脉冲神经网络领域，最常使用的脉冲神经元为 LIF（Leaky Integrate and Fire）[19] 神经元，仍然为点神经元，但相较于人工神经元却更加复杂一些，具有复杂的时空动态特性及脉冲发放特点。本团队在前期的工作 [7] 中指出，具有复杂内生动态的脉冲神经元可以在数学上等价为若干简单脉冲神经元的组合，这说明了存在使用由复杂神经元组成的小规模网络代替由简单神经元组成的大规模网络的可能性。

2.4 线性基础模型架构

具有线性复杂度的模型，例如线性 Transformer（LinFormer）[20]、状态空间模型（SSM，Mamba）[21]、线性 RNN（LinRNN）[22, 23] 等被提出用来取代 Transformer 架构中的传统 Softmax 注意力（自注意力）。这些线性模型具有不同的起源和数学形式，通常被认为是不同类型的模型。本团队前期的工作 [10]，调研了所有的线性基础模型，认为这些具有不同起源和形式的模型实际上是同一类模型，进而从逼近视角探索了这类模型相对于 Transformer 架构的最佳设计方案。为实现这一点，首先将现有的线性复杂度模型统一为线性注意力形式，然后确定最佳线性注意力设计的三个必要条件：（1）动态记忆能力；（2）静态近似能力；（3）最小参数近似。团队发现当时的线性模型都不能满足线性注意力逼近自注意力的所有三个必要条件，导致性能不佳。基于这些最佳逼近的必要条件，可以进一步提升线性注意

力机制的性能。本团队的最新研究结果表明，线性注意力中的衰减机制与脉冲神经元中的泄露机制能建立起联系，进而为未来构建兼具生物合理性与计算有效性的类脑模型打下基础。

2.5 长序列应用前景

潜在的科学研究场景包括：高能粒子物理实验中，需要从极长时间的事件流中发现极其罕见的粒子信号，需要在超长数据流（每秒钟产生约 10^8 个数据，单数据大小约 2.5MB）中持续追踪，避免因上下文切割错过异常事件；基因组学研究中，需要在极长的 DNA 或 RNA 序列（人类基因组全长约 30 亿个碱基对）中识别复杂的遗传信息和功能信号，关键的变异模式和长程调控关系极易被分割处理所打断，从而导致信息丢失；分子动力学模拟中，催化反应或电解质溶液中的离子迁移往往需要在皮秒到微秒甚至毫秒的时间尺度上追踪原子级运动，一次性整合完整的分子动力学轨迹，能避免错过跨时间尺度的关键转变过程。

潜在的日常应用场景包括：法律/医学文档分析中，常常需处理极其冗长且结构复杂的法律条文、合同文本、判例库/病例库（百万至千万字）等，具备超长序列能力的模型能够在一次推理中完整地“阅读”整个法律/医学文档体系，避免关键条款的适用条件、跨条文的约束关系的语义丢失；软件工程中，项目的代码量往往达到数百万行，分布在上千个文件之中，具备超长序列处理能力的模型可以一次性纳入大规模代码片段，持续追踪跨模块的调用关系与数据流，避免遗漏跨文件的依赖或设计模式；在复杂多智能模拟（经济、交通、博弈）中，每个智能体都有独立的状态与交互历史，整体模拟可能持续数百万步甚至更长，超长序列模型能够在一次推理中整合完整的多智能体交互轨迹，持续追踪全局动态，从而捕捉稀有的协同行为或突发的系统性变化，避免因上下文截断而错过关键规律。

3 技术要点

以实现国产自主可控的原生非 Transformer 架构的类脑脉冲大模型高效训练与推理为目标，通过借鉴大脑结构与功能，在基础概念、模型架构、训练算法、国产集群大规模计算优化等方面进行系统性探索。在国产 GPU 集群上完成了 SpikingBrain-7B 和 SpikingBrain-76B 的全流程验证，包括持续预训练（CPT）、长序列上下文扩展训练（上下文长度最高至 128k）和监督微调（SFT）等，适配了 vLLM 框架并提出了面向国产 GPU 集群的“动态脉冲神经元”模型和“线性基础模型”架构的 Triton/CUDA 兼容算子和集群通信原语适配方案。主要技术点总结如下：

- **生物启发的稀疏脉冲事件驱动计算：**提出了一套动态阈值脉冲化信息编码方案，将模型中计算量占比 90% 以上的稠密连续值矩阵乘法，替换为支持事件驱动的脉冲化算子，以实现高性能与低能耗二者得兼：脉冲神经元仅在膜电势累积达到阈值时发放脉冲事件，脉冲到达时触发下游神经元活动，无脉冲时则处于低能耗静息状态（事件驱动）。具体地，构建了自适应阈值神经元模型，模拟生物神经元脉冲发放的核心过程，随后通过虚拟时间步策略实现“电位-脉冲”的转换，将整数脉冲计数重新展开为稀疏脉冲序列。进一步，网络层面的 MoE 架构结合神经元层面的稀疏事件驱动计算，可提供微观-宏观层面的稀疏化方案，体现按需计算的高效算力分配。
- **国产集群的大规模分布式训练：**在数百卡曦云 C550 集群上进行了 SpikingBrain-7B

和 SpikingBrain-76B 模型的全流程训练验证, 开发周期持续数周, 展现出国产 GPU 集群的稳定支撑能力。具体地，在曦云 C550 集群上完成了训练数据处理、Megatron[24] 和 Colossal-AI[25] 训练框架适配、Triton/CUDA 算子库的兼容、集群通信原语适配、大规模分布式训练的稳定性验证、长序列并行训练的通信稳定性验证和推理框架部署等工作，首次验证了类脑脉冲大模型在国产算力平台上高效开发的可行性，模型规模达 76B，在训练损失和下游指标上均展现出鲁棒性。

- **高效且通用的模型转换工具：**为方便现有大模型向脉冲大模型的切换，研发了高效的转化工具。(1) 注意力模块：基于线性最佳逼近理论，对统一注意力图的建模分析，提出了一种高效的注意力模块转换方法。该方法通过映射现有 Transformer 的注意力投影权重，将原本具有二次复杂度的标准注意力模块，转化为稀疏且局部的滑动窗口注意力（SWA）[26, 27, 28] 或低秩的线性注意力模块 [11, 12, 13]，从而显著提升长序列的训练与推理效率。在此基础上，我们进一步构建了一套通用的模型转换范式，包括持续预训练（CPT）、多阶段长度扩展训练和监督微调（SFT）。实践表明，相较于从头训练，该转换方案资源消耗极低。(2) FFN 模块：针对 76B 规模的模型，我们探索了一种基于混合专家（MoE）[29, 30] 的前馈模块上采样方法。具体地，我们通过复制并数值缩放稠密模型中的前馈网络权重，将其高效转换为稀疏 MoE 结构。该方法显著提升了模型总参数容量，同时有效控制了计算与显存开销。

与自注意力模块相比，线性注意力模块具有更贴近人脑记忆机制的建模特性，其压缩式的“记忆状态”就像人脑一样持续运作：该模块不断接受外界信息输入，并压缩、更新固定大小的状态，且每时刻只与当前的状态进行交互和信息提取。MoE 组件所具有的“模块化稀疏激活特性”以及“不同专家间的功能分化特性”也长期被认为比“全量神经元密集激活”的单个巨大 MLP 更加契合人脑的信息处理机制。进一步，网络层面的 MoE 架构结合神经元层面的稀疏事件驱动计算，可提供微观-宏观层面的稀疏化方案，体现按需计算的高效算力分配。综上，本项目通过结合新型（混合）线性架构、稀疏 MoE、轻量级通用转换训练等技术, 以及动态阈值的稀疏脉冲激活编码等一系列关键技术, 成功在数百卡规模的国产集群上验证并践行了高效的类脑脉冲大模型开发流程。

基于上述技术, 报告了两款类脑脉冲大模型: 线性大模型 SpikingBrain-7B 和基于 MoE 的混合线性大模型 76B-A12B。这两款类脑大模型在训练时具有线性或近线性复杂度，显著提升了长序列训练效率，并能以极低的数据量（约主流大模型 2% 左右预训练数据量）实现与众多开源 Transformer 模型在 MMLU 等通用任务上相媲美的性能。在推理阶段，模型具有常数（SpikingBrain-7B）或部分层常数（SpikingBrain-76B）级别的复杂度和存储开销、事件驱动的脉冲特性，在 1M 长度 TTFT（提交提示到生成第一个 Token 所需的时间）加速达到 26.5 倍，长序列处理上展现出数量级的效率和速度提升。面向国产算力集群训练框架进行 Triton 算子加速和通信适配，我们在沐曦集群上能保持百卡规模训练的数周稳定运行，同时 7B 模型训练 MFU 达到 23.4%，TGS 达到 1558 （dp8pp4-8k 序列长度）。同时，所提出的脉冲化编码方案的稀疏性超过 69%，动态脉冲编码中长序脉冲占比约 5%，结合 MOE 可提供微观-宏观层面的稀疏化方案，为低功耗的类脑大模型运行提供有力支撑。

4 模型架构

4.1 核心组件

注意力机制 注意力机制作为大语言模型的核心时序交互模块，对输入进行投影得到 QKV 向量，然后由当前时刻的 $\mathbf{q}_t$ 和过去的 $\mathbf{k}_s, \mathbf{v}_s (s \leq t)$ 交互得到注意力输出 $\mathbf{o}_t$。

标准的 softmax 自注意力 [1] 机制建模整个序列上的全局、逐 token 的信息交互：

$$\mathbf{O} = \text{softmax}(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V}; \quad \mathbf{o}_t = \frac{\sum_{s=1}^{t} \exp(\mathbf{q}_t \mathbf{k}_s^\top)\mathbf{v}_s}{\sum_{s=1}^{t} \exp(\mathbf{q}_t \mathbf{k}_s^\top)}. \tag{1}$$

其中 $\mathbf{M}$ 是因果注意力掩码，生成方式为：$\mathbf{M}_{ij} = 1$ if $i \geq j$ and $\mathbf{M}_{ij} = -\infty$ if $i < j$。对于长度为 n 的序列而言，softmax 注意力在训练时具有极高的计算强度和足够的并行度，但计算复杂度是 $O(n^2)$; 同时，其在推理时维护历史的 KVcache，存储开销为 $O(n)$，这成为长序列处理场景下的主要瓶颈。

为了解决标准 softmax attention 的二次复杂度问题，许多线性复杂度的高效注意力变体被提出。其中一种是滑动窗口注意力 (Sliding Window Attention, SWA)[26, 27, 28]，通过将注意力计算范围限制在一个固定大小为 w 的局部窗口中, 高效建模局部的精确信息交互：

$$\mathbf{O} = \text{softmax}(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M}')\mathbf{V}; \quad \mathbf{o}_t = \frac{\sum_{s=t-w+1}^{t} \exp(\mathbf{q}_t \mathbf{k}_s^\top)\mathbf{v}_s}{\sum_{s=t-w+1}^{t} \exp(\mathbf{q}_t \mathbf{k}_s^\top)}. \tag{2}$$

其中 $\mathbf{M}'$ 是窗口因果注意力掩码，生成方式为：$\mathbf{M'}_{ij} = 1$ if $i - w + 1 \leq j \leq i$, otherwise $\mathbf{M'}_{ij} = -\infty$。这使得训练计算复杂度和推理显存开销分别降至 $O(n)$ 和 $O(1)$，因为 w 是和序列长度 n 无关的自定义常数。

另一种广泛使用的高效变体是线性注意力机制 (Linear Attention)[11, 12, 13]，通过去掉标准注意力中的 softmax 函数以达到线性复杂度, 能够表达为基于状态的线性循环形式：

$$\mathbf{O} = (\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V}; \tag{3}$$

$$\mathbf{o}_t = \sum_{s=1}^{t} (\mathbf{q}_t \mathbf{k}_s^\top)\mathbf{v}_s = \mathbf{q}_t \sum_{s=1}^{t} (\mathbf{k}_s^\top \mathbf{v}_s) = \mathbf{q}_t \mathbf{S}_t, \text{ where } \mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{k}_t^\top \mathbf{v}_t. \tag{4}$$

其中 $\mathbf{M}$ 是因果注意力掩码，生成方式为：$\mathbf{M}_{ij} = 1$ if $i \geq j$ and $\mathbf{M}_{ij} = 0$ if $i < j$。Linear Attention 具有 chunk-wise 并行形式，具有足够的计算并行度来支撑训练；而在 RNN 循环形式下具有固定大小的状态 [31, 12] 来支撑推理，因此使得训练计算复杂度和推理显存开销分别降至 $O(n)$ 和 $O(1)$。

不同的注意力各有独特的优势，比如标准注意力机制具有全局的精确信息检索能力，SWA 能高效建模局部信息流,而 Linear Attention 能高效压缩长程信息。混合注意力 (Hybrid Attention) 旨在结合不同形式的注意力建模机制，兼具建模的高效性和精确度。这包括两种常见范式，一种是**层间串行混合** [32, 33, 34]，将不同注意力按层堆叠形成链式结构：

$$\mathbf{o}_t^1 = \text{attention1}(\mathbf{x}_t), \quad \mathbf{o}_t = \text{attention2}(\mathbf{o}_t^1). \tag{5}$$

