SENTINEL：面向流水线并行去中心化训练的分阶段完整性验证

Hadi Mohaghegh Dolatabadi、Thalaiyasingam Ajanthan、Sameera Ramasinghe、Chamin P Hewa Koneputugodage、Gil Avraham、Yan Zuo、Violetta Shevchenko 和 Alexander Long Pluralis Research

arXiv:2603.03592v1 [cs.DC] 3 Mar 2026

在不可信的地理分布式节点上执行去中心化训练会带来严重的安全风险。现有拜占庭容错文献通过鲁棒聚合方法应对数据并行（DP）训练问题，但流水线并行（PP）面临着本质上不同的挑战。在 PP 中，模型层分布在各工作节点上，激活值及其梯度在阶段之间传递，而非进行聚合，因此传统 DP 方法并不适用。我们提出 SENTINEL，这是一种用于 PP 训练的验证机制，无需重复计算。SENTINEL 使用基于指数移动平均值（EMA）的轻量级动量监控，以检测阶段间通信是否遭到篡改。与现有针对 DP 的拜占庭容错方法在各副本之间聚合参数梯度不同，我们的方法验证层与层之间按顺序传输的激活值/梯度。我们针对这一新场景给出了理论收敛保证；当其放宽至标准训练时，可恢复经典收敛速率。实验表明，在最多 176 个工作节点组成的不可信分布式环境中，我们成功训练了参数量高达 40 亿的大语言模型，同时保持了模型收敛性和性能。

关键词：流水线并行、去中心化训练、对抗式机器学习

1. 引言

大语言模型（LLM）从根本上重塑了人工智能，在各种任务中展现出卓越性能（Bi 等，2024；DeepSeek-AI 等，2025；Dubey 等，2024；Jiang 等，2024；MetaAI，2025；OpenAI，2023；Yang 等，2024）。然而，训练最先进的大语言模型需要大量计算资源（据报道，GPT-4（Walker II，2023）、Llama-4（MetaAI，2025）、Qwen2.5（Yang 等，2024）等模型需要数万台集中部署的图形处理器），并伴随相应的能源和财务成本。这推动了去中心化训练方法的研究，以扩大参与大语言模型开发的范围（Ryabinin 等，2021；2023；Yuan 等，2022）。去中心化训练将分布式训练扩展到无信任环境，使独立协作者能够汇集各自的计算资源（可能相距甚远），从而开发模型，而无需依赖庞大的集中式基础设施。

通过两种主要的并行化方法——数据并行与流水线并行，可以在互联设备组成的网络上实现大语言模型的去中心化训练。数据并行（DP）（Li 等人，2020；Zhao等人，2023年）将不同批次的训练数据分配给各工作节点，但要求每个节点都能容纳整个模型，这对于去中心化环境中的十亿参数模型并不现实。流水线并行（PP）（Huang等人，2019；Krizhevsky等人，2017年）将模型按阶段划分到各工作节点，每个节点负责不同的模型阶段（层组），但这需要高带宽连接，而且容易受到节点掉线的影响。结合这两种互补方法，既能缓解数据并行的规模限制，也能降低流水线并行的脆弱性，并促成了 SWARM 等框架（Ryabinin 等人，2023年），使其能够通过分布式节点之间的互联网规模通信来训练十亿参数大语言模型。借助这些并行化技术，这类框架旨在提高节点利用率，同时尽可能降低带宽要求，从而让更多人能够进行大规模模型训练。

通讯作者：*hadi@pluralis.ai*
© 2026 Pluralis Research。保留所有权利。

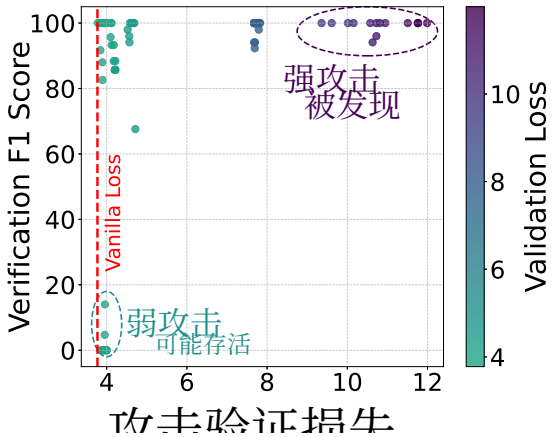


图 1：75 种以上不同攻击设置下 F1 分数与验证损失的散点图。强攻击（损失较高）更容易被检测到（F1 分数较高），而弱攻击可能漏网。因此，我们的验证方法能够检测出危害最大的攻击，避免其干扰训练。

