中国科学技术大学

本科毕业论文

面向 4 比特精度大语言模型预训练的 Stable-SPAM 优化器设计与实现

作者姓名：	胡皓天
学　　号：	PB21000098
专　　业：	电子科学技术
导　　师：	金西　副教授
完成时间：	2025 年 4 月 20 日

摘要

近年来，低比特训练作为降低大语言模型（LLM）预训练计算与存储成本的关键技术，受到广泛关注。尽管已有多种优化器结构在标准精度（如 FP16/BF16）下表现出良好的收敛性与稳定性，如动量调整、梯度裁剪与自适应学习率机制等，但这些方法在更低比特精度（如 FP8/FP4）下难以直接迁移，往往会引发训练不稳定、性能退化甚至发散等问题。本质上，这一困境源于现有优化器普遍依赖高精度数值环境，以确保梯度估计与动量累积的准确性，缺乏对低比特数值噪声与状态压缩的鲁棒性适配。此外，当前低比特训练研究多聚焦于量化精度本身，较少系统性探讨如何在低精度环境中保留高精度优化器的关键机制。因此，设计兼具结构继承性与数值鲁棒性的优化算法，对提升低比特训练稳定性与效率具有重要研究价值与应用前景。

本文[①]在系统评估多种现有低比特优化器的基础上，提出了一种新型优化器 `Stable-SPAM`，以提升 4 比特训练的稳定性。该方法在继承 SPAM 动量重置与梯度裁剪机制的基础上，引入三项关键改进：（1）通过记录历史尖峰梯度的最大值，自适应更新裁剪阈值；（2）基于整个梯度矩阵的历史 l_2 范数进行归一化；（3）周期性重置一阶与二阶动量以缓解尖峰梯度积累问题，从而稳定梯度分布，并提升收敛速度。

本文实验结果表明，`Stable-SPAM` 显著提升了 4 比特大语言模型训练的稳定性与效率。在 LLaMA-1B 模型上，`Stable-SPAM` 在 4 比特浮点精度和 4 比特整数量化精度设定下达到相同损失所需步数仅为 Adam 的一半，且在相同步数下困惑度比 Adam 降低约 4 点，充分验证其在低精度训练中的性能优势与实用价值。

关键词：大语言模型；大语言模型预训练；4 比特训练；优化器，机器学习

[①]本篇毕业论文的工作基于作者在 UT-Austin 的实习成果，已投稿至双盲评审的会议。论文版权归 VITA 研究团队共同所有，未经允许不得使用。

ABSTRACT

In recent years, low-bit training has gained significant attention as a key technique for reducing the computational and memory costs of large language model (LLM) pre-training. Although various optimizer designs—such as momentum adaptation, gradient clipping, and adaptive learning rates—have demonstrated strong convergence and stability under standard precision (e.g., FP16/BF16), these methods often fail to transfer effectively to lower-precision settings (e.g., FP8/FP4), resulting in instability, degraded performance, or even divergence. Fundamentally, this limitation arises from the reliance of existing optimizers on high-precision numerical environments to ensure accurate gradient estimation and momentum accumulation, lacking robustness to quantization noise and reduced state precision. Moreover, most existing work on low-bit training focuses primarily on quantization schemes, with limited attention to systematically preserving the structural advantages of high-precision optimizers in low-precision contexts. Therefore, developing optimization algorithms that integrate structural inheritance with numerical robustness is of significant importance for improving the stability and efficiency of low-bit training.

This thesis[①] proposes `Stable-SPAM`, a novel optimizer designed to enhance the stability of 4-bit training. Building upon the SPAM optimizer, it introduces three key improvements: adaptive clipping based on historical gradient spikes, global gradient norm normalization, and periodic resetting of momentum statistics.

Experimental results on LLaMA-1B show that `Stable-SPAM` achieves the same target loss with half the training steps required by Adam and reduces perplexity by approximately 4 points under comparable settings, demonstrating its effectiveness in low-precision training.

Key Words: LLMs; LLM Pre-training; 4bit training; Machine Learning; Optimizer

[①]This work is based on the author's internship at UT-Austin and has been submitted to a double-blind peer-reviewed conference.

目录

第一章　绪论

第一节　研究背景和意义

近年来，随着大语言模型参数规模持续扩大，从数十亿到上千亿乃至万亿参数，其训练所需的计算资源和内存开销也呈指数级增长。这使得优化器作为模型训练中的核心组件，其计算效率与内存占用问题愈发成为瓶颈。其中，Adam 优化器由于其良好的收敛性和鲁棒性，依然是目前 LLM 训练中的主流选择。但 Adam 在实际应用中存在两个显著问题：（1）Adam 需要维护一阶和二阶动量，带来至少 2 倍额外内存开销；（2）Adam 对学习率等超参数较为敏感，尤其训练早期容易发生梯度爆炸或震荡收敛现象。目前已有的研究大多通过舍弃二阶动量[1]、替代二阶动量项[2-3]和引入模块化结构[4]等方法来缓解内存瓶颈或提升训练稳定性。除了上述优化器结构的改进，以低比特精度（如 FP8、FP4）存储和计算模型权重与梯度，能够显著减少内存带宽需求与能耗，在大模型预训练中具有广泛的应用潜力。

然而，低比特精度虽然显著降低了计算与内存开销，但也引入额外挑战：（1）梯度与状态量的动态范围受限：在低比特表示下，由于可表示数值范围限制缩小，梯度和动量等状态变量容易发生数值截断或译出，无法捕捉训练过程中的微小变化（2）参数更新误差的累积效应：相比于高精度训练，低比特表示会引入更大的舍入误差与量化噪声，这种误差在每一次迭代中可能被进一步放大，并通过优化器状态传播至整个训练过程，这些会极大影响优化器的稳定性与有效性，给低比特精度下大语言模型的预训练带来巨大挑战。

第二节　国内外研究进展

一方面，尽管已有多种内存高效的优化器架构被提出，如 Adafactor[2]、Galore[3] 和 Adam-mini[4]，但在低比特精度训练中，这些方法易受梯度爆炸与误差累积的影响，往往导致训练发散。另一方面，现有的低比特训练方法已取得一定进展[5-7]，但尚未有效结合上述高效优化器架构，因而整体性能仍有限。

一、主流内存高效优化器架构

为了降低大语言模型预训练中的计算与内存开销，学术界与工业界提出了多种优化器架构，能够在显著减少优化器状态存储的同时，仍保持良好的训练

性能。

Adam 系列优化器及其变体在内存效率与训练稳定性方面持续受到广泛关注，多个工作分别从不同维度提出了改进方法，以适应大语言模型在低精度训练中的资源限制与动态变化。

Shazeer et al.[2] 提出了 Adafactor 优化器，这是最早面向内存瓶颈问题的重要工作之一。该方法通过近似因式分解的方式，仅保留参数维度的均值平方统计量，而不再维护完整的二阶动量矩阵，从而将内存需求 从 $O(n^2)$ 降至 $O(n)$。此外，Adafactor 还引入了动态学习率缩放机制，支持无需额外调参即可适用于多种模型结构，成为许多大规模 Transformer 训练中的默认优化器。

Chen et al.[1] 在此基础上进一步提出了 Symbolic SGD，其核心思想是只保留一阶动量信息，同时利用符号化近似来预测梯度趋势，完全摒弃了二阶动量项，从而将优化器状态开销压缩至极限。这种方法不仅节省内存，而且由于其无状态的特性，还具备更强的跨步长迁移性，适合于参数频繁更新的分布式训练环境。

Zhang et al.[4] 提出了轻量级优化器 Adam-mini，通过将模型参数划分为若干块，并为每个块分配一个共享的学习率缩放因子，避免了对每个参数单独维护二阶动量项。该方法基于 Hessian 结构的统计特性进行块划分，使得学习率分配仍具备一定的自适应能力。实验证明，Adam-mini 能在保持与 AdamW 接近的收敛性能的同时，将内存占用降低约 50%，大幅提升了训练吞吐量。

Huang et al.[8] 针对模型训练中的不稳定性问题，提出了 SPAM 优化器。该方法引入了动量重置机制（Momentum Resetting），在梯度方向剧烈变化时自动清空累积动量，以防止错误方向的放大；同时提出了对尖峰梯度敏感的裁剪策略（SpikeClip），对可能引发训练震荡的异常梯度进行平滑处理。该优化器在 BF16 精度下在多个 LLM 基准测试中显著减少了损失函数突变现象。

Zhao et al.[9] 从更系统的视角出发，分析了多种优化器在 BF16 精度下的行为差异。该研究重点考察了学习率、动量系数等超参数对训练稳定性的影响，并指出许多优化器在低精度下对这些参数具有更高敏感性，难以迁移。在此基础上，作者提出了一套超参数调优基准和分析框架，为优化器的实用性评估提供了可靠依据

二、现有低比特精度训练

在低比特训练方向，近年来也涌现出多个重要进展，推动了从 BF16 向 FP8、甚至 FP4 的迁移。

Wang et al.[5] 首次提出面向大语言模型预训练的 FP4 精度训练框架，解决了极低比特精度下的数值不稳定问题。该方法设计了可微分的量化估计器（differentiable quantization estimator），提升了参数更新的精度，并引入了激活值的异常值截断与补偿机制（clamping and compensation）以防止激活崩溃。此外，框架结合了向量级量化和混合精度策略，最终在 13B 参数、百亿级 token 规模下，取得了接近 FP8 和 BF16 精度的性能表现，为未来 FP4 硬件铺平了算法基础。