另一种是**层内并行混合** [35, 36]，将不同注意力机制对同一输入并行执行，并将输出进行整合：

$$\mathbf{o}_t^1 = \text{attention1}(\mathbf{x}_t), \quad \mathbf{o}_t^2 = \text{attention2}(\mathbf{x}_t), \quad \mathbf{o}_t = w_1 \cdot \mathbf{o}_t^1 + w_2 \cdot \mathbf{o}_t^2. \tag{6}$$

这两种方式都在实践中得到了广泛使用。通过调整不同注意力的占比可以自由地平衡性能和效率。我们的 SpikingBrain-7B 模型采用了层间混合的线性注意力和滑动窗口注意力(SWA);而 SpikingBrain-76B 模型则使用了层内混合的方式，结合了线性注意力、滑动窗口注意力（SWA）和全量 Softmax 注意力三种机制。

混合专家模型 (Mixture-of-Experts, MoE) 稀疏 MoE[29, 30] 的核心思想是，通过在原有 FFN 层中引入 N 个并行的专家网络（Experts）来提升网络容量，并通过路由器（Router）$\mathbf{W}_r$ 为每个输入的 token $\mathbf{x}$ 动态挑选其中 k 个最合适的专家进行计算：

$$\mathbf{p} = \sigma(\mathbf{W}_r\mathbf{x}), \quad \mathcal{I} = \{i \mid p_i \in \text{top-}k(\mathbf{p})\}, \tag{7}$$

$$\text{MoE\_activation}(\mathbf{x}) = \sum_{i\in\mathcal{I}} p_i \times E_i(\mathbf{x}). \tag{8}$$

其中，每一个专家 $E_i(\cdot)$ 本质上都是一个前馈神经网络（Feed-Forward Network, FFN）。路由器首先对输入进行处理，得到一组专家选择概率 $\mathbf{p}$，其中 σ 通常是 softmax 或者 sigmoid 函数；然后每个 token 仅激活其中概率最高的 TopK 个专家参与计算（$\mathcal{I}$ 是激活专家的索引集合），MoE 最终输出的是这些专家输出的加权和。与传统的 Dense 层不同，MoE 在上述过程中并不会同时激活所有专家，而是通过这种稀疏激活机制，在保证计算量近似不变的前提下，显著提升模型的参数容量和表达能力，这一特性使 MoE 在推理和训练中能兼具效率与性能。同时，已有研究表明 [37, 38, 5]：在 MoE 结构中引入始终激活的共享专家（Shared Expert），以及在模型浅层保留一定数量的 Dense 层，有助于提升训练稳定性和模型性能。

在实践中，为了在已有的 Dense 模型上的基础上高效扩展为 MoE 模型，可以采用上采样（Upcycling）[39, 40] 技术，在不损失原有性能的情况下高效扩展模型规模。其核心步骤如下：（1）将 Dense 模型中前馈网络的权重参数复制到所有 MoE 专家中，从而保证上采样后的模型在初始状态下与原模型保持一致；（2）为确保上采样后的 MoE 输出与原 Dense 模型的输出尺度一致，需要对专家输出进行适当缩放。

脉冲神经元 (Spike Neuron) 是脉冲神经网络（Spiking Neural Networks, SNNs）[41] 的核心组成单元，其建模方式通常分为两种范式：简化近似模型与详细模拟模型。其中，HH (Hodgkin–Huxley) [42] 模型属于后者，它通过一组非线性微分方程组精准描述生物神经元的膜电位动态变化过程，核心在于模拟跨膜离子流的协同作用。尽管此类模型能够精准复现神经元的精细尺度生物电现象（如复杂放电模式、不应期等），但其涉及多阶高阶微分运算，因此计算复杂度较高。基于此，目前基于 HH 模型的方法主要适用于小规模神经元群体的生理学研究，无法满足大型语言模型（Large Language Models, LLMs）对效率与延迟的需求。

与之相对，LIF (Leaky Integrate-and-Fire) 等简化模型在实际应用中更为常用。LIF 神经元是纳入了时间维度的胞体动态过程一阶近似模型。LIF 通常建模为积分、泄露、发放模型，对于输入令牌 $\mathbf{x}$（额外增加了时间维度，其中 $\mathbf{x}_t$ 表示第 t 个时间步长的输入），该神经元的膜电位 $\mathbf{v}_t$ 与脉冲输出 $\mathbf{s}_t$ 可表示如下：

$$\mathbf{v}_{t+1} = \begin{cases} \lambda(1-\mathbf{s}_t)\mathbf{v}_t + \mathbf{v}_{\text{reset}}\cdot\mathbf{s}_t + \mathbf{x}_{t+1}, & \text{hard reset} \\ \lambda\mathbf{v}_t - V_{\text{th}}\cdot\mathbf{s}_t + \mathbf{x}_{t+1}, & \text{soft reset} \end{cases}; \quad \mathbf{s}_t = 1 \ \text{ if } \ \mathbf{v}_t \geq V_{\text{th}} \ \text{ else } \ 0. \tag{9}$$

其中，膜电位在胞体中积累电荷，λ 代表衰减因子（模拟离子电位的自然泄露），$\mathbf{V}_{th}$ 代表固定的发放阈值，$\mathbf{v}_{reset}$ 为复位点位（通常设置为 0）。尽管该模型简化了基于离子的作用机制，并保留了自然神经元的核心逻辑，但对于大规模模型而言，它仍存在以下局限性：

i) 时间动态特性（主要由衰减因子和复位机制引入）导致训练复杂度升高且稳定性下降；ii) 即便集成到预训练模型中，冗余复杂度依然存在，使得模型构建和原始值模拟过程变得复杂；iii) 固定阈值可能导致脉冲生成效果欠佳，造成大量神经元沉默或过度激活，不仅给优化带来挑战，还会阻碍模型在精度与能效之间实现平衡。

为了解决这些局限性，我们提出自适应阈值脉冲神经元（Adaptive-threshold Spiking Neurons）。该模型在保留生物合理性的同时，对 LIF 神经元进行简化，以提升计算效率和建模精度，具体改进如下：

- 自适应阈值：受生物神经元自适应动态特性的启发，将发放阈值 $\mathbf{V}_{th}$ 设计为与膜电位相关的动态值。这可避免神经元过度兴奋或过度静息，从统计角度使神经元维持适度激活状态。
- 简化时序计算：移除衰减因子并使用 soft-reset 机制，实现从连续值到整数脉冲计数的一步转换。优化阶段，通过合并时间维度提升稳定性，并适配 GPU 的计算效率；推理阶段，可重新展开时间维度，且借助稀疏事件驱动异步硬件实现高能效计算。

我们的建模保留了生物神经元的核心特征，并通过合理简化消除冗余计算。这不仅能充分利用生物神经系统的能效优势，若与特定异步硬件结合，还有望在预训练大型语言模型（LLM）中实现高效的工程化落地。

4.2 模型整体结构

通过轻量训练，我们可以将基座 Transformer 模型转换为任意形式的高效注意力模型（详见 5）。因此可以根据具体需求，通过调控标准注意力层的保留占比、局部注意力的窗口大小等参数，来寻求性能和效率之间的 Trade-off。作为案例展示，我们从 Qwen2.5-7B-base 基座出发，开发了两款面向不同需求的模型：7B 模型具有更极致的效率优势（纯线性模型），而 76B MoE 模型则更注重性能的平衡（混合线性模型）。我们的两款模型同时适配 Huggingface 和 vLLM 推理框架，适合单卡和多卡部署的情形。

SpikingBrain-7B 模型 7B 模型具有纯线性复杂度，由线性注意力层和固定大小 4K 的滑动窗口注意力层（SWA）按照 1:1 的比例层间交错堆叠而成；FFN 部分则使用和 Base 模型相同的 SwiGLU 模块。在该模型中，不同的高效注意力模块协同配合，SWA 负责精确局部建模而线性注意力负责长程信息压缩。该模型的特点是，在训练时具有完全线性的计算复杂度，而推理时具有不随序列长度增长的、恒定大小的显存占用，使得其能够在处理长序列时具有显著的能效优势。在实现过程中，我们实例化线性注意力为门控线性注意力模块（Gated Linear Attention Module） [12]，引入低秩门控 $\mathbf{g}_t$ 来增强模型的表达和循环建模能力：

$$\mathbf{S}_t = \mathrm{diag}(\mathbf{g}_t) \odot \mathbf{S}_{t-1} + \mathbf{k}_t^\top \mathbf{v}_t, \tag{10}$$

$$\mathbf{o}_t = \mathbf{q}_t \mathbf{S}_t. \tag{11}$$

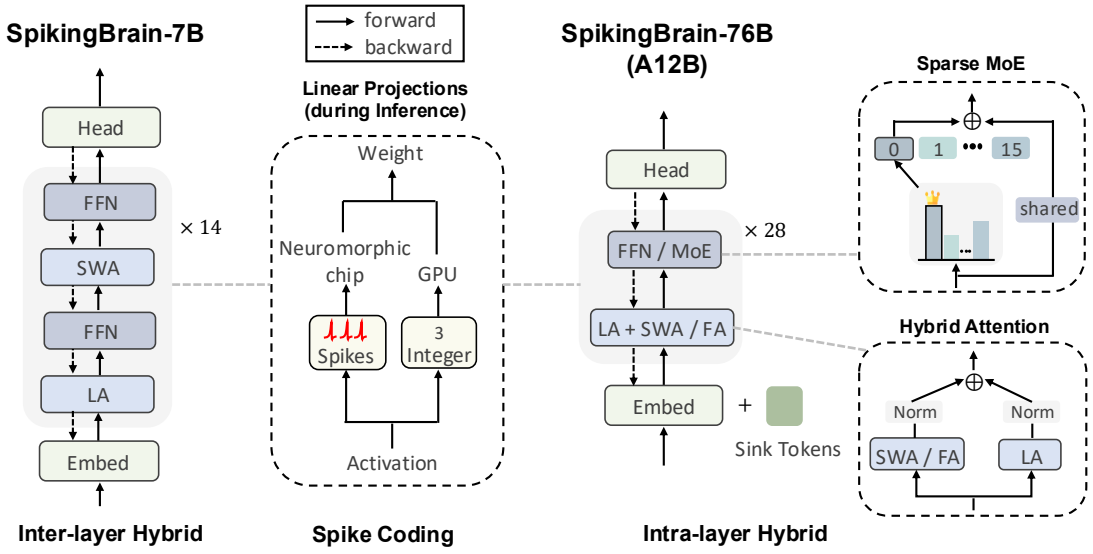

图 2: **SpikingBrain 整体模型架构**。其中 FA 表示全量 Softmax 注意力；SWA 代表滑动窗口注意力；LA 代表线性注意力。（左）SpikingBrain-7B 为层间混合的纯线性模型。（中）脉冲编码将激活值转换为整数计数用于 GPU 执行，或转换为脉冲序列用于事件驱动的神经形态硬件。（右） SpikingBrain-76B 为层内混合的混合线性 MoE 模型，配置包含 128 个 sink token、16 个路由专家以及 1 个共享专家。对于线性层，在第 $[1, 2, 3, 5, 7, 9, 11]$ 层布置了 7 个稠密 FFN，其余层均实现为 MoE 层。对于注意力模块在第 $[7, 14, 21, 28]$ 层采用“LA + FA”组合，在其他层均采用“LA + SWA”组合。

SpikingBrain-76B-A12B 模型 76B 模型由线性注意力层和固定大小 4K 的滑动窗口注意力层（SWA）进行 1:1 层内并行混合，同时按照 1:6 的比例在层间配备标准注意力层（Full Attention）。在并行混合时，两个注意力分支都会经过 RMSNorm，保证数值在同一量级，避免在训练初期出现不稳定。此外，我们额外使用 128 个可训练的 sink tokens[43, 35]，缓解 softmax 注意力的 attention sink 现象并能增强 SWA 在局部注意力建模的灵活性。具体地，我们在输入 embedding 前拼接 128 个可学习的嵌入向量，在每层中所有 tokens 均能关注该向量，同时 sink tokens 之间也能互相关注而没有因果掩码。我们基于 Flash Attention[44] 的接口函数实现具有上述功能的 kernel。我们同样使用门控线性注意力模块（Gated Linear Attention Module）构建模型，并进一步绑定 Key 向量和门控 [10, 45]，也即 $\mathbf{g}_t = 1 - \mathbf{k}_t$，而不引入新的门控投影参数。对于 FFN 部分则采用稀疏混合专家架构（MoE）；每个 MoE 层配有 16 个路由专家（激活 top-1）和 1 个共享专家，稀疏分配计算来保证每个 token 只激活约 15% 的模型参数。为稳定训练并控制参数增长，我们在一共 28 层中保留 7 层的 Dense FFN（第 1, 2, 3, 5, 7, 9, 11 层）。

与类脑机制的联系 我们的架构选择与生物大脑中观察到的机理紧密契合。(1) 线性注意力模块展现出与人类记忆类似的建模特性，它依赖于压缩和持续更新的“记忆状态”[46]。在每个时间步，它只从当前记忆中提取信息，呈现出类马尔可夫性质的行为。从生物学角度看，其状态化的时间递归可以被视为具有多分支形态的树突动力学的一种简化抽象。(2) MoE（专家混合）组件体现了模块化稀疏激活和功能分化的原理，这与神经回路中分布式和专业