尽管去中心化训练研究的重点一直是优化通信带宽和容错能力（Ajanthan et al.，2025年；Douillard et al.，2023年；2025年；Ryabinin 等人，2021年），激励驱动的去中心化训练能否成功，关键取决于参与节点的完整性与可信度（Lu 等人，2024）。在数据并行配置中，恶意行为者可以通过参数梯度投毒破坏全局更新；而在流水线并行（逐层模型并行）中，对手则可以破坏模型阶段之间的中间激活值或激活梯度。这些漏洞凸显了在去中心化的流水线并行训练环境中采用鲁棒机制的必要性。传统的拜占庭容错方法旨在应对较简单的数据并行威胁模型（Gorbunov 等人，2022；Malinovskyet al.，2024；Mhamdi 等人，2018年），却无法应对流水线并行中由分区模型执行引发的级联故障（Lu 等人，2024）。

本文首次全面探索了安全且可验证的去中心化流水线并行训练，识别并解决了此类设置所特有的漏洞。我们针对这一设置，系统地形式化定义了一系列恶意攻击；对于这些攻击，传统的基于检查点的验证并不奏效（Arun 等人，2025年）。为应对这一问题，我们提出了一种轻量级验证机制，利用验证器节点——部署在各阶段之间的可信中介——持续监控计算完整性，同时无需完整复制模型，也不会影响训练吞吐量。

我们的方法称为 SENTINEL，它实现了一种基于动量的异常检测系统，用于跟踪流水线各阶段的激活值和梯度的指数移动平均值（EMA）。在模型的每个阶段，验证器节点都会计算观测信号与其指数移动平均基线之间的统计散度指标。通过四分位距（IQR）分析确定的自适应校准阈值，可用于标记超过阈值、可能涉及恶意活动的偏差。这种轻量级验证机制带来的计算开销极小，同时能够及早检测梯度篡改攻击和激活篡改攻击。实证评估表明，即使存在恶意参与者，我们的系统仍能在规模扩展至数百个工作节点以上的去中心化流水线并行设置中，成功检测并缓解多种攻击，同时维持训练完整性和收敛稳定性。

我们的工作的贡献总结如下：

- 我们首次全面研究了采用混合数据-流水线并行的去中心化训练所特有的漏洞，并引入了一套训练中断攻击，作为评估未来系统安全性的基准测试。
- 我们提出一种轻量级验证方法，称为 SENTINEL，该方法在验证节点利用基于动量的监控。我们的理论分析表明，未被检测到的恶意工作节点对收敛性质的影响可忽略不计（见图 1）。
- 我们在涉及数百个工作节点的分布式环境中进行了大量实验，验证了

我们的验证框架在真实的去中心化训练场景中缓解恶意行为的有效性，其 F1 分数始终保持较高水平（$> 90\%$）。

- 我们将该方法与 SWARM 并行相结合，并通过在一个包含 128 个地理分布式实例的系统上进行训练，展示了其在现实世界去中心化训练生态系统中的出色通用性。

2. 问题陈述

本节概述用于去中心化训练的混合 DP-PP 架构和威胁模型，重点关注恶意工作节点行为。更多漏洞详见第 B 节。

2.1 威胁模型

我们考虑一个由 p 个阶段和 n 个工作节点组成的分布式流水线并行神经网络（见图 2b 和 2c）。该网络输出 $\boldsymbol{y} = F(\boldsymbol{x}) = f_p \circ f_{p-1} \circ \cdots \circ f_1(\boldsymbol{x})$。在第 t 次迭代时，参数为 $\boldsymbol{\theta}_t = (\boldsymbol{\theta}_t^{(1)}, \ldots, \boldsymbol{\theta}_t^{(p)}) \in \mathbb{R}^{D_{\text{total}}}$，其中每个阶段 s 包含参数 $\boldsymbol{\theta}_t^{(s)} \in \mathbb{R}^{D_s}$ 和 $D_{\text{total}} = \sum_{s=1}^{p} D_s$。每个阶段 s 均复制到并行运行的 d_s 个工作节点上，因此 $n = \sum_{s=1}^{p} d_s$。参数 $\boldsymbol{\theta}^{(s)}$在各阶段副本之间共享，但每个副本处理不同的小批次。因此，工作节点 (s, r) 在前向传播过程中计算副本特定的激活值 $\boldsymbol{h}^{(s,r)} = f_s(\boldsymbol{h}^{(s-1,r)}; \boldsymbol{\theta}^{(s)}) \in \mathbb{R}^m$，并在反向传播过程中计算副本特定的激活梯度 $\boldsymbol{g}^{(s,r)} = \nabla_{\boldsymbol{h}^{(s-1,r)}} \mathcal{L}(\boldsymbol{\theta})$，其中 $\mathcal{L}(\boldsymbol{\theta})$ 是训练损失。