Fishman et al.[6] 将 FP8 精度训练首次扩展至 2 万亿 token 级别，是此前工作的 20 倍，系统性揭示了长期训练中存在的稳定性问题。研究指出，SwiGLU 激活函数在长时间训练过程中会引发异常放大的现象，并通过理论与实证分析将其归因于权重对齐过程。为此，作者提出了 Smooth-SwiGLU 激活函数，以不改变函数本身行为的方式抑制不稳定性。同时，该工作还首次实现了 FP8 对 Adam 优化器一阶与二阶动量的量化，最终在 Intel Gaudi2 加速器上实现了最大 34% 的吞吐量提升。

Micikevicius et al.[7] 对 FP8 格式本身进行了系统设计与分析，提出了两种 8 比特浮点表示方式：E4M3（4 位指数，3 位尾数）与 E5M2（5 位指数，2 位尾数），分别面向推理与训练场景。文中详细评估了 FP8 在 CNN、RNN、Transformer 等主流神经网络结构中的适应性，并在超大规模（最多 175B 参数）模型训练中展示了与 16-bit 精度相当的性能表现。此外，作者还分析了 FP8 在后训练（post-training）量化中的优势，特别是在传统 INT8 难以覆盖的模型中展现出良好泛化能力。

第三节　本文的主要工作

尽管近年来针对大语言模型训练提出了多种优化器结构，如引入动量调整、梯度裁剪、自适应学习率等机制，在标准精度（如 FP16/BF16）设置下表现良好，但这些方法在更低比特精度（如 FP8/FP4）环境下往往难以直接迁移，常出现训练不稳定、性能退化甚至发散等问题。其根本原因在于现有优化器设计大多依赖较高数值精度以保持梯度估计与动量累积的准确性，缺乏对低比特数值噪声和状态约束的鲁棒性适配。

此外，目前的低比特训练方法主要关注量化精度本身的改进，较少系统性考虑如何在低精度环境中继承高精度优化器的关键优势，如自适应性与稳定性。因此，亟需一种兼顾低比特数值约束与优化器结构特性的训练策略，能够在保障效率的同时，显著提升模型在低比特精度下的收敛稳定性与最终性能。为此，本

文提出一种新颖的方法，在低比特训练环境中保留主流优化器的核心机制，有效缓解精度损失带来的训练不稳定问题，并实现更优的训练效果。

图 1.1 展示了在 C4 数据集上训练 LLaMA-130M、LLaMA-350M 与 LLaMA-1B 模型时，不同训练精度（包括 FP4、INT4、FP16 以及 BF16）与优化器设置下的验证集困惑度表现。图中横轴为学习率（Learning Rate），纵轴为最终验证损失，所有实验均采用 Adam 优化器（图 (1)）或在不同量化设置下进行训练（图 (2)-(4)）。其中，“Adam-BF16” 表示使用 Adam 优化器并在 BF16 精度下训练的结果，用作全精度参考基线。实验结果表明，在较低比特精度下（尤其是 INT4 与 FP4），优化器对训练稳定性和收敛性的影响显著，且不同模型规模下的敏感性也存在差异。

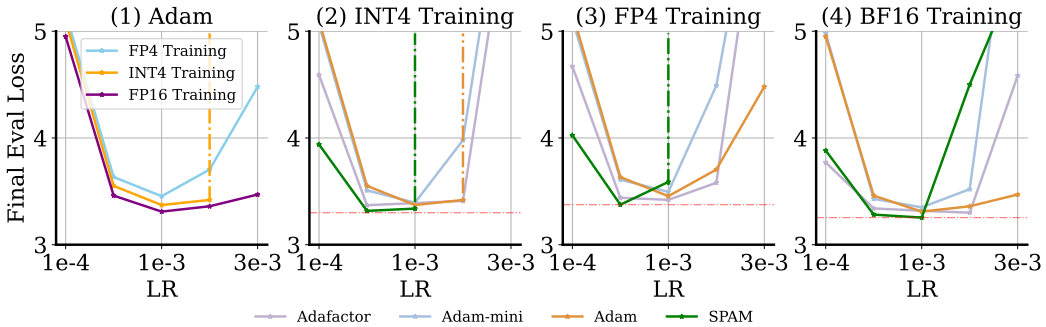

图 1.1 **图为在 C4 数据集上训练 LLaMA-130M 模型时，不同学习率下的最终验证损失。**虚线垂直线表示学习率再提升后模型无法继续训练（即训练损失变为 NaN）。红色虚线表示模型在各组实验中的最佳性能。

图 1.2 呈现了在 C4 数据集上训练 LLaMA-130M 模型时，采用 Adafactor、Adam-mini、Adam 以及 SPAM 等优化器，在不同学习率设定下的最终验证损失。横轴为归一化学习率，纵轴为验证损失。图中虚线垂直线表示对应优化器在更高学习率下将导致训练失败（损失为 NaN），表明该学习率为该优化器的稳定性上限。红色虚线标注了每组优化器实验中达到的最优性能位置。结果显示，不同优化器在低比特训练中的稳定性与收敛速度存在显著差异，其中 SPAM 优化器在多个学习率范围内均表现出更优的稳定性与最终性能。

本文对 Adam[10]、Adafactor[2]、Adam-mini[4] 和 SPAM[8] 等多种最新优化器进行了全面评估，重点分析其在权重和激活均为 4 比特精度的训练场景下，对学习率选择的有效性与鲁棒性。我们的研究揭示了以下几个关键发现：

所有评估优化器在 4 比特训练中对学习率更为敏感，尤其在大学习率下易发散，如图 1.1 所示。

1. SPAM 在各比特精度下均取得最低评估损失，但需精细调参；相比之下，Adafactor 对学习率更具鲁棒性，甚至优于 Adam。

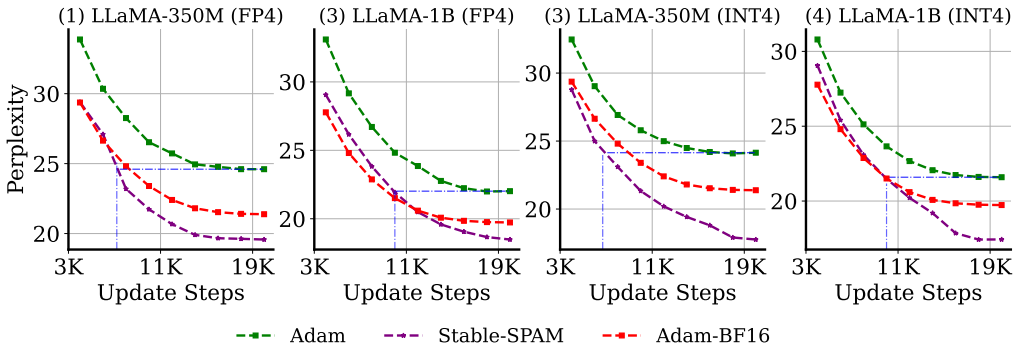

图 1.2 **4 比特大语言模型训练性能对比**。图中所示的 LLaMA-130M/350M/1B 模型均在 C4 数据集上进行训练。**Adam-BF16** 表示在 BF16 精度下使用 Adam 优化器进行训练。结果展示了各模型在验证集上的困惑度表现。

2. 4 比特训练的梯度范数远不如 BF16 稳定，频繁尖峰，易引发损失突增乃至高学习率下的发散。

3. 尽管 SPAM 中的 SpikeClip 缓解了 4 比特训练的梯度不稳，但仍难完全避免发散。

尽管 SPAM 对学习率较为敏感，但其在不同比特精度下均实现最低评估损失，体现出良好的优化潜力。基于此，本文提出 `Stable-SPAM`，旨在提升低精度大语言模型训练的稳定性。该方法在保留 SPAM 优势的同时（见表 4.5），显著增强了训练稳定性，代表了低比特优化的一项重要进展。

`Stable-SPAM` 在继承动量重置机制的基础上，进一步引入两项关键改进：（1）自适应尖峰感知裁剪（`AdaClip`），用于动态抑制异常梯度；（2）自适应梯度范数归一化（`AdaGN`），通过基于历史 l_2 统计对梯度整体进行归一化。实验与分析表明，上述机制可有效稳定 4 比特训练过程中的梯度行为，在性能上全面优于 Adam 与 SPAM。此外，使用 `Stable-SPAM` 训练的 4 比特 LLaMA-1B 模型不仅在准确率上超越了使用 Adam 训练的 BF16 版本，且在 4 比特设定下达到相同损失所需的训练步数仅为 Adam 的约一半。

第二章　4 比特训练稳定性分析

近期已有多项研究工作[8-9,11-12] 针对大语言模型（LLM）训练中的稳定性问题展开了探讨，涵盖诸如学习率不稳定、梯度尖峰（gradient spikes）以及损失突增（loss spikes）等现象。在本节中，我们在 4 比特 LLM 训练环境下进一步分析了不同优化算法的稳定性表现。

遵循 Zhao et al.[9], Wortsman et al.[11] 所描述的实验设置，我们使用从 1e-4 到 3e-3 范围内的一系列学习率来评估各优化器的最终性能。本次评估涵盖了两种广泛使用的优化器：Adam[10] 和 Adafactor[2]，以及两种近期提出的新方法：Adam-mini[4] 和 SPAM[8]。

此外，我们在整个 4 比特精度训练过程中持续监控全局梯度范数与训练损失。全局梯度范数定义如下：$\sqrt{\sum_{i=0}^{N} \|g_i\|_2^2}$ 其中 N 表示模型的层数，g_i 表示第 i 层的梯度。