化的信息处理方式非常相似。[47](3) 我们的脉冲编码方案则借鉴了生物系统中事件驱动和自适应稀疏的神经元激活机制 [41]。通过将网络层面的稀疏性（MoE）与神经元层面的脉冲稀疏性相结合，我们的方法实现了按需分配的计算，并提供了一种强大的双尺度效率策略。总而言之，这些研究结果为设计既高效又具有生物学合理性的大模型架构提供了一条有前景的途径。

5 训练范式

5.1 通用性：注意力图对应

在这里，我们对不同注意力图（attention maps）之间的关系进行了简要分析，旨在说明跨不同注意力类型进行参数迁移的可行性。首先，预训练 Tranformers 所采用的 Softmax 注意力图为：

$$\mathbf{A} = \text{softmax}(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M}) \in \mathcal{R}^{n\times n}. \tag{12}$$

以此为基础，SWA 是对该注意力图的基于强局部先验的稀疏化 [26, 27, 48]：

$$\mathbf{A} = \text{softmax}(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M}'). \tag{13}$$

而 Linear Attention 是对该注意力图的低秩近似，这里 $\mathbf{A}$ 的秩最大为 d（qk 向量的维度 [49, 20]）：

$$\mathbf{A} = \mathbf{Q}\mathbf{K}^\top \odot \mathbf{M}. \tag{14}$$

因此，基于 Attention map 的对应关系，我们可以直接以预训练 transformer 检查点 [50] 的 QKV 投影参数作为初始点，利用已经学到的 QK 相似度信息，通过少量数据的训练，使得 attention 迁移到局部、低秩的 special cases。由于 SWA 保留了注意力图的局部建模精度，而 Linear Attention 保持了注意力图的全局交互特性，所以使用 hybrid 范式将两者结合可以得到对原始注意力图更好的近似 [51, 49, 52]，能使得转换过程更加平滑从而促进 loss 收敛。得益于注意力图对应的通用性，我们可以构建任何形式的稀疏、局部、线性、混合的高效注意力图，并从任意的 softmax 注意力图进行迁移。

为了使得转换训练的成功收敛，我们遵循若干重要的实践上的选择要点：

- **对 linear attention 的 qk 施加非负激活** [50, 53]，比如 ReLU 或者 Sigmoid 函数等。因为 softmax attention map 具有非负性质，转换后的 attention 也需要具有此性质来平滑迁移。归一化过程将由线性注意力输出后的 Normalization 完成。
- **保持转换阶段重新初始化的参数是低秩的**，比如 RMSNorm 和 Gate 等。由于转换训练的过程学习率一般比 PT 更低，数据也更少，所以要学习大量随机初始化参数的难度较大。并且我们希望已有模型的权重来主导优化过程。因此，我们选择 reuse 模型的所有注意力和 FFN 部分的投影权重，保持转换中新引入的参数尽可能少。
- **将长序列扩展放到转换阶段完成**，可以在保持训练效率的同时有效扩大上下文长度。因为高效注意力机制具有亚二次复杂度，因此一个高效的方案是控制预训练二次注意力模型的训练长度和资源开销，然后将长序列扩展和 CPT 转换融合进行。

- **确保性能的转换过程下的充分训练**。这要求使用学习率的 warmup，并且采用跨架构（cross-architecture）的蒸馏 [54, 55] 或者全参数训练 [56]。为了实践的简便性，并考虑到沐曦 GPU 的显存容量，我们选择进行全参数训练，不冻结 backbone 参数。

5.2 高效性：基于转换的训练方案

转换流程 转换训练的多阶段流程包括: 持续预训练（CPT）加上长序列扩展（Long- context Extension）、监督微调（SFT）。作为展示，我们训练使用的所有数据均采样自开源高质量数据集，在实践过程中也可以选择混入专用领域数据进行垂类训练。

持续预训练过程包含三个阶段，在完成模型转换的同时逐步扩展上下文窗口长度。第一阶段使用序列长度为 8k、总规模为 100B tokens 的数据进行训练，旨在将注意力图重新调整至局部/低秩的 special case，并使模型损失达到收敛。随后，第二和第三阶段分别逐步将序列长度扩展至 32k 和 128k，并分别使用 20B 至 30B tokens 的数据进行训练。整个转换流程共使用约 150B tokens 的训练数据，以从头训练所需约 10T tokens 的数据量为参照，持续预训练过程仅使用了约 2% 的数据量，在训练资源、预算有限的场景下，高效地完成了模型转换。三个阶段的训练数据均使用 Matrix[57] 数据集，其中长序列数据通过最基础的分 domain 的 packing 策略产生。Rope base 维持基座模型的 1M 大小。

SFT 分三阶段训练，每一个阶段使用不同领域的数据，分阶段逐步提升模型在通用知识、对话、推理方面的能力。第一阶段，主要侧重于增强模型的基础语言理解和各领域的专业知识。该阶段使用 Infinity Instruct_fond [58] 作为训练数据集，其中包含科学知识、代码解析以及数学问题求解各类基础知识。训练数据包含 50w samples , 在 8k 序列长度训练。第二阶段，训练专注于对话能力与指令遵循专项提升。训练数据 (Infinity Instruct_chat) [58] 包含多轮对话，任务导向和知识问答等多种对话类型。第二阶段训练数据量与训练序列长度与第一阶段保持一致。第三阶段提升模型在简单推理任务上的能力，这一阶段采用通过 DeepSeek-R1 [59] 蒸馏得到的高质量推理数据集 [60, 61]，其中包含大量带有详细思维链标注的推理过程数据，涵盖数学证明、逻辑推理、案例分析等多种需要多步推理的问题类型。为了确保在不同语言环境下的推理能力，该阶段数据中英文比例控制为 1:1，在序列长度 8k 上共 15w samples 进行训练。

MoE 上采样 在实践中，为了在已有的 Dense 模型上的基础上高效扩展为 MoE 模型，可以采用上采样（Upcycling）[39] 技术，在复用原有参数知识的的同时高效扩展模型规模。上采样时，首先会把基础 Dense 模型的 FFN 完成复制成 N 个专家，并使用随机化的 Router，按照所挑选出的 k 个 FFN 专家的概率加权输出，保证模型的行为与原始 Dense 模型等价:

$$E_1(\mathbf{x}) := E_2(\mathbf{x}) := \cdots := E_N(\mathbf{x}) := \mathrm{FFN}(\mathbf{x}), \quad \text{at initialization.} \tag{15}$$

$$\mathbf{p} = \sigma(\mathbf{W}_r \mathbf{x}), \quad \mathcal{I} = \{i \mid p_i \in \text{top-}k(\mathbf{p})\}, \tag{16}$$

$$\text{MoE\_activation}(\mathbf{x}) = \sum_{i \in \mathcal{I}} p_i \times E_i(\mathbf{x}). \tag{17}$$

随着训练进行，路由的随机性、数据本身的噪声等会逐渐打破专家间的对称性，产生差异化梯度，推动权重逐步专门化。除了 N 个路由专家，我们的 76B 模型中还引入了 S 个对所有 token 始终激活的共享专家（$S = 1$），来稳定转换训练。

此外，上采样时如果直接复制并激活多个专家，会放大输出 scale，所以需要通过权重缩放 [40] 来保持上采样前后输出分布的一致性，从而保证模型的稳定训练。上采样后的 MoE 和原始 Dense FFN 模块的输出数值具有如下关系：

$$\text{MoE\_activation} = S \times \text{dense\_activation} + \frac{k}{N}\text{dense\_activation} \tag{18}$$

$$= (S + \frac{k}{N})\text{dense\_activation}. \tag{19}$$

因此，在我们的实现中，缩放系数的计算如下：

$$\text{scaling\_factor} = \sqrt[3]{\frac{1}{S + \frac{k}{N}}}. \tag{20}$$

我们使用该系数在初始化 MoE 权重时对共享和路由专家同时进行缩放：$E_i(\mathbf{x}) := \text{scaling\_factor} \times \text{FFN}(\mathbf{x})$.

5.3 模型的脉冲化编码

5.3.1 脉冲化编码方案与自适应阈值设计

受到生物计算机制（事件驱动和稀疏激活）的启发，同时出于训练精度和脉冲化高效性的考量，我们提出了一种特殊的脉冲化策略，来将激活值编码为等效的整数值或脉冲序列。该方法可以在训练期间或训练后进行，将大模型的激活值转换为脉冲。同时考虑到进一步提高能效，我们结合脉冲化过程，对权重和 KV Cache 进行了 INT8 量化。该方法与 SpikingBrain 轻量级的转换 Pipeline 相结合，无需进行完整的全量微调（Fine-Tuning, FT）；仅需少量校准数据即可优化量化参数。对于 SpikingBrain-7B，整个优化过程仅需单卡 15GB 显存耗时 1.5 小时，保持精度的同时大幅降低脉冲化落地成本。此外，脉冲神经元凭借其事件驱动和稀疏加法特性，为构建低能耗的大规模模型提供了可行路径。

我们的激活值的脉冲化采用解耦的两步策略，分为优化阶段的基于自适应阈值的整数脉冲发放，以及推理阶段在异步硬件上进行的脉冲化编码。这种方法使得基于整数的表达形式能够在 GPU 上支持计算高效的优化过程，而展开后的脉冲表达形式在与专用硬件结合时，则能提供事件驱动的、高能效的推理。

第一步：自适应阈值脉冲化——单步生成整数脉冲计数，维持适当发放活性 该阶段通过简化的自适应阈值神经元建模获得整数脉冲计数，核心是通过自适应阈值设计，让神经元在统计意义上始终保持“不过度兴奋、不过度静息”的适中活性，避免固定阈值导致的脉冲冗余或信息丢失。特别地，我们定义：

$$\mathbf{x} = \sum_{t=1}^{T} \mathbf{v}_t; \quad \mathbf{s}_{INT} = \sum_{t=1}^{T} \mathbf{s}_t = \text{round}(\frac{x}{V_{th}(x)}); \quad V_{th}(x) = \frac{1}{k} mean(abs(x)). \tag{21}$$

其中，$\mathbf{x}$ 代表大模型投影层输入的连续浮点激活值（等价于传统 SNN 中多时间步输入 $\mathbf{x}_t$ 的累积和），$V_{th}(\mathbf{x})$ 为与膜电位相关的自适应阈值，k 为调节神经元发放率的超参数，$\mathbf{s}_{INT}$ 为合并生成的整数脉冲计数。

我们采用简化的自适应阈值脉冲神经元，将连续的激活值转换为整数形式的脉冲计数。通过移除衰减因子（即，使用 IF 神经元模型）并利用软重置机制，提出的建模在推理阶段表达为：

$$\begin{aligned} \mathbf{v}_{t+1} &= \mathbf{v}_t - V_{th} \cdot \mathbf{s}_t + \mathbf{x}_{t+1}, \\ \mathbf{s}_t &= 1 \text{ if } \mathbf{v}_t \geq V_{th}(\mathbf{x}) \text{ else } 0. \end{aligned} \tag{22}$$

在优化阶段可以合并时间步表达为：

$$\mathbf{v}_T = \mathbf{x}, \quad \mathbf{s}_{INT} = \text{round}\left(\frac{\mathbf{v}_T}{V_{th}(\mathbf{x})}\right). \tag{23}$$

这种单步整数表达形式（formulation）使得在 GPU 上的优化过程计算高效且稳定，同时自适应阈值维持了适当的发放率。因此，我们的脉冲方案保留了生物神经元的基本特性，但又足够简化，从而对于大规模模型是可行的。

我们之所以在 SpikingBrain 架构中选择自适应阈值神经元，而非更常见的 LIF 神经元，主要动机在于前者能更好地调控神经元的发放活动。具体地，自适应阈值及超参数 k 对神经元发放活动的影响可归纳为以下几点：

- 从阈值动态性来看：$V_{th}(\mathbf{x})$ 由膜电位的绝对值均值决定——当输入膜电位较强时，$V_{th}(\mathbf{x})$ 同步升高，避免神经元因强输入过度发放脉冲（控制稀疏度，减少冗余计算）；当输入膜电位较弱时，$V_{th}(\mathbf{x})$ 随之降低，确保神经元仍能发放少量脉冲以保留关键信息（避免过度静息导致的精度损失）。在统计学意义上，膜电位通常呈现为带有离群值的长尾高斯分布，在这个前提下，绝对值均值近似等价于膜电位的 0.8 倍标准差，具有稳定的统计学意义，因此能够有效调节脉冲发放水平。

- 从超参数 k 的调节作用来看：k 直接控制阈值相对大小，进而影响神经元发放脉冲数的分布范围，当 k 取值较大时，$V_{th}(\mathbf{x})$ 降低，脉冲计数 $\mathbf{s}_{INT}$ 的数值区间更广（发放变多），适合对精度要求高、允许适度计算量的场景；当 k 取值较小时，$V_{th}(\mathbf{x})$ 升高，脉冲计数 $\mathbf{s}_{INT}$ 的数值区间变窄（发放变少），脉冲编码后更稀疏，适合边缘端低功耗场景。通过调整 k，可在精度与计算能效间灵活平衡。

- 对于离群值：在大模型中，激活值常常出现数量极少但赋值较大的离群值，这些值往往能够显著影响模型精度。自适应阈值由于其统计学意义而对离群值不敏感，神经元的发放因此对绝大多数值稳定地保持在适当水平，而对极少数离群值会生成远大于常规值的脉冲计数，这种特性类似于生物神经元对高强度输入的逆发响应，能确保离群值携带的信息不被过度损耗，同时在合适的异步硬件上，少数的离群值并不影响总体计算的高能效。