在当前的去中心化流水线并行训练框架中，例如 SWARM（Ryabinin 等人，2023年），工作节点在流水线阶段之间交换激活值和激活梯度，却不进行验证。与联邦学习（McMahan 等人，2017年）不同，后者的攻击主要针对权重梯度投毒；流水线并行设置容易遭受训练中断攻击，恶意工作节点可以通过发送损坏信号，在不被察觉的情况下扰乱训练。这尤其棘手，因为早期阶段工作节点中的损坏可能只有到后续层的工作节点才会显现：由于模型非线性，错误可能不断放大，并且只有经过多个阶段后才会超过检测阈值，使攻击者得以逃避检测，还可能因这种级联效应而将诚实工作节点误判为恶意节点。关于数据并行与流水线并行设置的全面比较，请参阅第A节中的 Q0。

为应对这些漏洞，我们引入专用的“验证器节点”作为可信中介，进行轻量级验证（见图2b）。这些验证器节点会拦截并验证阶段之间交换的所有训练信号，即使在 CPU 硬件上运行，也能以极低开销高效完成验证。它们的主要优势在于能够将恶意行为定位到具体阶段，从而防止流水线并行固有的级联损坏效应。在 SWARM（Ryabinin 等人，2023年）等框架中，验证器节点只需对负责协调工作节点的“训练器节点”进行简单扩展即可（见图2c，第G节）。形式上，我们对验证器节点所采用的威胁模型定义如下：

定义 2.1（数据并行与流水线并行威胁模型）。 考虑一个在分布式系统中按照上述流水线并行设置进行训练的神经网络。我们将可信的“验证器”节点置于连续阶段之间，所有通信都经由这些节点路由，因而可以被验证，我们的目标是检测并排除恶意参与者。具体而言，令 $B_s \subset \{1, 2, \ldots, d_s\}$ 为阶段s处的恶意工作节点子集，恶意工作节点所占比例定义为 $\gamma_s = |B_s|/d_s$，并且我们假设 $\gamma_s < 1/2$。我们的目标是检测恶意子集 B_s。

在我们的威胁模型中，恶意行为者可以采用各种策略来破坏训练。[1] 接下来，我们将根据这些攻击的计算需求及其对训练的潜在影响来介绍它们。

[1] 我们假设 s 第一层和最后一层由诚实工作节点管理，因为它们处理用户数据并计算损失，因需要保护。 我们还假设恶意工作节点之间无法相互串通，但之后会放宽这一假设。