所有实验均在 LLaMA-130M 和 350M 模型上，使用 C4 数据集进行，结果展示于图 1.1 和图 2.1 中。我们观察到如下现象：

一、低比特训练中的学习率敏感性问题

如图 1.1 所示，随着学习率的增大，4 比特精度下的训练表现出显著的评估损失上升趋势，而 BF16 精度下的训练在不同学习率下则相对稳定，性能波动较小。这一现象表明，4 比特精度训练对学习率更为敏感，训练稳定性显著下降。

二、低比特训练中的损失与梯度尖峰现象

图 2.1 展示了在 BF16 与 FP4（E1M2）精度设置下，LLaMA-130M 和 LLaMA-350M 模型在不同学习率条件下的训练损失与梯度范数变化曲线。可以观察到，BF16 精度训练过程整体较为平稳，而在 FP4 精度下，训练过程中频繁出现显著的损失尖峰。该现象在不同模型规模下均有发生，表明其具有普遍性。此外，损失尖峰通常伴随着梯度范数的剧烈波动，反映出模型训练过程中的显著不稳定性。

图 2.2 展示了 SpikeClip 方法在 4 比特大语言模型训练中的作用。左图反映了在应用 SpikeClip 之前与之后，梯度范数（Gradient Norm）的变化情况。可以观察到，在训练早期阶段，SpikeClip 有效抑制了异常尖峰的梯度值，显著缓解

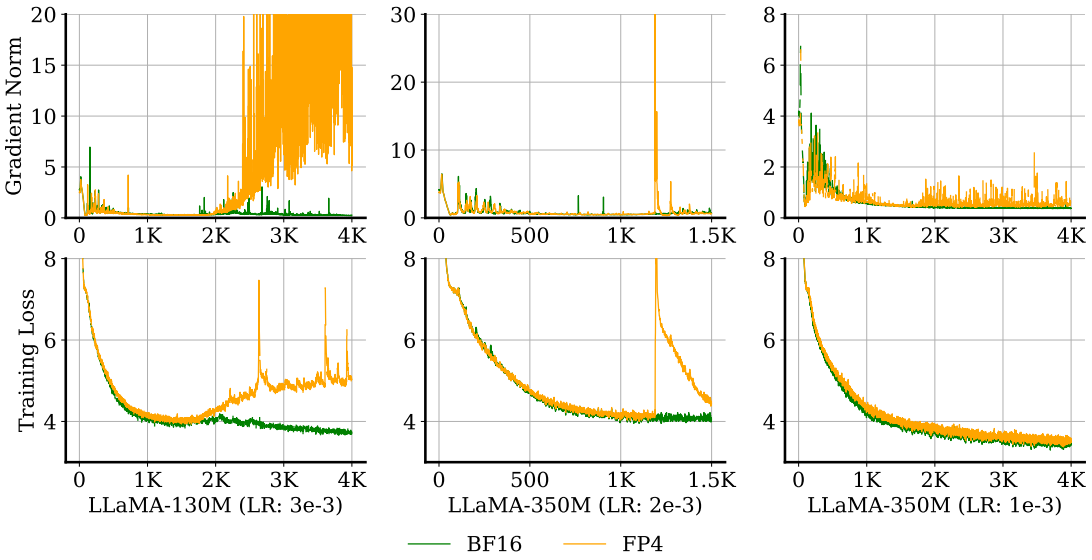

图 2.1　**BF16 与 FP4 精度下，Adam 优化器在不同学习率设置下的损失与梯度范数变化**。实验基于相同的训练配置，分别在 LLaMA-130M 和 LLaMA-350M 模型上进行。

了梯度爆炸问题。右图则比较了在使用与未使用 SpikeClip 的条件下，训练过程中的验证损失（Eval Loss）变化趋势。实验基于 LLaMA-130M 模型，采用 Adam 优化器，在 C4 数据集上进行训练。结果表明，引入 SpikeClip 有助于提升训练稳定性，使得模型更快收敛且最终验证损失更低。

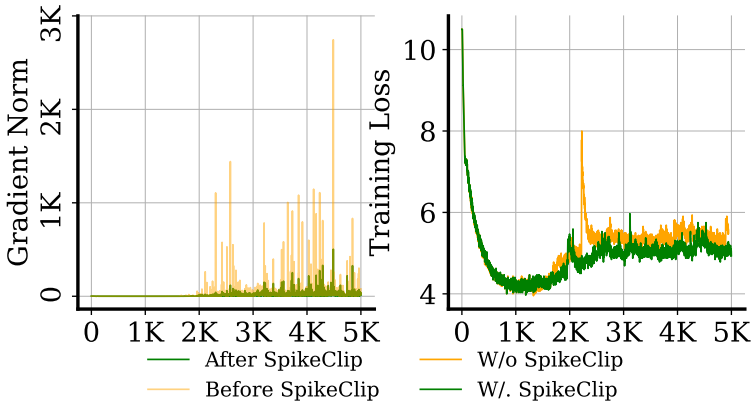

图 2.2　**SpikeClip[8] 对训练稳定性的影响**。左图展示应用梯度突变剪裁前后的梯度范数变化，右图对比使用与不使用该方法时的训练损失表现。实验基于 LLaMA-130M 模型，使用 Adam 优化器在 C4 数据集上进行训练。

三、SPAM 优化器在 4 比特训练中的性能与学习率敏感性分析

如图 1.1 所示，在 INT4 或 FP4 精度下，SPAM 优化器在使用最优学习率时，能够在多个优化器中取得最低的验证损失，展现出良好的性能表现。然而，随着学习率的升高，其验证损失迅速恶化，甚至在某些设定下发散为 NaN，显示出较强的学习率敏感性。

此外，我们监测了 SPAM 所引入的尖峰裁剪机制（SpikeClip）在训练过程中

的梯度范数与损失曲线变化。SpikeClip 通过检测梯度的二阶矩，识别并抑制异常梯度，从而提升训练稳定性，其具体数学表达如下：

$$g_i = \text{sign}(g_i) \cdot \sqrt{\theta V_i} \quad \text{满足} \quad \frac{g_i^2}{V_i} > \theta \tag{2.1}$$

其中，g_i 为第 i 个梯度分量，V_i 为对应的二阶动量，θ 为预定义阈值（其原始论文中默认使用 5000）。

本文实验证明，SpikeClip 在一定程度上可以缓解损失尖峰问题，但并不能完全阻止训练发散。一种可能的解释是，SpikeClip 是逐元素操作的，所使用的阈值可能偏高；若所有梯度分量同时整体升高，该机制仍可能保留较大的整体梯度范数。由于其仅关注个别异常值的裁剪，未能有效处理整体偏大的梯度情况。

这一问题也在图 2.2 中得到了验证，图中显示即便应用了 SpikeClip，梯度范数依然保持在较高水平。

第三章　Stable-SPAM 关键技术

为了解决 4 比特精度大语言模型（LLM）训练过程中的不稳定性问题，本文提出了一种稳定化的尖峰感知 Adam 优化器 —— `Stable-SPAM`。除继承自原始 SPAM 的动量重置机制外，`Stable-SPAM` 还引入了两项关键技术：自适应梯度范数归一化（`AdaGN`）与自适应尖峰感知裁剪（`AdaClip`）。这两项技术将在下文中进行详细说明。伪代码见附录 三。

第一节　自适应梯度范数归一化方法

如图 2.1 和图 2.2 所示，训练过程中损失的尖峰现象以及训练发散通常与梯度范数的剧烈上升同时出现，这与已有研究 Huang et al.[8], Takase et al.[12] 的发现一致。为应对这一问题，本文提出了 `AdaGN` 方法，通过基于梯度历史 l_2 范数统计信息对其进行自适应缩放，从而稳定训练过程中的梯度。

我们借鉴 Adam 优化器的思想，维护梯度范数的一阶与二阶动量的滑动平均值，以追踪其动态变化。具体计算过程如下：

$$g_{\text{norm}} = |g_t|^2 \tag{3.1}$$

$$m_{\text{norm}} = \gamma_1 \cdot m_{\text{norm}} + (1-\gamma_1) \cdot g_{\text{norm}} \tag{3.2}$$

$$v_{\text{norm}} = \gamma_2 \cdot v_{\text{norm}} + (1-\gamma_2) \cdot g_{\text{norm}}^2 \tag{3.3}$$

$$\hat{g}_t = \frac{g_t}{g_{\text{norm}}} \cdot \frac{m_{\text{norm}}}{\sqrt{v_{\text{norm}}} + \epsilon} \tag{3.4}$$

其中，$\hat{g}t$ 表示归一化后的梯度，γ_1 与 γ_2 分别为一阶与二阶动量的衰减系数，ϵ 为数值稳定性常量。通过将当前梯度 g_t 按照其历史均值 $\|m\|_{norm}$ 与平方均值 $\sqrt{v_{\text{norm}}}$ 之间的比例进行缩放，`AdaGN` 有效抑制了梯度范数的剧烈波动。

需要注意的是，由于梯度范数 g_{norm} 是针对每一层而言的一个标量，因此 `AdaGN` 引入的参数开销几乎可以忽略（每层仅需两个额外的标量参数）。

第二节　自适应尖峰感知机制

不同于 SPAM[8] 所采用的固定阈值尖峰裁剪方法，本文提出了自适应裁剪机制 `AdaClip`。其核心思想是：通过跟踪梯度分量的历史最大值，动态调整裁剪阈值，而非依赖预设的静态常数。