第二步：脉冲编码——虚拟时间步展开稀疏序列 在推理执行计算的阶段，需将第一步生成的整数脉冲计数 $\mathbf{s}_{INT}$ 重新展开为时间维度的稀疏脉冲序列，以适配事件驱动计算。其过程表达为：

$$\mathbf{s}_{INT} = \sum_{t=1}^{T} \mathbf{s}_t, \quad \mathbf{y} = V_{th}(\mathbf{x}) \cdot \sum_{t=1}^{T} (\mathbf{W}\mathbf{s}_t). \tag{24}$$

其中，$\mathbf{W}$ 是权重矩阵，$\mathbf{y}$ 是线性投影层的输出结果，由于 $\mathbf{s}_t$ 均为 0、1 或-1，因此矩阵运算的乘加操作被替换为了事件驱动的累加操作，实现计算能效的提升。

对 $\mathbf{s}_{INT}$ 重新展开为 $\mathbf{s}_t$ 的过程，我们设计了三种编码方式应对不同场景需求，核心目标是在合适的硬件支持下，将连续值矩阵乘法替换为稀疏事件驱动的加法运算，同时在不损失表达能力的前提下优化脉冲发放率与时间步数，从而实现更低的能耗与更高的计算稀疏性：

- **{0,1} 脉冲编码：**这是一种最基础的事件驱动脉冲编码方式。每次发放单位脉冲（值为 1）代表神经元状态的一次激活，脉冲计数通过在连续时间步内累积实现。该编码机制直观简洁，计算开销低，适用于极低脉冲计数场景，可有效降低系统复杂度。然而，在表示较大计数值时，该编码往往需要更多的时间步来完成计数表达；同时，由于缺乏对脉冲发放的定向优化，整体脉冲发放率偏高，进一步限制了系统的能效表现。

- **{-1,0,1} 脉冲编码**：为提升神经元在事件驱动计算中的表达能力与稀疏性，我们引入了三值脉冲编码（Ternary Spike Coding），其核心在于在传统 {0,1} 编码基础上增加抑制性脉冲 (-1)，形成 {-1, 0, 1} 的三种发放状态。相比只能表示正值的二值编码，三值脉冲编码具备双向表达能力，可通过“1”表示兴奋性脉冲，“-1”表示抑制性脉冲，而“0”表示静默状态，从而更贴近生物神经系统中“兴奋–抑制”调控机制。如图 3 (b) 所示，三值脉冲编码不仅提供 ±1 的对称表达能力，更通过对称量化策略重构了膜电位到脉冲计数的映射关系。相比传统二值编码将计数映射为 [0,1,2,⋯]，该策略将激活映射为 [-k, ..., 0, ..., +k] 的中心对称分布，使得低幅度计数（如 ±1）在统计上占据更大概率质量，有效吸收了原本分布尾部的高频大计数值。这种机制在不牺牲表达能力的前提下，时间步长度减半和总脉冲发放率显著减少 2 倍以上，提升了计算稀疏性与能效表现。

- **二进制脉冲编码**：二进制脉冲编码（Bitwise Spike Coding）可被视为一种事件序列编码方式，通过将整数计数值按位展开为跨时间步发放的脉冲事件，每个时间步仅对应计数值的一个二进制位，从而显著压缩时间维度开销。如图 3 (c) 所示，该机制支持三种实现形式，以适配不同符号表示与精度需求： 纯二进制编码适用于正整数，通过逐位发放实现极高的时序压缩能力，尤其适用于高计数场景，例如，当计数值为 256 时，传统的 {0,1} 编码需连续发放 256 个时间步，三值编码即便将时间步减半也仍需 128 步，而 8-bit 二进制脉冲编码仅需 8 个时间步即可完成表达； 双向二进制编码在纯二进制基础上引入 ±1 表征每位值，负数部分以 -1 替代 1，在保持表达等价性的前提下，相较于纯二进制编码时间步发放率显著减少一半同时时间步优化一步（计数值为 256 时仅需 7 个时间步）； 补码二进制编码则将符号信息融合至最高位，在保持二值逻辑简洁性和生物合理性的同时支持正负表示。相较于三值编码在高 bit 情况下仍需大量时间步展开，位编码方案通过显著压缩时间步长度，使得脉冲发放总数相较三值编码减少 8 倍（以 8-bit 计数为例），在保持表达精度的前提下，有效降低了整体脉冲通信开销与计算负载，尤其适用于高精度、低功耗、对时序资源敏感的神经形态计算任务。

5.3.2 硬件适配与落地潜力

当前我们的脉冲化方案能够在 GPU 兼容运行——通过将“时间维度合并为单步”的方式，规避了 GPU 同步架构对事件驱动计算的适配瓶颈，可直接在通用 GPU 硬件上完成仿

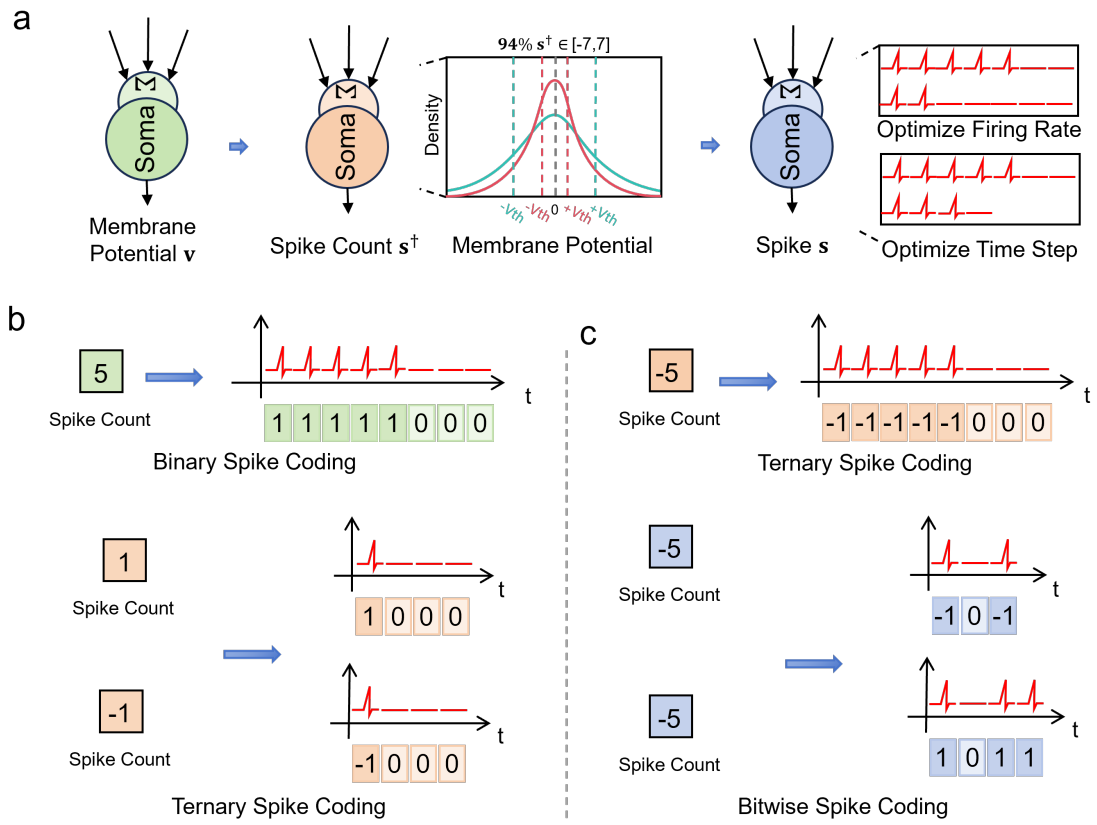

图 3: **三种脉冲编码方案示意图。** (a) 自适应阈值通过膜电位映射到脉冲计数，并在虚拟时间步展开为稀疏脉冲序列，实现从连续值激活到离散脉冲的转换与优化。(b) 三值脉冲 vs. 二值脉冲：二值以 0,1 表示“发/不发”，三值以 −1,0,1 同时编码兴奋与抑制。相较二值，三值可将时间步和发放率均减半。(c) 二进制按位脉冲 vs. 三值脉冲：将计数按二进制位跨时间步展开，实现时序压缩。在高计数场景下，所需时间步远少于三值，效率显著提升。

真推理验证。但需明确的是：GPU 的同步计算架构无法充分发挥脉冲信号“事件驱动、稀疏异步”的核心优势——现有 GPU 仍需按固定的高频时钟周期处理数据，无法像生物神经系统那样“无脉冲时待机、有脉冲时触发计算”。因此，要完全释放本方案的低能耗潜力，需适配专门的异步硬件架构（如基于异步电路设计的类脑芯片、脉冲处理器）处理矩阵运算：这类硬件可响应稀疏脉冲事件，无需高频时钟同步，无脉冲时电路处于静息低功耗状态，有脉冲时执行加法运算，最大化能效优势，为低功耗类脑大模型的边缘端部署（如工业控制、移动设备）提供可能的落地路径。通过该设计，我们为下一代高能效类脑计算硬件的研发提供了参考技术路径，推动大模型从“算力依赖型”向“能效优化型”升级。

6 MetaX 国产集群训练

在沐曦国产算力集群上进行类脑大模型的分布式训练面临诸多挑战，这包括：大规模并行训练的集群稳定性需求、长序列复杂并行拓扑下高强度的通信承载、混合注意力的 CUDA 和 Triton 算子适配等。在本工作中沐曦集群针对每个难点都进行了相应的优化操作以确保

SpikingBrain-7B 和-76B 模型训练的成功完成。本节将依次介绍分布式训练的适配、算子适配、训练所使用的并行拓扑。

6.1 分布式训练适配

在国产 GPU 集群上完成百亿、千亿参数规模模型训练任务，对 GPU 集群的算力、显存、通信诸多方面都提出了异一定程度的挑战。为了能够在国产 GPU 集群上高效稳定的完成训练任务，沐曦软件平台结合自身 GPU 硬件特点从 MoE 优化、计算通信并行、显存优化、自动调优与集群训练、算子融合多个方面实现了相应的适配，其中一些优化提供了“可插拔”的配置，扩展了训练框架的灵活性。

针对 MoE, 有四种策略优化模型训练初期的显存或计算压力：

- **“冷热专家”优化**：模型训练初期某些专家会被大量 token 路由，成为通信热点，通过将热门专家本地备份的方式削减初期的通信量，等待专家访问次数均衡时再取消本地备份。
- **自适应重计算**：训练初期的热点专家需要处理大量 token, 当 token 数量超过给定阈值，通过激活重计算技术降低显存开销。当训练后期专家 token 分配趋于均衡，不会触发重计算。
- **多粒度重计算** [62]：针对显存压力大的专家，将重计算分为三个层级，轻量级重计算仅计算激活函数，router 等，中度重计算会计算全连接层以及共享专家，全量重计算对整个 MoE Layer 进行重计算。通过多粒度重计算，达到计算与存储较好的平衡。
- **长度对齐**：训练过程中每个专家接收到的 token 数量不同可能引起 GEMM 效率的降低, 通过 dropping 与 padding 的方式, 对齐需处理 token 的长度, 提升整体的 GEMM 效率。

Metax 还根据模型分布式训练时通信开销高的特点，在计算-通信重叠层面做出了一些尝试。节点内部使用 SDMA 引擎高速通信数据，针对张量并行与专家并行，通过计算通信 kernel 融合的方式，减少通信 kernel 与计算 kernel 的冲突 [63]。针对显存优化，适配后的 megatron 支持细粒度 offload, 可以根据实际需求将部分 transformer layer 权重或优化器状态 offload 至 cpu；细粒度重计算，根据显存需求，选择性将 Norm 层、激活函数或单个 Transformer Layer 重计算，根据实际的显存平衡计算与显存的开销。

针对大规模集群的训练，沐曦软件栈还支持自动化调优与快速重启。自动化调优，引入一套基于算子、显存与通信三大模块的自动调优引擎：通过对常见算子基准测试，对常见网络拓扑的通信性能建模，全局遍历并行拓扑的搜索空间，确定 top-k 的并行拓扑候选，减少人工调参的频繁尝试。DLRover Flash checkpoint [64] 是一种快速保存模型权重的技术，因在大型分布式训练任务常因节点故障或网络抖动导致中断，故障导致的集群空闲和回滚训练会导致集群资源的浪费。DLRover Flash Checkpoint 机制，将训练状态（包括模型权重、优化器状态、学习率调度状态等）先写入 CPU，再异步持久化到分布式文件系统。通过 flash checkpoint 实现训练状态的快速存储，缩短写盘时间，节省了 85% 集群故障导致的训练回滚和空闲时间。另外内置工具可以自动给训练各环节“埋点”监测，能分层查看耗时，自动告警定位慢节点，节省排查时间，维持集群高速运行。

6.2 算子适配

类脑大模型 SpikingBrain 在国产沐曦 GPU 上的高效适配依托于完整的沐曦生态体系，其整体过程可以划分为 Triton 适配和 CUDA 向沐曦自研 MACA 框架的迁移两个主要部分。两条路径分别适用于 SpikingBrain 模型内部算子的不同部分，两者相辅相成，共同构成适应于沐曦 GPU 的类脑大模型硬件适配框架。（详见图 4）。