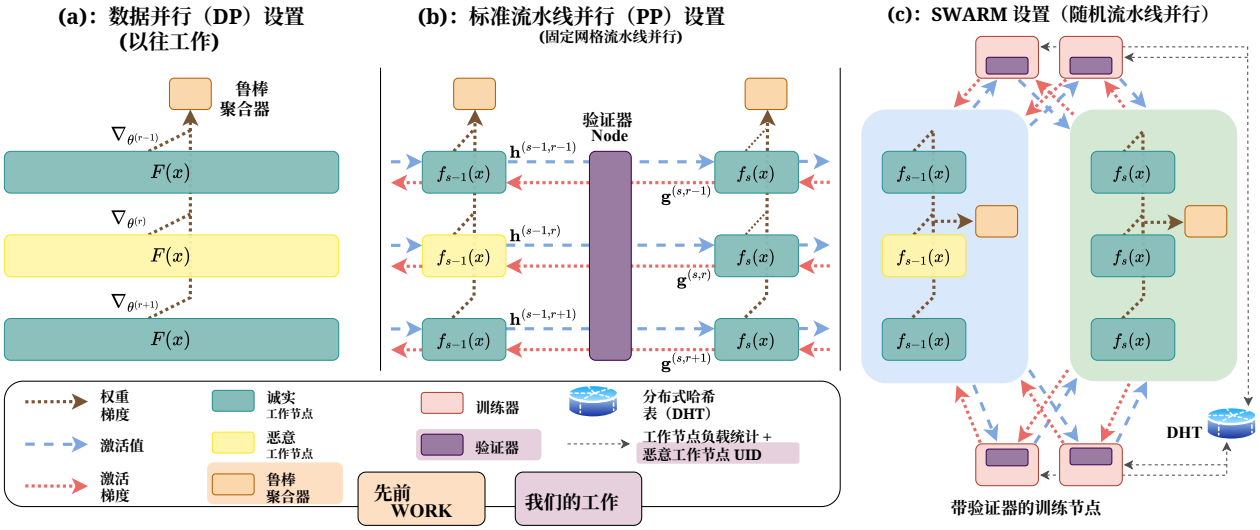


图2：分布式威胁模型。（**a**）在数据并行中，工作节点持有完整的模型副本，并且只发送权重梯度。**传统的拜占庭容错方法考虑这种情况，并使用鲁棒聚合。（b 和 c）本文考虑的威胁模型**（见第2.1节）。在流水线并行中，工作节点持有各自的层，并发送中间激活值 $\boldsymbol{h}$和激活梯度 $\boldsymbol{g}$，因此损坏会直接影响其他工作节点。（**b**）在标准设置中，存在由流水线阶段和数据并行副本组成的固定网格，通信经由我们的验证节点进行路由。（**c**）在为去中心化设置设计的 SWARM 设置中，数据由训练器节点随机路由。工作节点将其计算结果（$\boldsymbol{h}$ 和 $\boldsymbol{g}$）发送给训练器，训练器再将其路由到下一阶段中一个可用的工作节点。我们提出的验证器可以无缝地添加在这些训练器节点之上（第G节）。

攻击变体。令 $\boldsymbol{x}$ 表示可能被受损节点操纵的任意激活向量或梯度向量，并用 $\hat{\boldsymbol{x}}$ 表示其被操纵后的版本。我们考虑：

- **常量攻击：**攻击者提交一个常数向量，而不执行指定的计算，即 $\hat{\boldsymbol{x}} = \boldsymbol{c}$，例如由负一或零组成的向量。
- **随机值攻击：**攻击者用从标准正态分布中随机抽取的值替换向量元素：$\hat{\boldsymbol{x}} \sim \mathcal{N}(\mathbf{0},\ I)$。

- **缩放攻击：**攻击者将向量缩放 α 倍，即 $\hat{\boldsymbol{x}} = \alpha\boldsymbol{x}$。
- **随机符号攻击：**攻击者以 p 的概率翻转每个向量元素的符号。
- **偏置添加攻击：**攻击者向向量添加随机噪声：$\hat{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\epsilon}$，其中 $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\ \sigma^2 I)$。为使攻击更隐蔽，可以选择 σ，使其与原始向量的幅值相匹配。
- **延迟攻击：**攻击者使用历史值：$\hat{\boldsymbol{x}}_t = \boldsymbol{x}_{t-k}$，其中 k 表示延迟步数。
- **不可见噪声攻击：**受 ALIE 攻击（Baruch 等人，2019）的启发，攻击者用统计上不易察觉的边界值替换良性值：$\hat{\boldsymbol{x}} = \boldsymbol{\mu} + z_{\text{最大值}}(\boldsymbol{\sigma} \odot \boldsymbol{\epsilon})$，其中 $\boldsymbol{\mu}$ 为原始向量的均值，$\boldsymbol{\sigma}$ 为其逐元素标准差，$z_{\text{最大值}} = \sqrt{2}\cdot\text{erfinv}(2p-1)$，其中 $p = 1-\alpha$ 为分位数阈值，erfinv 是逆误差函数，且 $\boldsymbol{\epsilon} \sim \mathcal{N}$（$\mathbf{0}$，$I$）。