设 g_t 为第 t 步的梯度，我们首先计算出当前梯度张量中各分量绝对值的最大值 $g_{\max}$，再使用指数滑动平均更新裁剪阈值 $T_{\text{threshold}}$，并将所有超出该阈值的梯度分量按比例缩放。具体操作如下所示：

$$g_{\max} = \max_i(|g_t[i]|), \tag{3.5}$$

$$T_{\text{threshold}} = \gamma_3 \cdot T_{\text{threshold}} + (1-\gamma_3) \cdot g_{\max}, \tag{3.6}$$

$$\text{Mask}_{\text{spikes}} = (g_t > T_{\text{threshold}}), \tag{3.7}$$

$$g_t[\text{Mask}_{\text{spikes}}] = \frac{g_t[\text{Mask}_{\text{spikes}}]}{g_{\max}} \times T_{\text{threshold}}. \tag{3.8}$$

其中，$\gamma_3 \in [0,1]$ 控制滑动平均的响应速度。γ_3 越大，$T_{\text{threshold}}$ 对新的尖峰响应越慢，从而更新更平滑；反之则更新更快，适应性更强。

第三节　动量重置策略

与 Huang et al.[8] 一致，`Stable-SPAM` 采用动量重置机制（`MoRet`），周期性地重置 Adam 优化器中一阶与二阶动量的累计值。这一机制的作用在于消除尖峰梯度对动量历史造成的长效干扰。

由于 Adam 使用指数滑动平均来记录梯度历史信息，若某一步骤中出现极大的尖峰梯度，其数值将会在较长时间内持续影响优化器状态，甚至引发训练不稳定[8]。通过每隔固定步数 $\boldsymbol{\Delta T}$ 对动量项进行重置，`MoRet` 能有效消除异常梯度的累积影响，从而实现更加稳定、可靠的训练过程。

表 3.1　**在 C4 数据集上，不同优化器用于 LLaMA 模型的 INT4 与 FP4 训练的比较。**

	INT4 训练			**FP4 训练**		
	130M	**350M**	**1B**	**130M**	**350M**	**1B**
Adam	26.4	24.14	21.59	28.9	24.59	22.01
Adam+GradClip	26.30	21.64	19.74	28.27	20.84	20.25
Adafactor	25.11	20.45	20.65	26.89	20.53	20.03
SPAM	25.03	20.19	19.98	26.78	20.35	19.74
`Stable-SPAM`	**24.33**	**17.76**	**17.42**	**26.31**	**19.49**	**18.48**
Adam (BF16)	24.53	21.38	19.73	24.53	21.38	19.73
训练 Token 数	2.2B					

第四章　本文实验设置与结果分析

第一节　实验设置

为了验证所提出的 Stable-SPAM 的有效性，我们在 C4 数据集上对不同规模的 LLaMA 模型进行了大量实验。

一、基线方法

我们选取了五种流行的优化器作为基线方法，包括 Adam[10]、Adafactor[2]、Lion[1]、Adam-mini[4] 和 SPAM[8]。其中，Adam 和 Adafactor 是经过广泛验证和广泛使用的优化器，而 Adam-mini 和 SPAM 则是近期提出的新方法。此外，我们还将梯度裁剪（GradClip）[13] 与 Adam 结合作为额外的基线方法进行对比。

二、实验设置

参考 Zhao et al.[3], Lialin et al.[14]，我们在参数规模从 60M 到 1B 的 LLaMA 架构上进行训练。所有模型均采用 RMSNorm[15] 和 SwiGLU 激活函数[16]。对于每个模型规模，我们在各方法中保持相同的超参数配置，仅调整学习率。具体来说，我们对每个优化器的学习率从 1×10^{-4} 到 1×10^{-3} 进行网格搜索，步长为 2×10^{-4}。按照[8,12]中的设置，我们将 GradClip 基线的裁剪阈值设为 1。对于 Adafactor，我们采用原始论文[2]中的超参数设置，即 $\epsilon_1 = 10^{-30}$、$\epsilon_2 = 10^{-3}$ 和 $d = 1.0$。SPAM 的超参数配置参考[8]，将重置间隔设置为 500，学习率预热步数设置为 150，GSS 阈值设置为 5000。对于 Stable-SPAM，在 4 比特 LLM 训练中，我们设定 $\gamma_1 = 0.7$、$\gamma_2 = 0.9$、$\theta = 0.999$；在 BF16 训练中，则设定 $\gamma_1 = 0.85$、$\gamma_2 = 0.9999$ 和 $\gamma_3 = 0.999$。更详细的任务设置和超参数信息请参见附录 一。

第二节　实验结果

一、4 比特大语言模型训练结果分析

为评估 在 4 比特 LLM 训练中的表现，我们采用 FP4（E1M2: 1 位指数，2 位尾数）和 INT4（4 位整数）两种感知量化训练策略进行实验。C4 数据集上不同规模 LLaMA 模型的训练曲线如图 1.2 所示，最终困惑度结果汇总于表 3.1 中。

我们观察到 与 BF16 训练相比，4 比特训练会导致显著的性能下降。如表 3.1

所示，BF16（Adam）与 INT4/FP4（Adam）之间的困惑度差距在所有模型规模中均超过 1.5，凸显了低精度训练带来的挑战。图 1.2 表明，在 4 比特场景下，始终显著优于 Adam，甚至超过了 16 位 Adam 的表现。表 3.1 进一步展示了 优于其他先进优化器，如 Adafactor 和 SPAM。在所有基线方法中，引入 GradClip 有助于降低困惑度，而 Adafactor 与 SPAM 的表现均优于仅使用 GradClip 的方法。能在仅使用一半训练 Token 的情况下达到与 Adam 相同的性能。如图 1.2 所示，在大约一半训练步骤内达到了与 Adam 相同的困惑度。值得注意的是，在更大规模模型（如 LLaMA-350M 和 LLaMA-1B）上的表现尤为出色，展现出在大规模训练场景中的潜力。这可能是因为大模型的低精度训练更容易出现不稳定问题[17]，因此像 这样具备稳定特性的训练方法尤为受益。

二、极低精度训练结果

为评估 `Stable-SPAM` 在极低精度训练下的表现，我们在 LLaMA-350M 上进行了实验，采用了 A2W2（INT2）、A3W3（INT3）和 A4W4（INT4）配置。最终验证损失如图 4.1 所示。结果表明，在所有低精度设置中，`Stable-SPAM` 始终优于 Adam，并且在 INT3 训练下的表现甚至与 BF16-Adam 相当。

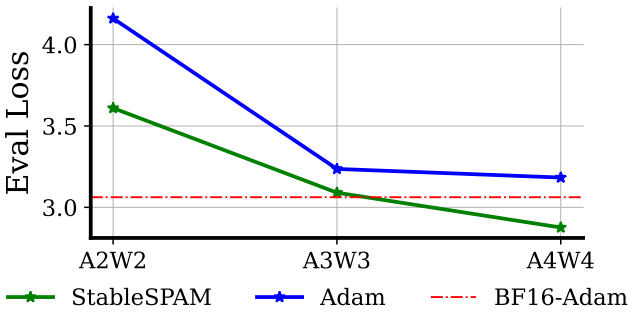

图 4.1　**图为 StableSPAM 在极低精度训练下的表现**。实验在 350M 模型上使用 C4 数据集进行训练。**BF16-Adam** 表示模型使用 Adam 优化器在 BF16 精度下进行训练。报告的是验证集上的最终损失。

三、BF16 精度下大语言模型的训练结果

为进一步评估 `Stable-SPAM` 的有效性，我们在多个不同规模的 LLaMA 模型上进行了标准 BF16 训练实验，实验基于 C4 数据集。训练曲线和最终困惑度分别如图 4.2 和 表 4.1 所示。表 4.1 显示，`Stable-SPAM` 在所有模型规模上都表现出优越的性能，显著超过排名第二的优化器。

此外，图 4.2 显示，在 LLaMA-350M 和 LLaMA-1B 上，`Stable-SPAM` 在仅使用一半甚至更少训练步数的情况下即可达到与 Adam 相同的性能，验证了

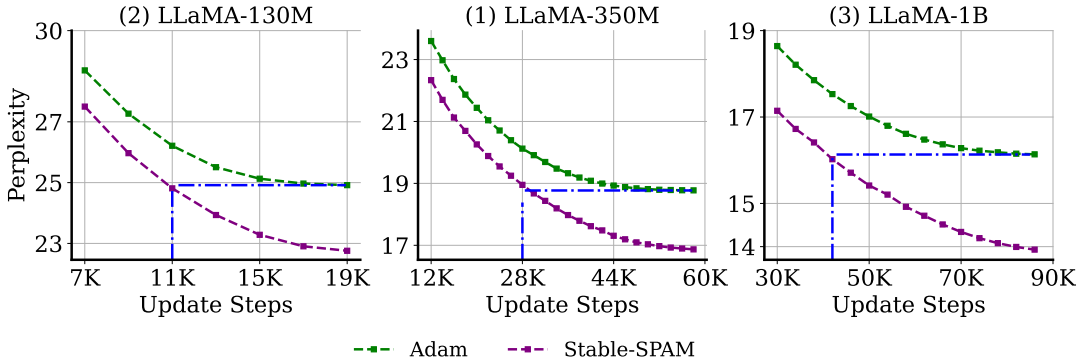

图 4.2 **不同模型规模下 BF16 精度训练的性能表现**。实验基于在 C4 数据集上训练的 LLaMA 模型进行。

Stable-SPAM 在 BF16 LLM 训练中能够以更少的训练 Token 达到相同的效果。上述结果表明，Stable-SPAM 的优势不仅适用于低精度训练，也适用于标准的 BF16 训练。