在 Triton 适配流程中 ，我们基于沐曦技术栈和 Triton 编译流程设计了四个逐步递进的阶段，以充分发挥沐曦 GPU 在编译优化和指令调度方面的潜力，同时保持适配过程对上层应用的透明性：

- **JIT 编译优化**：在 Triton 的即时编译环节中，对 SpikingBrain 用到的 Gated Linear Attention[12] 算子进行代码级别的重新组织，通过指令流水线优化与寄存器分配调整，使得算子在访存延迟和计算密度之间达到平衡。该阶段直接复用了 Triton 框架的硬件无关通用优化策略，实现了计算内核的动态高效执行。
- **网格搜索与架构匹配**：通过对不同 Block/Grid 配置的系统性搜索，结合沐曦 GPU 的流式多处理器规模和线程并发度特性，找到最优的映射策略。这一过程通过扩展 Triton 所支持的架构到沐曦 GPU 范围实现，显著提升了线性注意力算子中矩阵乘法部分的吞吐率。
- **固化缓存结构**：为减少重复计算与访存开销，Triton 内核中引入了针对沐曦片上缓存层级的固定结构化设计。例如，针对 L2 缓存容量与带宽特性，优化了权重矩阵与 Key-Value 序列的缓存复用策略，使得长序列推理的访存效率得到大幅提高。
- **沐曦编译器生成目标机器码**：在最后阶段，Triton 内核经过沐曦编译器后端转化为用于沐曦 GPU 执行的目标机器码。编译器不仅完成常规的代码生成，还针对沐曦 GPU 的张量核心、SIMD 指令集以及特定的访存对齐需求做了深度优化，确保算子在硬件层面以接近峰值性能的方式运行。

在 CUDA 向 MACA 框架迁移的过程中 ，同样包含四个紧密衔接的环节：

- **调用层适配**：对原有 CUDA API 与运行时接口进行封装与重定向，使其能够在不改变用户调用方式的前提下，平滑映射到沐曦的 MACA 框架。这一设计极大降低了开发者迁移成本。
- **优化分析**：结合性能剖析工具，识别出在 SpikingBrain 模型中表现出瓶颈的算子，如长序列 Attention 的 softmax、exp/sum、点积累积（dot-product accumulation）以及部分归一化操作。通过对这些算子进行 MACA 原生优化，充分利用硬件的张量加速单元。
- **缓存与架构匹配**：与 Triton 适配类似，进一步在 MACA 执行引擎中嵌入对沐曦 GPU 缓存层次的感知机制，使关键数据（如中间累积矩阵、位置编码缓存）能够驻留在高速缓存中，减少全局显存往返开销，并针对性地进行硬件相关优化。

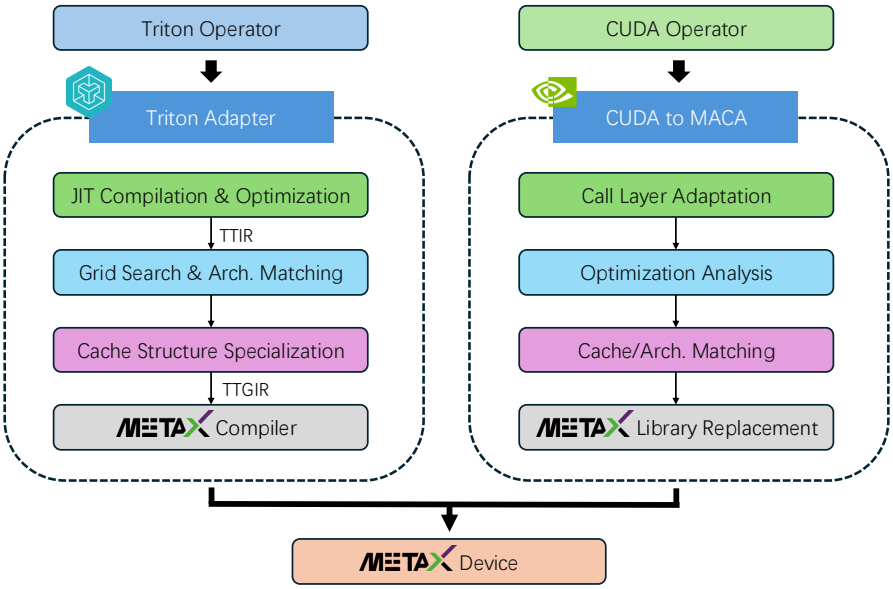

图 4: **沐曦平台上的 CUDA 和 Triton 算子适配**。适配从两条互补路径出发：基于 Triton 的适配与迁移至 MACA 框架的 CUDA 改写，分别覆盖不同的算子子集，并共同构成面向 MetaX GPU 的统一硬件适配框架。

- **替换沐曦算子加速库**：最终，将部分基础算子替换为沐曦自研的高性能库函数，使用 Hotspot 算子加速库对基础调用进行替代，并使用专门为沐曦硬件优化的 mcFlashInfer2 等高级算子库，进而在硬件底层获得更稳定的性能提升。

值得强调的是，类脑大模型针对沐曦硬件的整个适配与迁移过程始终坚持用户友好性的设计理念。无论是 Triton 内核优化，还是 CUDA 向 MACA 的迁移，下游用户均可在保持原有的编程习惯和接口调用方式的前提下，无需大量修改模型代码即可在沐曦 GPU 上顺利适配并充分利用硬件性能。与此同时，沐曦生态提供了统一的调试与性能分析工具，使开发者能够透明地观察模型在硬件上的执行特性，从而进一步微调和优化。

6.3 并行技术和训练拓扑

训练大型语言模型所需的内存通常超出单个 GPU 的容量。为了使此类模型的训练切实可行，必须采用高效且可扩展的分布式训练技术以及其他内存缩减方法。这些方法可以在保持训练效率的同时，分散计算和存储负载。

数据并行（DP） 数据并行 [65] 的核心是将训练数据划分为多个批次，每个批次由独立的 GPU 处理。每个 GPU 都维护着完整的模型副本，并独立执行前向传播与反向传播计算，仅在反向传播开始时进行梯度同步。数据并行具有通信开销低、可扩展性强的优势：随着计算资源的增加，数据并行的规模可相应扩大，通过提升总批次大小提高整体吞吐量，从而缩短训练时间。在训练阶段，我们采用 ZeRO（Zero Redundancy Optimizer）[66] 消除优化器状态的冗余，有效分散 GPU 的内存压力。

流水线并行（PP） 流水线并行按“层”将模型划分为不同阶段，每个 GPU 分配模型的一个子层集合。在前向传播与反向传播过程中，GPU 之间通过点对点（peer-to-peer, p2p）通信传递中间激活值。为实现高效的流水线执行，我们采用 1F1B [67, 68] 调度算法（即每完成一个阶段的前向计算就启动该阶段的反向计算）。此外，通过在混合专家层中穿插密集型前馈网络层，可缓解“流水线首阶段内存占用通常高于其他阶段”的内存不平衡问题。

专家并行（EP） 在混合专家模型中，大部分参数集中于专家网络。专家并行 [69] 通过将这些专家网络跨多个 GPU 进行分割，解决参数集中带来的内存与计算压力。这需要通过全对全 (all-to-all, a2a）通信进行 tokens 的分发和收集。我们采用分组 GEMM 核来加速多专家间的计算，并引入辅助损失函数，以促进 token 在不同专家间的均衡路由。

序列并行（SP） 长序列训练中，存储中间激活值会产生巨大的内存开销。序列并行通过沿“序列长度维度”分割输入序列来解决这一问题，使得每个 GPU 处理输入序列的一个连续片段。再通过全对全 (all-to-all, a2a）或点对点等通信原语在设备间交换必要信息，从而实现对整个序列的自注意力机制计算。在实验中，我们采用 DeepSpeed Ulysses [70] 框架实现高效的序列并行；此外，本研究还针对线性注意力的变体设计了专用序列并行算法，以进一步优化性能。在线性注意力的实现中，我们根据序列并行（SP）的规模采用了不同的通信策略。对于小规模的序列并行，我们通过 AllGather 操作来汇集各个设备的中间局部状态 [71]。而对于更大规模（如跨节点）的并行场景，我们转而使用 ZeCO [72] 序列并行技术中的 All-Scan 原语，从而有效地消除了通信瓶颈。

SpikingBrain-7B 训练拓扑 SpikingBrain-7B 模型的训练基于 Colossal-AI [25] 框架。在处理 128k 的长序列时，我们设计了 32 路数据并行与 8 路序列并行相结合的并行方案。为了优化显存占用，我们整合了 ZeRO-2 [66] 和激活重计算（activation recomputation）[73, 62] 两项关键技术。其中的序列并行功能基于 DeepSpeed Ulysses [70] 实现，它通过在节点内进行 all-to-all 通信来交换数据。为了满足序列并行的切分条件，我们对注意力头的数量进行了 padding 操作。

SpikingBrain-76B 训练拓扑 SpikingBrain-76B 模型的训练则基于 Megatron [24] 框架。在 8k 序列长度的基础训练阶段，我们采用了 128 路数据并行、8 路专家并行与 4 路流水线并行相结合的并行设置，通过 ZeRO [66] 和选择性激活重计算 [73, 62] 技术来降低显存开销并使用辅助损失函数保证专家并行间的负载均衡。当序列长度扩展至 32k 和 128k 时，我们在原有并行策略的基础上，分别使用 4 路和 8 路的序列并行。76B 混合模型架构融合了线性注意力和（局部）Softmax 注意力。针对这两种不同的注意力机制，我们采用了不同的并行实现：对于 Softmax 注意力分支，我们利用 DeepSpeed Ulysses [70] 技术，通过 all-to-all 通信和注意力头 padding 来实现高效的序列并行切分；对于线性注意力分支，我们则采用基于 AllGather 的序列并行算法，以有效降低其通信开销。

7 结果展示

7.1 下游任务评测

我们首先将基于转换得到的两款高效模型在多个下游任务[2]上进行了全面评测，并与同量级的开源 Hybrid、Transformer 与 MoE baselines 进行对比，其中也包括性能表现突出的基座模型 Qwen2.5-7B。为了确保结果的可比性，所有测试均在相同条件下完成，并统一使用 OpenCompass [74] 框架。在评价指标选择上，我们更关注与预训练相关的通用性指标，因为这类指标能够较好地反映模型在经过少于 200B tokens 的高效转换训练后，是否能够有效继承基座模型的知识与建模能力，并在保持良好泛化性的同时展现出显著的长序列效率优势，而不会明显受限于数据质量的影响比如 SFT 后的对齐能力。

如表 1 所示，我们的 7B 纯线性模型在多个 benchmark 上均恢复了接近 90% 的基座模型性能，整体水平已达到与 Mistral-7B、Llama3-8B 等先进 Transformer 模型相当的水准。这表明高效的线性注意力架构在合理的训练策略下，能够在大幅降低推理复杂度的同时保持较强的通用建模能力。然而需要指出的是，由于注意力机制的显著结构改动，相比于 Qwen2.5 base，转换的 7B 纯线性模型的性能仍存在一定差距，这表明在追求极致效率的同时，模型能力的恢复程度可能受到架构变化的约束。

表 1: **SpikingBrain-7B 预训练模型的性能测评**。所有模型均基于 Huggingface 框架进行推理，benchmark 均使用 ppl-based evaluation。除 Qwen2.5 以外，其他 baselines 的中文训练较差，因此 CMMLU 和 Ceval 呈现明显劣势。

	SpikingBrain-7B	Falcon-Mamba [75]	Mistral [76]	Zamba-v1 [77]	Llama3.1 [78]	Qwen2.5 [79] (base)
Params	7B	7B	7B	7B	8B	7B
Tokens	+150B	5.8T	–	1T	15T	7T
Complexity Type	Linear	Linear	Linear	Hybrid	Quadratic	Quadratic
Benchmarks						
MMLU [80]	**65.84**	63.24	62.56	58.19	65.74	74.21
CMMLU [81]	**71.58**	42.50	44.58	38.42	52.44	81.73
ARC-C [82]	**43.32**	47.53	45.13	37.18	51.96	44.04
HS [83]	**70.89**	71.50	75.81	52.02	71.60	72.81
Ceval [84]	**69.80**	41.93	47.04	36.40	51.46	81.60

进一步地，如表 2 所示，76B 混合线性 MoE 模型在扩大参数规模并引入更精细的注意力设计后，几乎完全恢复了基座模型的性能表现。实验表明，该模型在使用更少激活参数的情况下，性能已经接近甚至超过了 Llama2-70B、Mixtral-8×7B、Gemma2-27B 等一系列代表性的 Transformer 模型。且相比于 7B 模型，76B 在各项指标上都有全面提升，这充分说明在转换框架下，通过在 Attention 与 FFN 结构上的灵活设计，可以在性能与效率之间实现更优的权衡，在实践中可以按需选择架构偏好。

之后，我们基于转换得到的预训练模型进行了三阶段的 SFT 训练，以进一步赋予模型指令跟随与链式思维 (CoT) 的推理能力。如表3评测结果显示，在通用知识、长序列建模与指令遵循等多个维度上，我们的对齐模型均取得与开源的同量级 chat 模型 baselines 相当

[2]本节评估架构设计和转换流程的有效性。所有模型均保留浮点格式；脉冲模型的结果在小节 7.5 中展示.