3. 基于动量的工作者验证

普通流水线并行在前向传播和反向传播中仍容易受到通信损坏的影响。当恶意工作节点注入被污染的激活值或激活梯度时，其影响可能沿网络逐步扩散，甚至在几乎无法察觉的情况下损害模型收敛。全量计算复制（Lu 等人，2024年；Rajput 等人，2019）等朴素方法虽能保证检测到攻击，却会降低训练吞吐量；而随机采样验证则可能漏掉某些仅需几次迭代就能破坏训练的攻击（见图 7）。

我们提出 SENTINEL：一种轻量且具有坚实统计学依据的验证机制，通过利用激活值及其梯度的指数移动平均，建立可靠的参考点，以检测异常行为。我们设计了一种算法，利用四分位距自适应地设定异常检测测试的阈值。在较宽松的假设条件下，我们从解析上证明，在我们的验证框架下未被检测到的受损工作节点对最终模型收敛的影响可以忽略不计。

3.1 提出的方法

我们方法的核心洞见是，在正常的分布式训练场景中，每个工作节点的激活值和梯度都应与整体总体呈现统计一致性。现有的工作复制方法，例如 Lu 等人 (2024年)，需要分配一半计算资源用于工作验证，因此会消耗大量计算资源。指数移动平均值作为我们提出的轻量级验证方法的基础，具有三项关键优势：

- **计算效率**：计算和更新指数移动平均统计量所需的计算量和内存开销极小（复杂度为 $\mathcal{O}(m)$，其中 m 是激活值/梯度的大小），因此适用于计算资源受限的验证器节点。
- **时间平滑**：指数移动平均值能够自然地平滑小批次噪声，同时捕捉合法工作节点输出的底层分布，从而形成一个鲁棒的参考点。
- **自适应性**：随着训练分布发生变化，指数移动平均值会自动适应这些变化，同时抵御恶意工作节点造成的突然偏差。

因此，我们设计了 SENTINEL，包含四个关键组件：（1）使用指数移动平均值作为统计参考点；（2）选择合适的距离度量来检测偏差；（3）采用自适应阈值进行异常检测；（4）处理分布式架构中的级联效应。下文将详细介绍这些组件。有关我们方法的详细分步概述，请参阅附录中的算法 1 。此外，有兴趣的读者可参阅第 G 节，了解 SENTINEL如何集成到 SWARM （ Ryabinin 等人，2023年）中。

以指数移动平均作为参考点。我们利用每一层激活值和梯度的指数移动平均值作为统计参考点来检测偏差。由于指数移动平均值可作为期望值的鲁棒近似（Robbins和Monro，1951年），因此适用于检测异常。对于非恶意行为者，我们预期偏差较小，因为优化轨迹通常保持平滑。形式上，每个验证器节点都会维护激活值的运行指数移动平均值：

$$\boldsymbol{m}_t^{(s)}(\boldsymbol{h}) = \beta_h \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{h}) + (1-\beta_h)\frac{1}{d_s}\sum_{r=1}^{d_s} \boldsymbol{h}_t^{(s,r)}, \tag{1}$$

其中 $\boldsymbol{m}_t^{(s)}(\boldsymbol{h})$ 表示阶段 s的激活值指数移动平均，$\beta_h \in [0,1)$ 是衰减率。**我们使用类似的方程来计算梯度 $\boldsymbol{m}_t^{(s)}(\boldsymbol{g})$ 的指数移动平均，其衰减率为** $\beta_g \in [0,1)$。为获得可靠的初始估计，我们设置了一个仅包含诚实工作节点的“预热”阶段，在此期间收集并稳定指数移动平均统计量，然后再开始验证（更多说明请参见第 A 节中的 Q4）。预热期结束后，每当工作节点提交新的信号，验证器都会进行轻量级统计检验，将这些信号与已建立的指数移动平均进行比较。偏差决定是否将某个工作节点标记为恶意。具体而言，对于工作节点 $\boldsymbol{h}_t^{(s,r)}$ 提交的激活值 $,r)$，验证器计算：

$$\Gamma_t^{(s,r)} := \Omega\left(\boldsymbol{h}_t^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{h})\right), \tag{2}$$

其中，$\Omega(\cdot,\ \cdot)$ 是一种合适的距离度量。如果 $\Gamma_t^{(s,r)} > \tau$，则将工作节点标记为恶意，并跳过指数移动平均的更新，以维护验证完整性。我们对 $\boldsymbol{g}_t^{(s,r)}$ 使用类似的检测器。