表 4.1 **BF16 精度下不同优化器的性能比较**。图中报告的指标为验证集上的困惑度（Perplexity）。

优化器	60M	130M	350M	1B
Adam-mini	34.10	24.85	19.05	16.07
Adam	34.09	24.91	18.77	16.13
Adam + GradClip	33.33	24.88	18.51	15.22
Adafactor	32.57	23.98	17.74	15.19
SPAM	30.46	23.36	17.42	14.66
Stable-SPAM	**28.84**	**22.21**	**16.85**	**13.90**
训练 Token 数	1.1B	2.2B	6.4B	11.6B

四、MoE 模型训练结果分析

Stable-SPAM 同样提升了 MoE 模型的训练稳定性。我们使用 Stable-SPAM 和 Adam 优化器，在 BF16 精度下训练了 OLMoE-1B-7B 模型[18] 共 3,000 步。下表的结果显示，Stable-SPAM 在整个训练过程中均显著优于 Adam，这表明 Stable-SPAM 有效增强了 MoE 的训练稳定性。

表 4.2 在总计 3,000 步更新下，OLMoE 模型的验证损失。Adam 和 Stable-SPAM 的学习率均设为 $\text{lr} = 1 \times 10^{-4}$。

优化器	第 1,000 步	第 2,000 步	第 3,000 步
Adam	5.43	4.73	4.65
Stable-SPAM	5.28	4.25	4.11

五、强化学习任务中的训练结果

Stable-SPAM 通过稳定异常梯度来改善训练过程，这一现象同样出现在非量化训练（例如 BF16）中。已有研究[19-20] 记录了 BF16 语言模型训练中的梯度和损失突增现象，这也解释了为何 Stable-SPAM 能优于 BF16 下的 Adam。为了进一步验证其在语言建模之外的普适性，我们在强化学习（MuJoCo）和时间序列预测（天气预报）任务中开展了额外实验，特别是在天气预测训练集中人为地引入了 10% 的异常数据以加剧训练不稳定性。如下面的表格所示，Stable-SPAM 在这些任务中依然稳定地优于 Adam，体现出其在多种任务中的广泛有效性。

表 4.3　在三个 MuJoCo 环境（HalfCheetah、Ant、Hopper）中的最终测试奖励

优化器	HalfCheetah	Ant	Hopper
Adam	5276.1 ± 1542.9	3835.6 ± 759.5	2447.5 ± 1037.9
Stable-SPAM	6762.6 ± 1414.2	4907.6 ± 954.6	3435.1 ± 1178.3

表 4.4　天气预测任务中的最终测试损失。报告的是 10 次重复实验的平均测试损失。异常数据通过向 10% 随机选取的输入值添加高斯噪声生成：$X = X + \mathcal{N}(0, \text{Severity} \times \max(X))$，其中 X 为输入，S 为严重程度。

优化器	$S = 0$	$S = 2$	$S = 5$
Adam	0.151	0.2003	0.346
Stable-SPAM	0.150	0.186	0.316

第三节　与现有优化器的集成实验

虽然 `AdaGN` 与 `AdaClip` 是专为 `Stable-SPAM` 设计的，但我们也探讨了它们是否可以与其他优化器兼容。为此，我们将 `AdaGN` 和 `AdaClip` 应用于两种近期提出的优化器：Lion[1] 和 Adam-mini[4]。我们分别对 Lion 与 Adam-mini 原始版本，以及与 `AdaGN` 和 `AdaClip` 组合的版本进行了对比实验，实验设定为 4 比特训练，模型为 LLaMA-60M 和 LLaMA-130M，数据集为 C4。

表 4.5 的结果显示，在 LLaMA-60M 与 130M 下，`AdaGN` 和 `AdaClip` 在 INT4 和 FP4 训练场景中均能稳定提升 Lion 与 Adam-mini 的性能。值得注意的是，在 LLaMA-130M 的 INT4 训练中，Lion 的困惑度提升高达 5.88，而在 LLaMA-60M 的 FP4 训练中，Adam-mini 的提升为 1.72。这些改进充分说明了 `AdaGN` 与 `AdaClip` 的通用性与有效性。

表 4.5 **AdaGN 与 AdaClip 在 Lion 与 Adam-mini 优化器上的表现**。实验基于 LLaMA-60M/130M 的 4 比特训练。

优化器	INT4 训练		FP4 训练	
	60M	**130M**	**60M**	**130M**
Lion	39.36	35.28	39.89	34.20
Lion+AdaGN+AdaClip	**38.49**	**29.40**	**36.75**	**31.63**
Adam-mini	34.84	29.79	36.37	32.95
Adam-mini+AdaGN+AdaClip	**34.61**	**29.65**	**34.65**	**32.39**
训练 Token 数	1.1B			

第四节 对训练稳定性的作用分析

为验证我们提出的 AdaGN 和 AdaClip 在提升 LLM 训练稳定性方面的有效性，我们进行了如下两方面的分析：

首先，我们比较了三种设置下的训练损失与梯度范数曲线：仅使用 Adam、使用 Adam + AdaGN、以及使用 Adam + AdaGN + AdaClip。实验采用 FP4 精度，在 LLaMA-130M 上进行，学习率为 3e-3。如图 4.3 所示，单独使用 Adam 时，训练损失存在发散趋势，梯度范数频繁震荡；加入 AdaGN 后，训练损失开始收敛，梯度范数显著减小；在此基础上引入 AdaClip，进一步平滑了梯度范数和损失曲线。

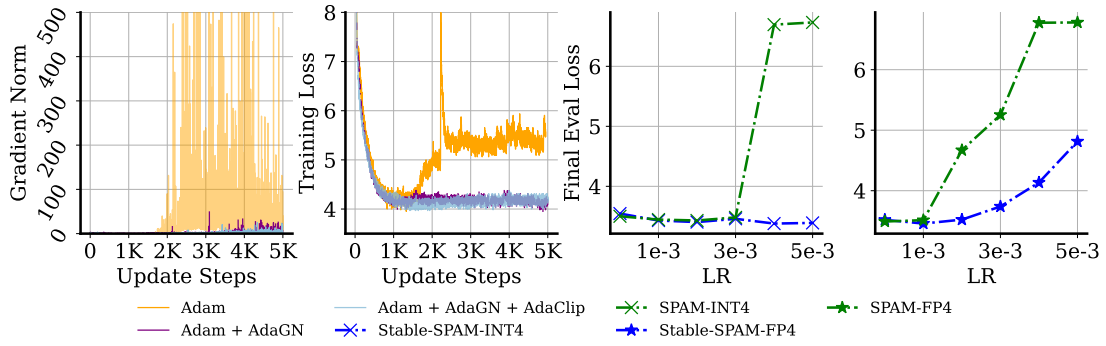

图 4.3 **AdaGN 与 AdaClip 对稳定 FP4 LLM 训练的作用**。左两图为 LLaMA-130M（学习率 = 3e-3），右两图为 LLaMA-60M。

其次，我们展示了在不同学习率（从 5×10^{-4} 到 5×10^{-3}）下的最终模型表现，实验基于 LLaMA-60M，分别采用 FP4 和 INT4 设置。如图 4.3 所示，Stable-SPAM 展现出更为平稳的性能曲线，说明其在不同学习率下依然具有较高稳定性。

上述结果充分表明，所提出的 AdaGN 与 AdaClip 方法能够显著提升 LLM 训练的稳定性与一致性。

第五节 消融实验设计与结果分析

为验证 Stable-SPAM 中三个组成部分——MoRet、AdaGN 和 AdaClip——的有效性，我们进行了全面的消融实验。具体来说，我们采用了两种方法:

(1) 我们将 MoRet、AdaGN 和 AdaClip 逐步引入到 Adam 优化器中，评估它们在 FP4 与 BF16 两种训练设置下的独立和组合性能提升;

(2) 我们将 AdaClip 替换为 SpikeClip[8]，将 AdaGN 替换为 GradClip[13]，进一步分析我们所提出组件的独特贡献。

表 4.6 总结了实验结果，主要观察如下:

MoRet 在 FP4 和 BF16 设置下均能稳定提升性能;

在 FP4 训练中，单独使用 AdaGN 提升有限，但与 AdaClip 结合后显著降低了最终困惑度;

相反，在 BF16 设置下，AdaGN 单独就能带来较大提升，加入 AdaClip 后收益较小。这种差异可能源于 FP4 训练中更频繁出现极端梯度尖峰现象，因而更需要 AdaClip 来有效修正偏移的更新方向。

最后，将 AdaClip 替换为 SpikeClip[8]、AdaGN 替换为 GradClip[13] 会导致困惑度上升，进一步验证了我们提出的 AdaGN 和 AdaClip 的有效性。

表 4.6 **Stable-SPAM 的消融实验**。实验基于 LLaMA-60M 和 C4 数据集。

优化器	FP4	BF16
Adam	35.47	34.09
Adam + MoRet	32.40	31.47
Adam + MoRet + AdaClip	31.97	30.29
Adam + MoRet + AdaGN	32.26	28.96
Adam + MoRet + AdaGN + AdaClip (**Stable-SPAM**)	**31.40**	**28.84**
Adam + MoRet+AdaGN+SpikeClip[8]	32.01	28.90
Adam + MoRet+GradClip[13]+AdaClip	31.95	29.87
Adam + MoRet+AdaGN+AdaClip (**Stable-SPAM**)	**31.40**	**28.84**
训练 Token 数	1.1B	

第六节 超参数敏感性分析

Stable-SPAM 引入了四个超参数: γ_1、γ_2、γ_3 和 ΔT，用于扩展 Adam 的功能。其中，γ_1 和 γ_2 类似于 Adam 中的 β_1 和 β_2，用于控制一阶矩 m_{norm} 和二阶