表 2: **SpikingBrain-76B 预训练模型的性能测评**。所有模型均基于 vLLM 框架进行推理，benchmark 均使用 ppl-based evaluation。除 Qwen2.5 以外，其他 baselines 的中文训练较差，因此 CMMLU 和 Ceval 呈现明显劣势。

	SpikingBrain-76B (76B-A12B)	Jamba [32] (52B-A12B)	Mixtral-8*7B [85] (47B-A13B)	LLama2-70b [86]	Gemma2-27B [87]	Qwen2.5 [38] (base)
Params	12B/76B	12B/52B	13B/47B	70B	27B	7B
Tokens	+160B	–	–	2T	13T	7T
Complexity Type	Hybrid	Hybrid	Quadratic	Quadratic	Quadratic	Quadratic
Benchmarks						
MMLU [80]	**73.58**	67.17	71.23	69.57	75.94	75.31
CMMLU [81]	**78.83**	51.11	52.70	52.94	61.80	81.50
ARC-C [82]	**42.00**	49.10	48.98	46.21	57.64	43.56
HS [83]	**73.31**	79.39	79.42	79.15	79.34	73.37
Ceval [84]	**78.89**	48.94	54.39	49.26	61.02	81.68

的性能。同时值得注意的是，SFT 过程并未导致模型过拟合或损害预训练中获得的通用能力，这表明我们的（混合）线性架构在对齐训练中具备稳定性与可扩展性。由于目前数据质量的限制，我们主要展示了模型在国产集群上的有效对齐训练结果，而更大规模、更高质量的后训练对齐将留待未来工作。

表 3: **SpikingBrain Chat 模型的性能测评**。所有模型均基于 Huggingface 框架进行推理，benchmark 均使用 generation-based evaluation。除 Qwen2.5 以外，其他 baselines 的中文训练较差，因此 CMMLU 和 Ceval 呈现明显劣势。对于 QuALITY 和 IFEval, 我们展示了经过两阶段 SFT 后的 non-CoT 模型的结果，来避免思维链对输出结果的干扰。

	SpikingBrain-7B	**SpikingBrain-76B**	Llama3 [78]	Qwen2.5 [38]	Mixtral [85]
Params	7B	12B/76B	8B	7B	13B/47B
Complexity Type	Linear	Hybrid	Quadratic	Quadratic	Quadratic
Benchmarks					
MMLU [80]	**65.57**	**73.71**	68.69	75.17	71.03
CMMLU [81]	**68.76**	**77.41**	55.17	79.14	51.03
HS [83]	**68.95**	**86.63**	76.80	85.39	75.63
Ceval [84]	**69.07**	**76.32**	55.01	77.93	50.88
NQ [88]	**21.47**	**21.55**	30.97	17.67	28.48
TrQ [89]	**57.03**	**55.13**	65.78	55.72	71.00
QuALITY [90]	**60.12**	**69.56**	66.25	73.63	51.34
IFEval [91]	**42.70**	**49.72**	73.01	73.20	48.06

7.2 长序列处理速度测评

除了任务性能，我们还特别关注了模型在长序列推理场景下的效率表现。得益于（混合）线性注意力与 MoE 机制的结合，两款模型在推理速度方面展现出了显著优势。考虑到 7B 模型参数量更紧凑、效率优势更明显，我们基于该模型在 HuggingFace 框架下适配了多卡序列并行推理 (ZeCO + a2a/p2p)，并成功支持最长 4M 的高效 Prefill。如表 4 和图 5 所

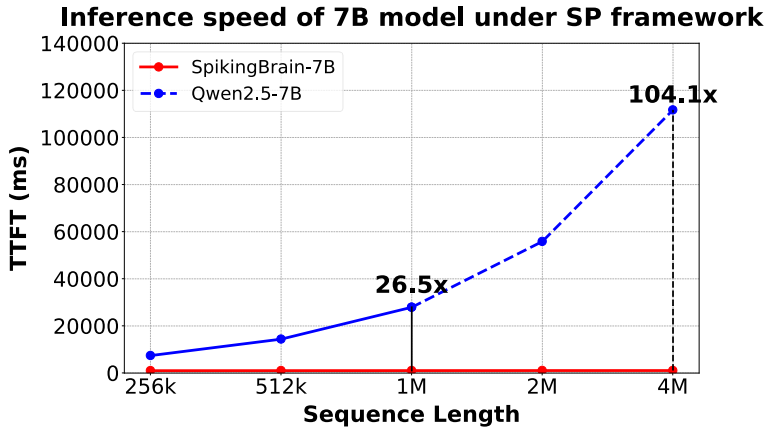

图 5: **序列并行下的 TTFT 比较。** SpikingBrain-7B 与 Qwen2.5-7B 基线在不同输入长度上的生成首个 token 时间 (TTFT) 延迟比较。对于超过 2M Token 的输入，Qwen2.5-7B 由于资源限制和注意力头数的约束，难以直接评估；因此，我们通过拟合缩放曲线推算得出结果。

示，在输入长度达到 1M 时，7B 模型的 TTFT（生成首个 token 的时延）相较于使用 Full Attention 与 A2A 通信的 Qwen baseline 提升 26.5 倍，并且随着序列长度和 GPU 数量的扩展，推理时间开销几乎保持恒定。由于资源和模型注意力头数的限制，Qwen baseline 在 4M 长度下已经无法评测，根据拟合 scaling 曲线，保守估计速度提升超过 100 倍。这一结果充分证明了我们方法在极长序列场景下的优越性与可扩展性，为更高效大模型在实际应用中的落地提供了有力支撑。

表 4: **在序列并行（SP）条件下 SpikingBrain-7B 的推理速度比较。**评测指标为 TTFT（毫秒），即完成预填充并生成首个 token 的时延，取决于输入提示的长度。在 SP 配置下，我们的 7B 模型在线性注意力模块中使用 ZeCO，在 SWA 模块中采用 P2P 通信，而 Qwen2.5 基线则使用 A2A 通信。所有测量均在 NVIDIA H100 GPU 上进行，并取 10 次运行的平均值。符号“–”表示由于资源限制无法测量。

Sequence length	GPU count	**SpikingBrain-7B**	Qwen2.5-7B
256k	8	1015	7419
512k	16	1037	14398
1M	32	1054	27929
2M	64	1070	–
4M	128	1073	–

同时，我们还针对 7B 和 76B 规模的模型，在未启用序列并行技术的条件下，进行了系统性的推理速度评测。测试环境涵盖了在 MetaX C550 GPU 上的 HuggingFace 单卡框架以及 vLLM 的单卡/多卡部署，实验结果如表 5 所示。可以观察到：在任意框架下，我们的 7B 线性模型在 128k 长度推理时均展现出相较 Qwen 的成倍加速效果 (1.41× on HuggingFace, 2.75× on vLLM)，并且在性能上与采用 SWA 技术、同样具备线性复杂度的 Mistral 模型相

当甚至更优。对于大规模的 76B 模型, 其推理速度同样表现突出, 不仅显著优于 Llama2-70B, 在开启 Expert Parallel (EP) 的场景下，甚至超过了 MoE 基线模型 Mixtral-8x7B。这些结果充分验证了我们方法在不同模型规模和推理框架下的广泛适用性和性能优势。

在长序列训练的场景下，我们同样基于 7B 模型和 Qwen baseline 在序列并行框架下进行了分布式训练的 TGS 测试。实验结果显示在 128k 的长度下我们的 7B 模型在训练吞吐量上展现 5.36× 倍的优势，和推理时 TTFT 的表现一致。

7.3 CPU 侧推理

在本节中，我们展示了如何在基于 CPU 的移动推理框架上部署压缩后的 1B 参数规模模型，其中包括 SWA 和 GLA 混合的 SpikingBrain 线性模型，以及 Llama3.2[78]。本次部署以 llama.cpp 作为推理后端。如图 6 所示，整个工作流由四个主要步骤构成：

（1）在模型适配过程中，首先需将训练完成的模型权重转换为 GGUF 格式，并结合合适的量化策略。与通用训练格式相比，GGUF 强调紧凑编码与跨平台一致性，在 CPU 推理中的加载时延与常驻内存占用更可控。通过将浮点数权重压缩为低比特整数（如 Q4、Q5 等方案），并在计算过程中配合高效的解码机制与矩阵乘法内核，可以显著提升计算吞吐率, 同时有效降低内存占用。（2）随后，我们为模型建立专用的计算图映射机制。在该过程中, 需要针对模型的特定层级，在推理引擎代码中构建架构注册表与张量映射表。这样一来，加载的权重文件中的每个张量均可与计算图中的结构一一对应，并附带其形状、精度格式（如 float32、float64 等）等信息。(3) 在此基础上，我们依据模型的前向传播流程实现关键算子, 包括注意力机制、旋转位置编码（RoPE）[92]、残差连接等操作。同时，引入一系列计算图优化措施，如算子融合（Operator Fusion）：将矩阵乘法、偏置加法与非线性激活函数整合为单一执行内核，从而减少中间存储与访存开销，提升缓存利用率；图级优化（Graph-level Optimization）：将 RMSNorm、Softmax、RoPE 等操作进行拆解与重组，使其可融合到矩阵乘法或底层内核中，从而降低额外计算成本。(4) 最终加载量化后的权重，并以低精度执行前向计算。默认情况下，对权重应用 Q4_0 量化，以平衡计算吞吐量与内存使用。此外, 我们使用紧凑的单步整数激活而不是展开的脉冲序列，以确保与当前 CPU 硬件的兼容性。

值得强调的是, SpikingBrain-1B 线性模型的一个关键创新点在于其线性注意力与 SWA 的层间混合，这避免了对全部历史 Keys 与 Values 的全量计算。这一设计不仅有效降低了 KV-cache 占用, 还显著减少了推理延迟。相比于传统基于 Transformer 架构的模型, 我们的计算图显式地整合了这些机制，以进一步优化 KV-缓存管理。我们使用 1k 的输入来评估不同输出长度下的解码速度。硬件环境包括：Intel Core i5-12600KF CPU、64 GB 内存，操作系统为 Ubuntu 22.04.4 LTS，通过 CMake 3.28.3 进行编译。如图 7 所示，SpikingBrain-1B 模型在解码过程中保持恒定的计算和存储开销，随着输出序列长度的增加，吞吐量也保持稳定。相比之下，Llama3.2-1B 基线模型由于需要进行完整的 KV-缓存计算，解码速度急剧下降。总体而言，与 Llama3.2-1B 基线相比，我们的模型在序列长度分别为 64k、128k 和 256k 时分别实现了 4.04 倍、7.52 倍和 15.39 倍的加速。该工作流使推理引擎能够无缝地将类脑模型适配到 CPU 上进行部署，在计算效率、内存利用率和硬件适配性方面取得了显著提升。这为在资源受限的环境中高效运行大模型提供了一种实用的解决方案。

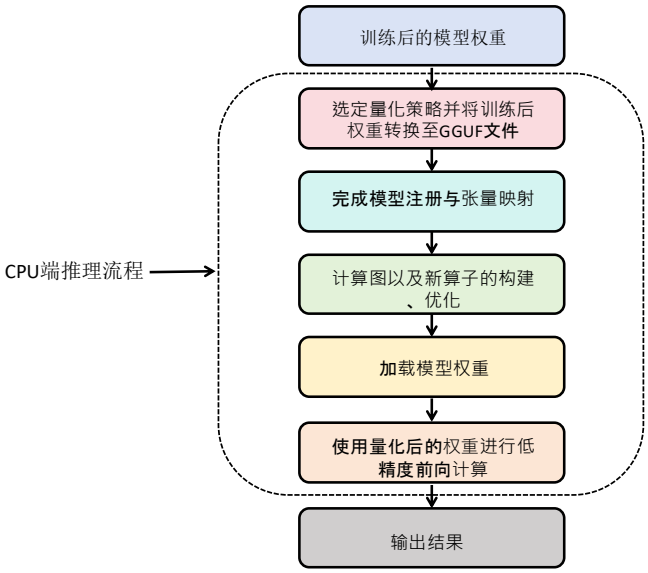

图 6: **CPU 端推理流水线概述**。工作流程包括四个主要步骤：权重量化与转换 GGUF、模型注册与张量映射、计算图与算子优化，以及量化推理。

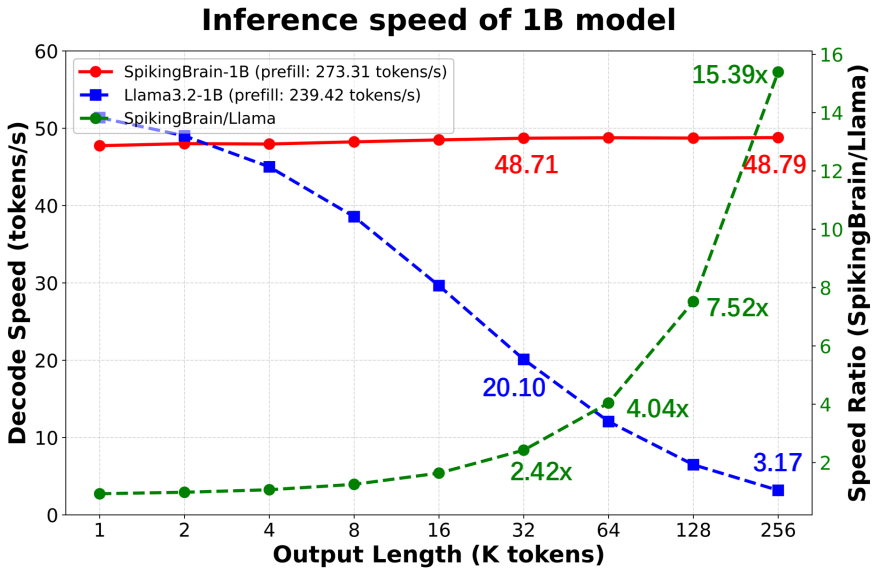

图 7: **在基于 CPU 的移动推理框架下, 不同输出长度的解码速度的比较**。图片展示了 1B 参数 SpikingBrain 线性模型和 Llama3.2-1B 基线的结果，该评估均在 Intel Core i5-12600KF CPU 上进行。默认情况下，采用 Q4_0 权重量化来平衡吞吐量和内存效率。