距离度量的选择 Ω。距离函数会显著影响验证的敏感性。我们不采用单一度量指标，而是使用一组度量指标 $\mathcal{M}$，包括 平均绝对差、归一化欧几里得距离、切片 *Wasserstein* 距离和符号翻转比率，以鲁棒地检测各种攻击类型。这些度量指标能够捕捉分布偏移的不同方面，而这些偏移可能表明存在恶意行为，定义如下：

- **平均绝对差（L_1）**：该指标衡量信号之间的平均绝对偏差：

$$\Omega_{L_1}(\boldsymbol{h}_t^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{h})) = \mathbb{E}[\|\boldsymbol{h}_t^{(s,r)} - \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{h})\|_1], \tag{3}$$

  其中 $\mathbb{E}[\cdot]$ 表示对所有特征的期望。
- **归一化欧几里得距离（L_2）**：该指标计算白化表示之间的平方差：

$$\Omega_{L_2}(\boldsymbol{h}_t^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{h})) = \mathbb{E}[\|\bar{\boldsymbol{h}}_t^{(s,r)} - \bar{\boldsymbol{m}}_{t-1}^{(s)}(\boldsymbol{h})\|^2], \tag{4}$$

  其中 $\bar{\boldsymbol{x}}$ 表示 $\boldsymbol{x}$的白化（z分数标准化）版本。
- **符号翻转比率（SFR）**：该指标量化符号相反的坐标所占比例，其取值范围限于$[0, 1]$：

$$\Omega_{SFR}(\boldsymbol{h}_t^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{h})) = \mathbb{E}[\mathbf{1}(\text{sign}(\boldsymbol{h}_t^{(s,r)}) \neq \text{sign}(\boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{h})))]. \tag{5}$$

- **切片 Wasserstein 距离（SW）**：该方法通过随机投影来近似瓦瑟斯坦距离：

$$\Omega_{SW}(\boldsymbol{h}_t^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{h})) = \mathbb{E}_{\boldsymbol{u}\sim\mathcal{S}^{d-1}}[W_1((\boldsymbol{h}_t^{(s,r)})_{\#}\boldsymbol{u}, (\boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{h}))_{\#}\boldsymbol{u})] \tag{6}$$

  其中 W_1 为 1-瓦瑟斯坦距离，$(\cdot)_{\#}\theta$ 表示到来自单位球面 $\mathcal{S}^{d-1}$的随机单位向量$\boldsymbol{u}$上的投影。

我们在整个训练过程中跟踪每个阶段的所有这些度量指标。这些相同的度量指标用于衡量所提交梯度 $\boldsymbol{g}_t^{(s,r)}$ 与其动量 $\boldsymbol{m}_{t-1}^{(s)}$（$\boldsymbol{g}$）之间的偏差。若某个工作节点在任一度量指标上超过阈值，则将其标记。多样化的度量指标集合可针对各种攻击向量提供鲁棒性，因为不同的攻击可能体现在不同的统计特性中。基于神经网络的距离度量也可能是一个有前景的候选方案，但留待未来工作。关于各度量指标影响的消融实验，请参阅第5.1节。

自动阈值 τ 更新。每种距离度量都需要一个阈值 τ 来标记异常行为。在前文所述的初始“预热”期间，我们收集每个阶段各度量指标的有效偏差并计算四分位距。随后，我们使用 Tukey 栅栏（Tukey，1977年）来确定有效偏差界限。具体而言，令 $\mathcal{H}_l^s$ 表示过去 l 次迭代中从所有 d_s 个工作节点在阶段 s所收集的有效偏差历史。令 q_1、 q_2和 q_3 分别为这些偏差的第25、第50（中位数）和第75百分位数。我们将检验统计量定义为：

$$\text{if } |\Gamma_t^{(s,r)} - q_2| \geq k(q_3 - q_1) \Rightarrow \text{flag node } (s,r) \text{ as adv.} \tag{7}$$

其中 $k = 1.5$ 是常规选择（Tukey，1977年）。我们选择通过迭代过程自适应地调整 k，通过扩大或缩小阈值来维持选定的假阳性率（例如，$< 1\%$）。该动态阈值会持续将新的良性偏差纳入历史窗口，使验证系统能够在整个训练过程中自动适应不断变化的数据（见图 12）。更多详情请参见算法 5。