矩 v_{norm} 的平滑程度。较大的 γ_1 与 γ_2 会使更新更加平滑，更加依赖历史梯度范数的统计值来调节当前梯度范数。

γ_3 则用于决定识别尖峰梯度的阈值，其值越大，阈值变化越平滑、越保守，从而导致更多梯度被归类为尖峰梯度。

为了研究这些超参数的影响，我们绘制了在不同超参数值下的最终困惑度曲线：γ_1 从 0.5 到 0.9，γ_2 从 0.8 到 0.999，γ_3 从 0.9 到 0.999，ΔT 从 250 到 5000。实验基于 LLaMA-60M，在 C4 数据集上以 FP4 设置训练 1.1B Token。见图 4.4 。

如图 4.4 所示，过小或过大的超参数值都会导致性能下降。然而，这些超参数具有直观的解释，使得它们的调节过程相对简单，并且通常只需少量调整即可。在本论文中，我们采用 $\gamma_1 = 0.7$、$\gamma_2 = 0.9$、$\gamma_3 = 0.999$ 和 $\Delta T = 1000$ 作为默认配置，已在所有 4 比特训练任务中展现良好效果。

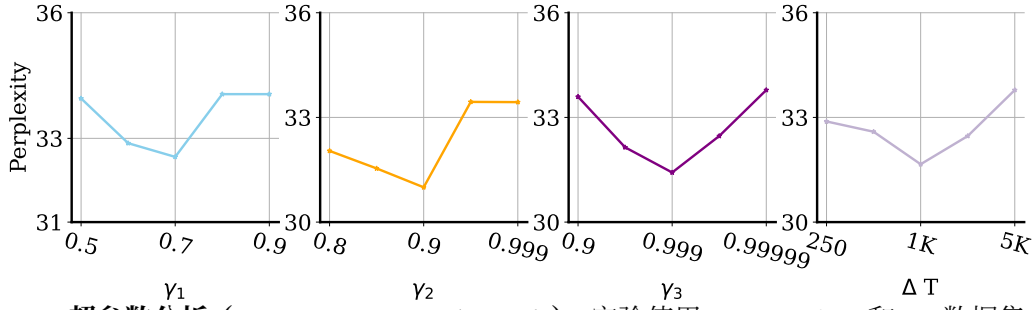

图 4.4 **超参数分析（Hyper-parameter Analysis）**。实验使用 LLaMA-60M 和 C4 数据集，在 FP4 训练设定下进行，总训练 Token 为 1.1B。

第五章　总结与相关工作

第一节　相关工作

大语言模型训练的不稳定性：大语言模型（LLM）训练过程中的不稳定性问题，常表现为损失突增（loss spike）和灾难性发散（catastrophic divergence）[21-22]，已引发大量关于训练稳定性技术的研究。这些方法大致可以分为三类：（1）梯度预处理（2）结构修改（3）初始化策略。

梯度预处理方法通常在优化初期通过缩放和裁剪梯度来稳定训练。例如，梯度裁剪（gradient clipping）[13] 是一种广为人知的技术，它将梯度范数整体缩放至一个固定值。随后，Adafactor[2] 提出将裁剪对象从原始梯度改为参数更新。近期，SPAM[8] 利用历史梯度统计信息检测并裁剪异常梯度。然而，这些方法普遍存在一个问题：需要手动设定阈值。

在结构层面，Xiong et al.[23] 发现 Post-LayerNorm（Post-LN）会放大梯度，在较大学习率下易引发不稳定，而 Pre-LayerNorm（Pre-LN）能够保持梯度范数，从而实现更稳定的训练。Embed LayerNorm（Embed LN）对嵌入层进行归一化[24]，但可能影响模型性能[25]；Embed Detach[26-27] 则通过截断梯度缓解损失突增。DeepNorm[28] 通过缩放残差连接稳定超深模型，而 αReparam[29] 则利用谱归一化参数化方式防止注意力熵塌缩。

初始化策略则为模型提供了补充的稳定性优势。Scaled Embed[12] 可稳定 LayerNorm 的梯度，而 Scaled Initialization[30] 通过 $\mathcal{N}(0, \sqrt{2/(5d)}/\sqrt{2N})$ 分布降低方差。Fixup[31-32] 则完全去除 LayerNorm，启发了无归一化（norm-free）架构。尽管这些方法不断得到改进，但训练稳定性仍是大语言模型发展的关键挑战之一。

低精度大模型训练：低精度训练[33-37] 近年来成为提升训练计算效率和内存效率的有力手段。其中，FP16[38] 与 BF16[39] 是当前最广泛使用的低精度格式。为了进一步提高效率，8 比特训练逐渐受到关注。例如，LM-FP8[40] 支持 FP8 精度训练，而[17] 指出，随着训练规模扩大（如超过 250B tokens），激活值中的异常值（activation outliers）问题愈发严重，挑战了低比特数据格式的表达范围。

为了解决这一问题，[17] 提出一种平滑策略，而[41] 则引入 Hadamard 变换来缓解激活异常值的影响。此外，数据格式的选择也对训练表现有重要影响。INT8 是目前最广泛支持的低精度格式，而 FP8 在 NVIDIA Hopper GPU 架构中得到了专门支持。MX 格式[42] 虽具有更强的表达能力，但目前硬件支持尚不广泛。

在本工作中，我们重点研究了低精度训练中存在的稳定性问题，并通过优化器设计提出了改进方法。我们的方法可与现有技术兼容，为提升低精度训练的稳定性提供了互补的解决方案。

第二节　结论

本文系统研究了大语言模型在 4 比特量化训练中面临的训练不稳定性问题。我们发现，尽管低精度训练大幅降低了内存和计算成本，但它也显著增强了对学习率的敏感性，并提高了梯度与损失突增的发生概率。

为了解决上述问题，我们提出了 Stable-SPAM 优化器，该方法融合了三项关键技术：AdaClip、AdaGN 和 MoRet。在不同规模的 LLaMA 模型上进行的实证研究表明，Stable-SPAM 不仅有效提升了 4 比特训练的稳定性，还在性能上优于现有优化器，甚至在某些情况下超过了 BF16 的训练效果。

此外，我们进一步验证了这些稳定化策略的广泛适用性，AdaClip 和 AdaGN 等组件在 Lion 和 Adam-mini 等优化器中同样展现出良好效果。

参考文献

[1] CHEN X, LIANG C, HUANG D, et al. Symbolic discovery of optimization algorithms[J]. Advances in neural information processing systems, 2024, 36: 1-13.

[2] SHAZEER N, STERN M. Adafactor: Adaptive learning rates with sublinear memory cost[C]//International Conference on Machine Learning. 2018: 4596-4604.

[3] ZHAO J, ZHANG Z, CHEN B, et al. Galore: Memory-efficient llm training by gradient low-rank projection[J]. arXiv preprint arXiv:2403.03507, 2024, abs/2403.03507: 1-10.

[4] ZHANG Y, CHEN C, LI Z, et al. Adam-mini: Use fewer learning rates to gain more[J]. arXiv preprint arXiv:2406.16793, 2024, abs/2406.16793: 1-10.

[5] WANG R, GONG Y, LIU X, et al. Optimizing Large Language Model Training Using FP4 Quantization[J]. arXiv preprint arXiv:2501.17116, 2025, abs/2501.17116: 1-33. DOI: 10.48550/arXiv.2501.17116.

[6] FISHMAN M, CHMIEL B, BANNER R, et al. Scaling FP8 training to trillion-token LLMs[J]. arXiv preprint arXiv:2409.12517, 2024, abs/2409.12517: 1-22. DOI: 10.48550/arXiv.2409.12517.

[7] MICIKEVICIUS P, STOSIC D, BURGESS N, et al. FP8 Formats for Deep Learning[J]. arXiv preprint arXiv:2209.05433, 2022, abs/2209.05433: 1-32. DOI: 10.48550/arXiv.2209.05433.

[8] HUANG T, ZHU Z, JIN G, et al. SPAM: Spike-Aware Adam with Momentum Reset for Stable LLM Training[J]. arXiv preprint arXiv:2501.06842, 2025, abs/2501.06842: 1-9.

[9] ZHAO R, MORWANI D, BRANDFONBRENER D, et al. Deconstructing what makes a good optimizer for language models[J]. arXiv preprint arXiv:2407.07972, 2024, abs/2407.07972: 1-8.

[10] KINGMA D P. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014, abs/1412.6980: 1-10.

[11] WORTSMAN M, LIU P J, XIAO L, et al. Small-scale proxies for large-scale transformer training instabilities[J]. arXiv preprint arXiv:2309.14322, 2023,

abs/2309.14322: 1-10.

[12] TAKASE S, KIYONO S, KOBAYASHI S, et al. Spike No More: Stabilizing the Pre-training of Large Language Models[J]. arXiv preprint arXiv:2312.16903, 2023, abs/2312.16903: 1-14.

[13] GOODFELLOW I. Deep learning[Z]. 2016.

[14] LIALIN V, MUCKATIRA S, SHIVAGUNDE N, et al. Relora: High-rank training through low-rank updates[C]//The Twelfth International Conference on Learning Representations. 2023.

[15] SHAZEER N. Glu variants improve transformer[J]. arXiv preprint arXiv:2002.05202, 2020, abs/2002.05202: 1-10.

[16] ZHANG B, SENNRICH R. Root mean square layer normalization[J]. Advances in Neural Information Processing Systems, 2019, 32: 789-798.