表 5: **不同模型、框架与序列长度下的推理时延（毫秒）比较**。测试环境包括在 MetaX C550 GPU 上的 HuggingFace 单卡部署，以及在 vLLM 下的单卡与多卡部署。时延的度量方式为处理输入序列并生成 128 个输出 token 所需的时间。

	SpikingBrain -7B	Mistral -7B-v0.1	Qwen2.5 -7B	SpikingBrain-7B	Mistral -7B-v0.1	Qwen2.5 -7B	**SpikingBrain -76B**	Mixtral -8x7B-v0.1	Llama -2-70b
GPUs	1			1			4		
Frameworks	HuggingFace			vllm			vllm		
Expert Parallel	Non-MoE						Enable / Disable		Non-MoE
32k	8452	10202	7969	4369	4570	6421	9451 / 5305	10477 / 3209	15881
64k	11685	16467	16257	7399	7084	15093	14077 / 8142	19062 / 4840	32953
128k	15974	28382	38487	12048	11867	45141	27106 / 14109	34983 / 8046	86120

7.4 国产集群训练指标

我们从训练效率与系统稳定性两个方面衡量 MetaX GPU 集群的训练性能指标。在训练效率指标方面，包括模型 FLOPs 利用率（MFU）和每秒每 GPU 处理 Tokens (TGS)，MetaX GPU 集群在整个模型训练过程中表现出较好的性能。其中，SpikingBrain-7B 模型的 TGS 达到 1558，MFU 达到 23.4% (8-way DP, 4-way PP, PP micro-batch size 2, global batch size 512)，体现了较高的计算效率和有效的资源利用。

在系统稳定性方面，通过系统监测集群运行，集群表现出显著的可靠性和鲁棒性。训练工作持续进行了超过两周的时间，中途过程没有中断，展现了国内硬件和软件生态系统的稳定性和成熟度。

这些结果突出了国内 GPU 集群高效可靠地支持大规模、长时间训练任务的能力，验证了 MetaX GPU 集群能够稳定完成百亿级参数模型的训练工作。

7.5 脉冲化结果分析

我们以 SpikingBrain -7B 模型上应用激活值脉冲化编码 - 8 位定点权重量化（INT8 Quantization）协同优化方案为例，通过脉冲化和异步硬件事件驱动计算范式实现能效与精度的平衡。

7.5.1 方案部署与精度保持

此外，我们还测试了模型在常用语言建模 Benchmarks 上的性能对比。具体来说，我们对模型中所有线性投影层的激活输入应用脉冲化编码，同时使用对称 INT8 量化其相应的权重。使用 128 个文本样本的小集合对量化参数进行校准调优。如表 6 所示，在常识推理、MMLU 和 CMMLU 基准测试中的评估表明，该方案下的平均性能下降在 SpikingBrain-7B 和-76B 模型中都被限制在约 2% 以内，这证实了方案在精度保留上的有效性。

7.5.2 脉冲分布特性

为验证脉冲化编码的稀疏、高效特性，本研究对按位三值 (bitwise–ternary) 脉冲编码模式下脉冲发放的统计特性展开分析：

表 6: **应用脉冲化方案前后 SpikingBrain 模型的性能比较**。评估基于常识推理基准、MMLU 和 CMMLU 进行，结果表明自适应阈值脉冲编码与 8 位权重量化相结合导致的性能下降可忽略不计（约 1–3% 之间）。

	Winogrande	Arc-e	Arc-c (norm)	HellaSwag (norm)	PIQA	MMLU	CMMLU	Avg.
SpikingBrain-7B	0.6992	0.8047	0.5566	0.6777	0.7949	0.6751	0.6904	0.6998
SpikingBrain-7B -W8ASpike	0.6895	0.7861	0.5410	0.6758	0.7979	0.6546	0.6677	0.6875
SpikingBrain -76B	0.7275	0.8125	0.5615	0.7000	0.8125	0.7247	0.7740	0.7304
SpikingBrain -76B-W8ASpike	0.7148	0.7949	0.5371	0.6863	0.8004	0.7081	0.7512	0.7133

我们收集了所有线性投影层输入脉冲整数的绝对值。脉冲计数的详细分布如图 8所示。以 SpikingBrain-7B 模型的分布为例，结果显示 94.1% 的值落在 [0,7] 范围内，只有约 1% 的值超过 16。扩展为按位三值脉冲后，每个通道发放的平均脉冲数量仅为 1.13，而 18.4% 的通道根本没有发放任何脉冲。由于 94.1% 的值在 7 范围内（即在 INT3 范围内），我们进一步分析了步长为 3 下的发放率。通过用 0 填充 3 步窗口内的非活动位，整体稀疏度达到约 69.15%。

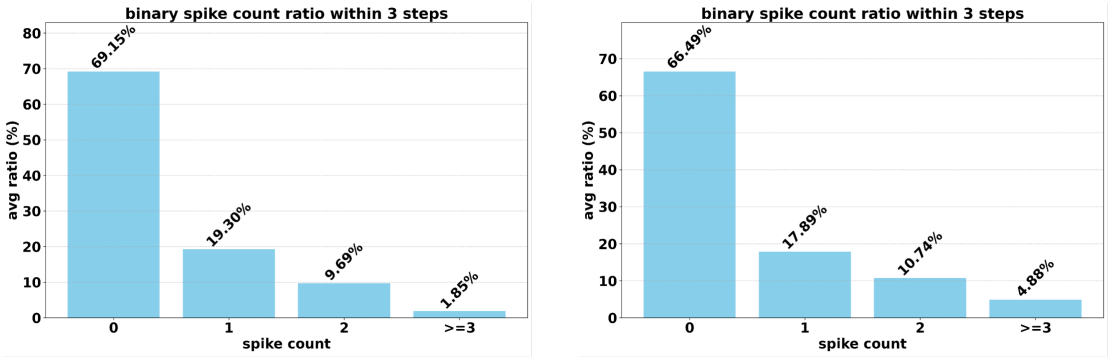

图 8: **按位脉冲编码方案的脉冲计数分布**。分别展示了 SpikingBrain-7B（左）和 SpikingBrain-76B（右）的结果。

7.5.3 脉冲可视化接口

我们提供了统一的可视化接口来展示不同编码策略下的神经脉冲发放过程，如图 9所示对比了传统的二值脉冲（Binary Spike Coding）、三值脉冲（Ternary Spike Coding）以及位编码脉冲（Bitwise Spike Coding）的时间-神经元二维发放图：

- Binary 脉冲编码通常需要较长的时间步和较高的发放频率，整体脉冲分布较为密集；
- Ternary 脉冲编码在发放表达上更为稀疏，依靠正负脉冲的引入显著压缩了发放总量和所需时长；
- Bitwise 脉冲编码则进一步提升了压缩能力，通过位展开策略在保证表达精度的同时显著减少时间开销。该可视化工具也为分析不同编码策略下的脉冲行为提供了直观支撑。

7.5.4 异步硬件能效优化机制

统计显示，模型推理过程中存在 18.4% 的通道无脉冲输出（即脉冲数为 0）。在异步硬件的事件驱动计算范式下，无脉冲触发时将省略这部分权重读取路径（含片外 DRAM 至片上 SRAM 的搬运、SRAM 至计算单元的数据搬运），直接等比例减少内存访问开销。

在计算方面，基于 45nm 的硬件能耗数据，对不同计算范式的 MAC 操作能耗进行估算对比：

- 传统 16 位浮点计算（FP16 MAC）：单次操作能耗为 1.5pJ，需完成完整的浮点乘法与加法运算，无关激活值是否有效；
- 8 位定点计算（INT8 MAC）：单次操作能耗降至 0.23pJ，但仍依赖同步时钟驱动，存在无效时钟周期的能耗开销；
- 本方案（事件驱动脉冲化计算 + 权重 INT8 量化）：MAC 能耗基于“脉冲触发 - 权重加法”估算，即能耗 = 平均脉冲数 ×INT8 权重加法能耗（INT8 加法器能耗约 0.03pJ / 次），计算得平均单次 MAC 能耗约 0.034pJ（1.13×0.03pJ）。

结果显示，与 FP16 MACs 相比，我们的方案能耗降低了 97.7%；与 INT8 MACs 相比，能耗降低了 85.2%，分别带来了高达 43.48 倍和 6.76 倍的能效提升。这表明，将脉冲驱动计算与量化相结合，能够有效大幅降低能耗开销，同时做到精度损失可控。

8 结论

本项目全面展示了我们在沐曦国产 GPU 集群上进行高效类脑大模型训练的创新实践。我们通过一系列关键技术，包括非 Transformer 的新型（混合）线性架构、稀疏 MoE 开发、轻量级通用转换训练方案，以及动态阈值的稀疏脉冲激活编码，成功在数百卡规模的国产集群上验证并践行了高效的类脑脉冲大模型开发流程。

此次研究成功发布了两款模型：线性类脑脉冲大模型 SpikingBrain-7B 和 MoE 混合线性类脑脉冲大模型 SpikingBrain-76B-A12B。这两款模型的核心优势在于：（1）训练效率：模型具备线性或近线性的复杂度，显著提升了长序列训练效率，并能以极低的数据量（仅不到 2%）实现与众多开源 Transformer 模型相媲美的性能。（2）推理性能：模型在推理阶段具有事件驱动的脉冲特性，并展现出常数（7B）或部分层常数（76B）级别的复杂度与存储开销，在长序列处理上实现了数量级的效率和速度飞跃（例如，1M 长度 TTFT 加速超过 20 倍）。这一突破性实践不仅为国产算力平台上的高效大模型研发提供了宝贵经验，更对未来大模型的规模化部署与应用开辟了新路径。

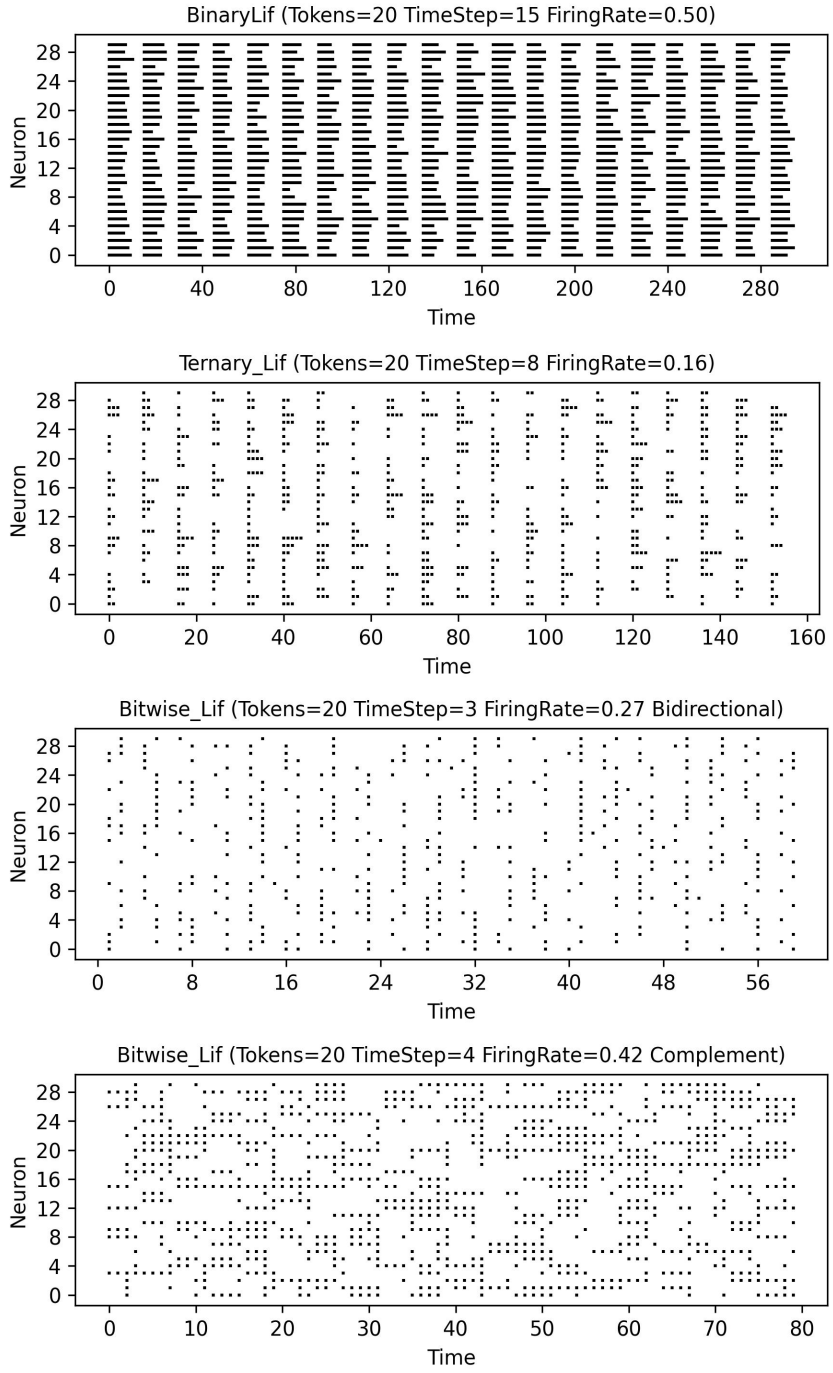

图 9: **不同脉冲编码方案下的时间–神经元发放图**。展示了在相同输入条件下，不同脉冲编码策略下的脉冲发放分布，包括二值、三值以及两种位编码。横轴为时间（Time），表示 token 时间步 × 扩展时间步的总时长；纵轴为神经元编号（Neuron）。黑点表示对应神经元在该时间点发生脉冲发放。

参考文献

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[2] OpenAI. GPT-5. `https://openai.com/gpt-5/`, 2025.