处理级联效应。 在我们的基于流水线并行的分布式架构中，早期阶段的被污染激活值可能影响下游工作节点，进而导致诚实节点被错误分类。我们通过两种互补机制来解决这一问题：

1. **自底向上的恶意节点识别**：当阶段 s 中的工作节点被标记为恶意节点时，验证器会通知所有下游验证器，暂停对受影响小批次的偏差统计数据收集，并将后续异常标记为“受上游影响”。为确保训练不中断，在反向传播过程中，验证器会发送已存储的梯度 EMA，从而在不泄露任何行为变化的情况下实现参数更新。更多信息请参阅第 D.2 节和图5。

2. **带宽恕机制的违规计数器**：验证器不会因一次偏差就封禁工作节点，而是各自维护一个违规计数器。严重偏差（$\times 100$ 超过阈值）会导致立即封禁，而较轻微的偏差则会使计数器加一。工作节点在发生 c 次违规后会被封禁，但连续 $T_{\text{forgiveness}}$ 步无违规后，计数器会递减，从而允许其从暂时性异常中恢复。我们在实验中使用$c = 5$ 和 $T_{\text{forgiveness}} = 100$。

3.2 理论分析

作为对我们方法的补充，我们在较宽松的条件下给出理论保证，分析：（1）分布式训练在有界恶意扰动下的收敛行为；（2）我们的系统在每个阶段维持诚实多数所需的条件。

定理3.1（有界扰动下的收敛性（非正式））。 考虑一种分布式训练设置：使用 *PP* 将模型层分配给各工作节点，并使用基于动量的验证，在前向传播或反向传播中验证每个工作节点的贡献。[a] 此外，假设每个阶段的工作节点中，恶意工作节点不足一半（即 $\gamma_s < 1/2$），并且我们使用固定阈值 τ，依据式 (2) 验证工作节点。[b] 在这些较宽松的条件下，使用动量随机梯度下降优化的非凸损失函数训练将收敛到一个驻点邻域，该邻域的大小与 τ 成正比。

定理3.1 指出，恶意工作节点通过将扰动控制在阈值以下来逃避检测 τ，只能使最终解偏离最优解，且偏离幅度与 τ成正比。形式化定理见第E.1节。另请参阅常见问题（第A节）中的 Q8。

请注意，SENTINEL基于这样一个假设：任一阶段中恶意工作节点的数量都不到工作节点总数的一半（即，$\gamma_s < 1/2$）。接下来，我们将量化在给定恶意工作节点总预算的情况下，该假设以高概率成立的条件。证明见第E.2节。

引理 3.2（诚实多数保证）。考虑一个包含 p 个流水线阶段的分布式训练系统，每个阶段都由 d 个工作节点副本组成。设 b 为恶意工作节点总数，$\epsilon \in (0,1)$ 为一个较小的正常数。如果工作节点随机分配到各个阶段，且 $b \leq dp/2 - p\sqrt{d/2}\ln(p/\epsilon)$，则以至少 $1-\epsilon$ 的概率，每个流水线阶段中的恶意工作节点都严格少于 $d/2$ 个。

4. 相关工作

我们基于动量的验证方法与三个大体上独立发展的主要研究方向相交叉。关于相关工作的更全面综述，见第C节。

去中心化大语言模型训练 已成为推动人工智能发展的民主化力量。Tasklets（Yuan 等人，2022）和 SWARM（Ryabinin 等人，2023年）等框架在

[a] 我们不考虑在用于在数据并行副本之间同步参数梯度的“全归约”操作期间的不诚实行为（这是所有先前拜占庭容错文献所针对的设定）。这是一个互补的维度，可以采用任何先前的拜占庭容错工作。我们在第5.1节中对SENTINEL与这些防御机制的兼容性进行了实验。[b] 这是为了放宽我们的条件。实践中，我们在每次迭代中使用四分位距自动设置该阈值。

通信效率和应对非恶意故障的容错能力方面取得了显著进展，但仍容易受到对抗性参与者的攻击。我们的工作旨在解决这些漏洞。

机器学习中的拜占庭鲁棒性 传统上主要关注联邦学习场景，在这些场景中，每个工作节点都会计算完整的模型更新。Krum（Blanchard 等人，2017年）、Bulyan（Mhamdi 等人，2018年）和 CENTEREDCLIP（Karimireddy 等人，2021 年）等经典方法依赖于比较各工作节点的完整梯度，这与流水线并行架构存在根本性的不匹配，因为在这种架构中，每个工作节点只计算模型的一部分。SENTINEL 专为流水线并行的独特约束而设计。