[17] FISHMAN M, CHMIEL B, BANNER R, et al. Scaling FP8 training to trillion-token LLMs[J]. arXiv preprint arXiv:2409.12517, 2024, abs/2409.12517: 1-13.

[18] MUENNIGHOFF N, SOLDAINI L, GROENEVELD D, et al. OLMoE: Open Mixture-of-Experts Language Models[EB/OL]. 2024. https://arxiv.org/abs/2409.02060. arXiv: 2409.02060 [cs.CL].

[19] A A, et al. On the Instability of BF16 Training in Language Models[J]. Unspecified Journal, 2023, 33: 1-34.

[20] B A, et al. Gradient Spikes in Mixed-Precision Training: An Empirical Study[J]. Unspecified Journal, 2024, 27: 1-27.

[21] CHOWDHERY A, NARANG S, DEVLIN J, et al. Palm: Scaling language modeling with pathways[J]. Journal of Machine Learning Research, 2023, 24(240): 1-113.

[22] MOLYBOG I, ALBERT P, CHEN M, et al. A theory on adam instability in large-scale machine learning[J]. arXiv preprint arXiv:2304.09871, 2023, abs/2304.09871: 1-10.

[23] XIONG R, YANG Y, HE D, et al. On layer normalization in the transformer architecture[C]//International Conference on Machine Learning. 2020: 10524-10533.

[24] DETTMERS T, LEWIS M, SHLEIFER S, et al. 8-bit optimizers via block-wise quantization[J]. arXiv preprint arXiv:2110.02861, 2021, abs/2110.02861: 1-15.

[25] SCAO T L, WANG T, HESSLOW D, et al. What language model to train if you have one million gpu hours?[J]. arXiv preprint arXiv:2210.15424, 2022, abs/2210.15424: 1-33.

[26] DING M, YANG Z, HONG W, et al. Cogview: Mastering text-to-image generation via transformers[J]. Advances in neural information processing systems, 2021, 34: 19822-19835.

[27] ZENG A, LIU X, DU Z, et al. Glm-130b: An open bilingual pre-trained model [J]. arXiv preprint arXiv:2210.02414, 2022, abs/2210.02414: 1-21.

[28] WANG H, MA S, DONG L, et al. Deepnet: Scaling transformers to 1,000 layers [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024, 46: 6761-6774.

[29] ZHAI S, LIKHOMANENKO T, LITTWIN E, et al. Stabilizing transformer training by preventing attention entropy collapse[C]//International Conference on Machine Learning. 2023: 40770-40803.

[30] NGUYEN T Q, SALAZAR J. Transformers without tears: Improving the normalization of self-attention[J]. arXiv preprint arXiv:1910.05895, 2019, abs/1910.05895: 1-13.

[31] ZHANG H, DAUPHIN Y N, MA T. Fixup initialization: Residual learning without normalization[J]. arXiv preprint arXiv:1901.09321, 2019, abs/1901.09321: 1-20.

[32] HUANG X S, PEREZ F, BA J, et al. Improving transformer optimization through better initialization[C]//International Conference on Machine Learning. 2020: 4475-4483.

[33] WANG N, CHOI J, BRAND D, et al. Training deep neural networks with 8-bit floating point numbers[J]. Advances in neural information processing systems, 2018, 31: 20-34.

[34] LIN J, ZHU L, CHEN W M, et al. On-device training under 256kb memory[J]. Advances in Neural Information Processing Systems, 2022, 35: 22941-22954.

[35] XI H, CAI H, ZHU L, et al. COAT: Compressing Optimizer states and Activation for Memory-Efficient FP8 Training[J]. arXiv preprint arXiv:2410.19313, 2024, abs/2410.19313: 1-12.

[36] XI H, CHEN Y, ZHAO K, et al. Jetfire: Efficient and accurate transformer

pretraining with int8 data flow and per-block quantization[J]. arXiv preprint arXiv:2403.12422, 2024, 2403.12422: 1-33.

[37] WORTSMAN M, DETTMERS T, ZETTLEMOYER L, et al. Stable and low-precision training for large-scale vision-language models[J]. Advances in Neural Information Processing Systems, 2023, 36: 10271-10298.

[38] MICIKEVICIUS P, NARANG S, ALBEN J, et al. Mixed precision training[J]. arXiv preprint arXiv:1710.03740, 2017, abs/1710.03740: 1-14.

[39] KALAMKAR D, MUDIGERE D, MELLEMPUDI N, et al. A study of BFLOAT16 for deep learning training[J]. arXiv preprint arXiv:1905.12322, 2019, abs/1905.12322: 1-13.

[40] PENG H, WU K, WEI Y, et al. Fp8-lm: Training fp8 large language models[J]. arXiv preprint arXiv:2310.18313, 2023, abs/2310.18313: 1-10.

[41] ASHKBOOS S, NIKDAN M, TABESH S, et al. HALO: Hadamard-Assisted Lossless Optimization for Efficient Low-Precision LLM Training and Fine-Tuning[J]. arXiv preprint arXiv:2501.02625, 2025, abs/2501.02625: 1-14.

[42] ROUHANI B D, ZHAO R, MORE A, et al. Microscaling data formats for deep learning[J]. arXiv preprint arXiv:2310.10537, 2023, abs/2310.10537: 1-15.

[43] NIE Y, NGUYEN N H, SINTHONG P, et al. A time series is worth 64 words: Long-term forecasting with transformers[J]. arXiv preprint arXiv:2211.14730, 2022, abs/2211.14730: 1-14.

附录 A　补充材料

第一节　架构与超参数

我们在本节中介绍用于 4 比特和 BF16 预训练的 LLaMA 模型架构和超参数配置，参考 Zhao et al.[3], Lialin et al.[14]。表 A.1 列出了不同模型规模下的主要超参数。

所有模型的最大序列长度均设为 256，批大小为 512，即总计 131K tokens。我们在所有实验中均采用 2000 步的学习率预热（warmup），并使用余弦退火（cosine annealing）策略将学习率衰减至初始值的 10%。

表 A.1　本文所使用的 LLaMA 模型配置。

参数量	隐藏层维度	中间层维度	注意力头数	层数
60M	512	1376	8	8
130M	768	2048	12	12
350M	1024	2736	16	24
1 B	2048	5461	24	32

对于每种模型规模（从 60M 到 1B），我们在 $1e{-}4$ 到 $1e{-}3$ 范围内以 2×10^{-4} 为步长调节学习率，最终通过验证集困惑度选择最优学习率。我们分别在表 A.2 和 表 A.3 中报告了 `Stable-SPAM` 在 4 比特和 BF16 训练下的详细超参数配置。

表 A.2　本文中 `Stable-SPAM` 在 4 比特预训练实验下的超参数配置。

超参数	LLaMA-130M	LLaMA-350M	LLaMA-1B
学习率（LR）	$1e-3$	$4e-4$	$2e-4$
$\boldsymbol{\Delta T}$	1000	1000	1000
γ_1	0.7	0.7	0.7
γ_2	0.9	0.9	0.9
γ_3	0.999	0.999	0.999

表 A.3　本文中 `Stable-SPAM` 在 BF16 预训练实验下的超参数配置。

超参数	LLaMA-60M	LLaMA-130M	LLaMA-350M	LLaMA-1B
标准预训练（Standard Pretraining）				
学习率（LR）	$1e-3$	$8e-4$	$4e-4$	$2e-4$
$\boldsymbol{\Delta T}$	1000	1000	1000	1000
γ_1	0.85	0.85	0.85	0.85
γ_2	0.99999	0.99999	0.99999	0.99999
γ_3	0.999	0.999	0.999	0.999

第二节　时间序列预测任务

我们在时间序列预测任务上进行了额外实验。为了模拟梯度异常现象，我们在数据中以 10% 的概率引入异常点。实验在 Weather 时间序列数据集 ① 上进行，采用 PatchTST[43] 模型，重复运行 10 次，实验结果如图 A.1 所示。

实验结果显示，随着异常数据强度（S）的增加，`Stable-SPAM` 相对于 Adam 的性能优势愈发显著；此外，`Stable-SPAM` 在所有设置中均优于 SPAM，进一步验证了我们方法的稳定性和有效性。

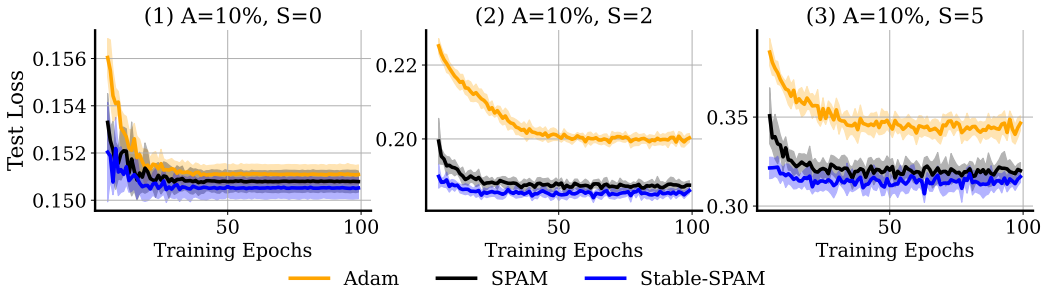

图 A.1　**天气时间序列数据集训练过程中的测试损失**。异常数据通过在 10% 随机选取的输入上添加高斯噪声生成，具体形式为：$X = X + \texttt{Gaussin}(0, S \cdot \texttt{Max}(X))$，其中 X 为输入，S 为异常强度。

① https://www.bgc-jena.mpg.de/wetter/

第三节　伪代码

算法 A.1 展示了 Stable-SPAM 优化器的伪代码实现。

算法 A.1 Stable-SPAM

Input: A layer weight matrix $w \in \mathbb{R}^{m \times n}$, learning rate α, decay rates $\beta_1 = 0.9$, $\beta_2 = 0.999$, initial parameters w_0, $\gamma_1 = 0.7, \gamma_2 = 0.9$ for AdaGN and $\gamma_3 = 0.999$ for AdaClip, momentum reset interval ΔT, small constant $\epsilon = 1 \times 10^{-6}$, and total training steps T.