[3] Google DeepMind. Gemini 2.5 Pro. `https://deepmind.google/models/gemini/pro/`, 2025.

[4] Anthropic. Introducing Claude Opus 4.1. `https://www.anthropic.com/news/claude-opus-4-1`, 2025.

[5] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[6] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[7] Linxuan He, Yunhui Xu, Weihua He, Yihan Lin, Yang Tian, Yujie Wu, Wenhui Wang, Ziyang Zhang, Junwei Han, Yonghong Tian, et al. Network model with internal complexity bridges artificial intelligence and neuroscience. *Nature Computational Science*, 4(8):584–599, 2024.

[8] Man Yao, JiaKui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. In *Advances in Neural Information Processing Systems*, volume 36, pages 64043–64058, 2023.

[9] Man Yao, JiaKui Hu, Tianxiang Hu, Yifan Xu, Zhaokun Zhou, Yonghong Tian, Bo XU, and Guoqi Li. Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. In *The Twelfth International Conference on Learning Representations*, 2024.

[10] Yuhong Chou, Man Yao, Kexin Wang, Yuqi Pan, Rui-Jie Zhu, Jibin Wu, Yiran Zhong, Yu Qiao, Bo XU, and Guoqi Li. MetaLA: Unified optimal linear approximation to softmax attention map. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[11] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[12] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *International Conference on Machine Learning*, pages 56501–56523. PMLR, 2024.

[13] Tri Dao and Albert Gu. Transformers are ssms: generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, pages 10041–10071, 2024.

[14] Xinhao Luo, Man Yao, Yuhong Chou, Bo Xu, and Guoqi Li. Integer-valued training and spike-driven inference spiking neural network for high-performance and energy-efficient object detection. In *European Conference on Computer Vision*, pages 253–272. Springer, 2025.

[15] Man Yao, Xuerui Qiu, Tianxiang Hu, Jiakui Hu, Yuhong Chou, Keyu Tian, Jianxing Liao, Luziwei Leng, Bo Xu, and Guoqi Li. Scaling spike-driven transformer with efficient spike firing approximation training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(4):2973–2990, 2025.

[16] Liangwei Fan, Hui Shen, Xiangkai Lian, Yulin Li, Man Yao, Guoqi Li, and Dewen Hu. A multisynaptic spiking neuron for simultaneously encoding spatiotemporal dynamics. *Nature Communications*, 16(1):7155, 2025.

[17] Man Yao, Ole Richter, Guangshe Zhao, Ning Qiao, Yannan Xing, Dingheng Wang, Tianxiang Hu, Wei Fang, Tugba Demirci, Michele De Marchi, Lei Deng, Tianyi Yan, Carsten Nielsen, Sadique Sheik, Chenxi Wu, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-based dynamic computing with asynchronous sensing-computing neuromorphic chip. *Nature Communications*, 15(1):4464, 2024.

[18] Guoqi Li, Lei Deng, Huajin Tang, Gang Pan, Yonghong Tian, Kaushik Roy, and Wolfgang Maass. Brain-inspired computing: A systematic survey and future trends. *Proceedings of the IEEE*, 112(6):544–584, 2024.

[19] Corinne Teeter, Ramakrishnan Iyer, Vilas Menon, Nathan Gouwens, David Feng, Jim Berg, Aaron Szafer, Nicholas Cain, Hongkui Zeng, Michael Hawrylycz, et al. Generalized leaky integrate-and-fire models classify multiple neuron types. *Nature communications*, 9(1):709, 2018.

[20] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[21] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[22] Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length, 2018.

[23] Daniel Goldstein, Fares Obeid, Eric Alcaide, Guangyu Song, and Eugene Cheah. Goldfinch: High performance rwkv/transformer hybrid with linear pre-fill and extreme kv-cache compression, 2024.

[24] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[25] Shenggui Li, Hongxin Liu, Zhengda Bian, Jiarui Fang, Haichen Huang, Yuliang Liu, Boxiang Wang, and Yang You. Colossal-ai: A unified deep learning system for large-scale parallel training. In *Proceedings of the 52nd International Conference on Parallel Processing*, pages 766–775, 2023.

[26] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[27] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[28] Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *ArXiv*, abs/2310.06825, 2023.

[29] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

[30] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[31] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

[32] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

[33] Liliang Ren, Yang Liu, Yadong Lu, Chen Liang, Weizhu Chen, et al. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In *The Thirteenth International Conference on Learning Representations*, 2024.

[34] Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

[35] Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, et al. Hymba: A hybrid-head architecture for small language models. *arXiv preprint arXiv:2411.13676*, 2024.

[36] Jingwei Zuo, Maksim Velikanov, Ilyas Chahed, Younes Belkada, Dhia Eddine Rhayem, Guillaume Kunsch, Hakim Hacid, Hamza Yous, Brahim Farhat, Ibrahim Khadraoui, et al. Falcon-h1: A family of hybrid-head language models redefining efficiency and performance. *arXiv preprint arXiv:2507.22448*, 2025.

[37] Damai Dai, Chengqi Deng, Chenggang Zhao, Rx Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, 2024.

[38] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. Qwen2 technical report. *ArXiv*, abs/2407.10671, 2024.

[39] Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv preprint arXiv:2212.05055*, 2022.

[40] Ethan He, Abhinav Khattar, Ryan Prenger, Vijay Korthikanti, Zijie Yan, Tong Liu, Shiqing Fan, Ashwath Aithal, Mohammad Shoeybi, and Bryan Catanzaro. Upcycling large language models into mixture of experts. *arXiv preprint arXiv:2410.07524*, 2024.

[41] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[42] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500, 1952.

[43] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[44] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024.

[45] Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. In *First Conference on Language Modeling*, 2024.

[46] Albert Gu. On the tradeoffs of state space models and transformers, 2025.

[47] John P O' Doherty, Sang Wan Lee, Reza Tadayonnejad, Jeff Cockburn, Kyo Iigaya, and Caroline J Charpentier. Why and how the brain weights contributions from a mixture of experts. *Neuroscience & Biobehavioral Reviews*, 123:14–23, 2021.

[48] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

[49] Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention. *Advances in Neural Information Processing Systems*, 34:17413–17426, 2021.

[50] Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A Smith. Finetuning pretrained transformers into rnns. In *2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, pages 10630–10643. Association for Computational Linguistics (ACL), 2021.

[51] Jyotikrishna Dass, Shang Wu, Huihong Shi, Chaojian Li, Zhifan Ye, Zhongfeng Wang, and Yingyan Lin. Vitality: Unifying low-rank and sparse approximation for vision transformer acceleration with a linear taylor attention. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 415–428. IEEE, 2023.

[52] Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. Simple linear attention language models balance the recall-throughput tradeoff. In *International Conference on Machine Learning*, pages 1763–1840. PMLR, 2024.

[53] Jean Mercat, Igor Vasiljevic, Sedrick Keh, Kushal Arora, Achal Dave, Adrien Gaidon, and Thomas Kollar. Linearizing large language models. *arXiv preprint arXiv:2405.06640*, 2024.

[54] Junxiong Wang, Daniele Paliotta, Avner May, Alexander Rush, and Tri Dao. The mamba in the llama: Distilling and accelerating hybrid models. *Advances in Neural Information Processing Systems*, 37:62432–62457, 2024.

[55] Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. Lolcats: On low-rank linearizing of large language models. *arXiv preprint arXiv:2410.10254*, 2024.

[56] Yu Zhang, Songlin Yang, Rui-Jie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, et al. Gated slot attention for efficient linear-time sequence modeling. *Advances in Neural Information Processing Systems*, 37:116870–116898, 2024.

[57] Ge Zhang, Scott Qu, Jiaheng Liu, Chenchen Zhang, Chenghua Lin, Chou Leuang Yu, Danny Pan, Esther Cheng, Jie Liu, Qunshu Lin, Raven Yuan, Tuney Zheng, Wei Pang, Xinrun Du, Yiming Liang, Yinghao Ma, Yizhi Li, Ziyang Ma, Bill Lin, Emmanouil Benetos, Huan Yang, Junting Zhou, Kaijing Ma, Minghao Liu, Morry Niu, Noah Wang, Quehry Que, Ruibo Liu, Sine Liu, Shawn Guo, Soren Gao, Wangchunshu Zhou, Xinyue Zhang, Yizhi Zhou, Yubo Wang, Yuelin Bai, Yuhan Zhang, Yuxiang Zhang, Zenith Wang, Zhenzhu Yang, Zijian Zhao, Jiajun Zhang, Wanli Ouyang, Wenhao Huang, and Wenhu Chen. Map-neo: Highly capable and transparent bilingual large language model series. *arXiv preprint arXiv: 2405.19327*, 2024.

[58] Jijie Li, Li Du, Hanyu Zhao, Bo-wen Zhang, Liangdong Wang, Boyan Gao, Guang Liu, and Yonghua Lin. Infinity instruct: Scaling instruction selection and synthesis to enhance language models. *arXiv preprint arXiv:2506.11116*, 2025.

[59] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[60] Cong Liu, Zhong Wang, ShengYu Shen, Jialiang Peng, Xiaoli Zhang, ZhenDong Du, and YaFang Wang. The chinese dataset distilled from deepseek-r1-671b. `https://huggingface.co/datasets/Congliu/Chinese-DeepSeek-R1-Distill-data-110k`, 2025.

[61] Sathwik Tejaswi Madhusudhan, Shruthan Radhakrishna, Jash Mehta, and Toby Liang. Millions scale dataset distilled from r1-32b. https://huggingface.co/datasets/ServiceNow-AI/R1-Distill-SFT, 2025.

[62] Vijay Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. *arXiv preprint arXiv:2205.05198*, 2022.

[63] NVIDIA. Communication overlap. `https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/features/optimizations/communication_overlap.html`, 2025.

[64] Qinlong Wang, Bo Sang, Haitao Zhang, Mingjie Tang, and Ke Zhang. Dlrover: An elastic deep training extension with auto job resource recommendation. *CoRR*, 2023.

[65] Leslie G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, August 1990.

[66] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

[67] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm. *arXiv preprint arXiv:2104.04473*, 2021.

[68] Aaron Harlap, Deepak Narayanan, Amar Phanishayee, Vivek Seshadri, Nikhil Devanur, Greg Ganger, and Phil Gibbons. Pipedream: Fast and efficient pipeline parallel dnn training. *arXiv preprint arXiv:1806.03377*, 2018.

[69] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*, pages 18332–18346. PMLR, 2022.

[70] Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models. *arXiv preprint arXiv:2309.14509*, 2023.

[71] Weigao Sun, Disen Lan, Yiran Zhong, Xiaoye Qu, and Yu Cheng. Lasp-2: Rethinking sequence parallelism for linear attention and its hybrid. *ArXiv*, abs/2502.07563, 2025.

[72] Yuhong Chou, Zehao Liu, Ruijie Zhu, Xinyi Wan, Tianjian Li, Congying Chu, Qian Liu, Jibin Wu, and Zejun Ma. Zeco: Zero communication overhead sequence parallelism for linear attention. *arXiv preprint arXiv:2507.01004*, 2025.

[73] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[74] Paola A Buitrago and Nicholas A Nystrom. Open compass: accelerating the adoption of ai in open research. In *Practice and Experience in Advanced Research Computing 2019: Rise of the Machines (learning)*, pages 1–9. 2019.

[75] Jingwei Zuo, Maksim Velikanov, Dhia Eddine Rhaiem, Ilyas Chahed, Younes Belkada, Guillaume Kunsch, and Hakim Hacid. Falcon mamba: The first competitive attention-free 7b language model. *arXiv preprint arXiv:2410.05355*, 2024.

[76] Mistral AI team. Mistral 7b. https://mistral.ai/news/announcing-mistral-7b, 2023.

[77] Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7b ssm hybrid model. *arXiv preprint arXiv:2405.16712*, 2024.

[78] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.

[79] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[80] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[81] Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. Cmmlu: Measuring massive multitask language understanding in chinese. *arXiv preprint arXiv:2306.09212*, 2023.

[82] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[83] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[84] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in Neural Information Processing Systems*, 36:62991–63010, 2023.

[85] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou

Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[86] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[87] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[88] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[89] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

[90] Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel Bowman. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358, Seattle, United States, July 2022. Association for Computational Linguistics.

[91] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

[92] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *ArXiv*, abs/2104.09864, 2021.