Output: Optimized parameters w_T.

while $t < T$ **do**

  $g_t \in \mathbb{R}^{m \times n} \leftarrow -\nabla_w \phi_t(w_t)$ // Gradient of the objective at step t.

  $g_{\max} \leftarrow \mathrm{Max}(\mathrm{abs}(g_t))$

  $T_{\text{threshold}} \leftarrow T_{\text{threshold}} \cdot \theta + (1-\theta)\, g_{\max}$

  $\hat{T}_{\text{threshold}} \leftarrow \dfrac{T_{\text{threshold}}}{1-\theta^t}$ // Bias correction for threshold

  $\text{Mask}_{\text{spikes}} \leftarrow \left(\mathrm{abs}(g_t) > \hat{T}_{\text{threshold}}\right)$

  if $sum\left(Mask_{\text{spikes}}\right) > 0$ **then**

    $g_t[\text{Mask}_{\text{spikes}}] \leftarrow \dfrac{g_t[\text{Mask}_{\text{spikes}}]}{g_{\max}} \times \hat{T}_{\text{threshold}}$

  end

  $g_{\text{norm}} \leftarrow \|g_t\|_2$

  $m_{\text{norm}} \leftarrow \gamma_1\, m_{\text{norm}} + (1-\gamma_1)\, g_{\text{norm}}$

  $v_{\text{norm}} \leftarrow \gamma_2\, v_{\text{norm}} + (1-\gamma_2)\, g_{\text{norm}}^2$

  $\hat{m}_{\text{norm}} \leftarrow \dfrac{m_{\text{norm}}}{1-\gamma_1^t}, \hat{v}_{\text{norm}} \leftarrow \dfrac{v_{\text{norm}}}{1-\gamma_2^t}$ // Bias-corrected norm estimates

  $\text{adaptive\_norm} \leftarrow \dfrac{\hat{m}_{\text{norm}}}{\sqrt{\hat{v}_{\text{norm}}} + \epsilon}$

  $g_t \leftarrow \dfrac{g_t}{g_{\text{norm}}} \times \text{adaptive\_norm}$

  if $(Mod(t,\ \Delta T) = 0)$ **then**

    $\boldsymbol{m} \leftarrow \mathrm{zeros\_like}(\boldsymbol{m})$

    $\boldsymbol{v} \leftarrow \mathrm{zeros\_like}(\boldsymbol{v})$

  end

  $\boldsymbol{m}_t \leftarrow \beta_1\, \boldsymbol{m}_{t-1} + (1-\beta_1)\, g_t$

  $\boldsymbol{v}_t \leftarrow \beta_2\, \boldsymbol{v}_{t-1} + (1-\beta_2)\, g_t^2$

  $\hat{\boldsymbol{m}}_t \leftarrow \dfrac{\boldsymbol{m}_t}{1-\beta_1^t}$ // bias correction

  $\hat{\boldsymbol{v}}_t \leftarrow \dfrac{\boldsymbol{v}_t}{1-\beta_2^t}$ // bias correction

  $w_t \leftarrow w_{t-1} - \alpha\, \dfrac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon}$

  $t \leftarrow t+1$

end

return w_T.

第四节　符号说明

一、数学符号

表 A.4　本文数学符号对照表

符号	说明
$\mathbf{g}_t$	第 t 步的梯度向量
$\mathbf{m}_t$	一阶动量估计（梯度指数移动平均）
$\mathbf{v}_t$	二阶动量估计（梯度平方指数移动平均）
$\hat{\mathbf{m}}_t$	偏差校正后的一阶动量
$\hat{\mathbf{v}}_t$	偏差校正后的二阶动量
η	学习率（step size）
β_1, β_2	动量衰减系数
ϵ	数值稳定常数
$\|\cdot\|_2$	ℓ_2 范数
$\|\mathbf{g}_t\|_2$	第 t 步梯度的 ℓ_2 范数
c_t	第 t 步自适应梯度裁剪阈值
t	训练迭代步数
b	Block 大小（block size）
L	序列长度（sequence length）
D	隐向量/嵌入维度（hidden size）
N	模型参数总数
PPL	困惑度指标（Perplexity）

二、本文缩写与术语

表 A.5　本文缩写与术语

缩写	说明
LLM	Large Language Model，大语言模型
FP16	16 位浮点格式（1 符号位 + 5 指数位 + 10 尾数位）
BF16	Brain Floating Point 16，兼容 IEEE-754 的 16 位格式
FP8 / FP4	8 位 / 4 位浮点格式（文中用 E4M3/E5M2 与 E2M1/E3M0 表示）
INT4	4 位定点整数格式
Adam	Adaptive Moment Estimation 优化器
Adafactor	近似分解二阶动量的内存高效 Adam 变体
Adam-mini	块级共享二阶动量的轻量化 Adam 变体
SPAM	Spike-aware Momentum 优化器（含 SpikeClip 与 Momentum Reset）
Stable-SPAM	本文提出的改进型 SPAM，用于 4 比特稳定训练
GLU / SwiGLU	Gated Linear Unit 与其 Swish 变体激活函数
C4	Colossal Clean Crawled Corpus 数据集
MoE	Mixture of Experts 模型架构

致谢

谨以此文感谢四年来遇到的最好的你们。

首先，我衷心感谢科大的金西老师！感谢您一直以来的悉心指导与支持！正因为有您的帮助，我才能顺利完成学业并顺利毕业。

感谢我在 UCSD 暑期科研期间遇到的丁雨霏老师，您全心投入的支持与富有洞见的指导，是我科研旅程中难得的宝贵经历；感谢 UT Austin 的汪张扬老师，是一次很特别的机会让我深入大语言模型这个领域；感谢 UCI 实习期间的黄思陶老师，和您合作的那段时间让我慢慢找到了真正喜欢的研究方向。感谢东南大学的潘存华老师，在您的指导下我第一次接触并迈入了科研的大门。感谢北大林亦波老师在研究过程中给予的细致指导。同时也感谢所有在项目合作帮助过我的师兄、合作者：振宇、钟凯、tianjin、keyi、haocheng... 在此我还想感谢郅琛学长，在我准备转专业、出国最迷茫的时候，毫无保留全心全意地帮我想办法。感谢我的班主任王老师、侯老师，在生活中给予了我巨大帮助！衷心祝愿每一位老师和同学未来顺顺利利、每天都有好心情，愿大家都能过上自己喜欢的生活，收获属于自己的小确幸与精彩！

感谢科大四年来一路相伴的好友们，一起度过的快乐瞬间将永远留在我的记忆里。无论我们今后身处何方，愿我们前程似锦，万里无忧！

感谢路过我青春的所有人们。

感谢你[①]曾来过。

我无数遍感谢他人的爱意和真诚，让我在很多时刻都得到了救赎。开心的日子是闪着光的。无论同行的人现在去哪了，都不应该被删除被遗忘。

我想由衷地感谢我的爸爸妈妈。感谢你们为我倾注的全部心血，给予我生命、陪伴我成长。我会永远铭记你们的深情厚爱，愿你们健康长寿，平安喜乐。

最后，我想由衷的感谢自己！祝小胡健健康康，快快乐乐，万事顺遂！记得给自己买喜欢的橘子和三明治！（tunafish 或者 avocado 都挺不错）

2025 年 4 月

[①] Beep Beep AMX

在读期间取得的科研成果

已发表论文

1. H. Xu, **H. Hu**, S. Huang, *Optimizing High-Level Synthesis Designs with Retrieval-Augmented Large Language Models*, in **Proc. IEEE LLM Aided Design Workshop (LAD)**, pp. 1–5, 2024.
2. **H. Hu**, K. Zhong, C. Pan, X. Xiao, *Ambiguity function shaping via manifold optimization embedding with momentum*, **IEEE Communications Letters**, vol. 27, no. 10, pp. 2727–2731, 2023.
3. T. Huang*, **H. Hu***, Z. Zhang*, G. Jin, X. Li, L. Shen, T. Chen, L. Liu, Q. Wen, *et al.*, *Stable-SPAM: How to Train in 4-Bit More Stably than 16-Bit Adam*, in **Proc. International Conference on Learning Representations (ICLR) SCOPE Workshop**, 2025.
4. H. Zhang, Y. Xu, **H. Hu**, K. Yin, H. Shapourian, J. Zhao, R. R. Kompella, *et al.*, *Optimizing Quantum Communication for Quantum Data Centers with Reconfigurable Networks*, in **Proc. International Symposium on Computer Architecture (ISCA)**, 2025, pp. 739–752.
5. K. Zhong, J. Hu, Z. Zhao, X. Yu, G. Cui, B. Liao, **H. Hu**, *MIMO radar unimodular waveform design with learned complex circle manifold network*, **IEEE Transactions on Aerospace and Electronic Systems**, vol. 60, no. 2, pp. 1798–1807, 2024.
6. K. Zhong, J. Hu, Y. Cong, G. Cui, **H. Hu**, *RMOCG: A Riemannian manifold optimization-based conjugate gradient method for phase-only beamforming synthesis*, **IEEE Antennas and Wireless Propagation Letters**, vol. 21, no. 8, pp. 1625–1629, 2022.