安全分布式系统 原则为我们的验证器节点架构提供了指导，该架构借鉴了分布式计算中可信中介的设计理念。尽管 Lu 等人（2024年）的初步研究指出了流水线并行训练中的潜在漏洞，但他们基于冗余的方案会大幅降低吞吐量，从而抵消分布式训练的主要优势。我们的轻量级验证机制以极低的计算开销提供鲁棒的安全保证。

5. 实验结果

本节介绍我们的实验结果。除非另有说明，我们使用一个 0.6B 参数的 Llama-3（Dubey 等，2024；Liang 等，2025）模型（16 层、32 个注意力头、1024 的隐藏维度和上下文长度），将其分布在包含 128 个工作节点的 8×16 数据-流水线并行网格上。我们采用 AdamW（Loshchilov 和 Hutter，2019）优化器，初始学习率为 $6e{-4}$，并在 FineWeb（FW）（Penedo 等人，2024）、OpenWebText（OW）（Gokaslan 等人，2019）和 Common Crawl（C4）（Raffel 等人，2020）数据集上训练模型 5k 步。我们将每个流水线阶段中 25% 的工作节点随机指定为恶意节点（恶意与诚实节点比例为 2:6），并且这些节点中仅有 25% 会同时激活，以弱化我们的“无共谋”假设。训练先进行 1k 步的预热期，然后启动验证。根据原始情形下的验证运行结果，我们设置 $\beta_h = 0.9$ 和 $\beta_g = 0.8$，用于激活验证和梯度验证。最后，自适应阈值采用过去 100 步的滑动窗口。我们通过各种消融研究放宽这些假设，以考察其影响。详细的实验设置和扩展结果见 F 节。

度量指标。 我们使用精确率（Pr）、召回率（Re）和 F1 分数评估验证方法，以衡量其检测恶意工作节点的有效性，同时尽可能减少假阳性。为量化检测效率，我们将检测速度 报告为从恶意行为开始到恶意工作节点被封禁期间的平均迭代次数。我们还通过比较各方法在最后一个训练步骤的平均验证损失，来比较其收敛率。

验证性能。 我们在 C4数据集上训练了采用和未采用我们验证机制的 Llama-3-0.6B 模型，并在表中报告结果。 1 （有关 C4 和其他数据集的更全面结果，请参见第 F节。）我们的实验得出三项关键发现：

1. 在流水线并行中，激活攻击与梯度攻击同样具有威胁性，但拜占庭容错文献一直忽视了这类风险。2. 攻击在激活域和梯度域中的效果存在显著差异。同一种技术在针对激活值时可能严重扰乱训练，但应用于梯度时影响却微乎其微（例如，不可见噪声攻击）。3. 我们的指数移动平均验证方法在各种攻击类型上都取得了较高的 F1分数。当攻击造成的偏差微不足道并逃过检测时，其对收敛的影响仍然有限，这证实了定理 3.1 中的理论分析：未检测到的攻击只能使参数偏离最优点，并停留在其邻域内。图1 还通过散点图证实了这一关系。该图展示了表 1、15 和 16 中 75 种以上不同攻击设置的 F1分数与验证损失之间的关系。

表1：Llama-3-0.6B 在 C4数据集上的攻击检测性能。所示度量指标包括精确率、召回率、F1分数（均以百分比表示）、平均检测速度（以迭代次数计）以及验证损失。

模式攻击		SENTINEL（我们的方法）					无验证
		PR. (%) ↑	RE. (%) ↑	F1 分数 (%) ↑	检测速度 ↓	验证损失 ↓	验证损失 ↓
-	无（原始方法）	100.0	100.0	100.0	N/A	3.819	3.821
ACTIVATION MANIPULATION	常数（零）	100.0	100.0	100.0	6.5	3.809	11.761
	常数（一）	100.0	100.0	100.0	6.33	3.817	7.778
	随机值	100.0	100.0	100.0	6.48	3.827	7.778
	缩放（$\alpha = -1$）	100.0	100.0	100.0	6.38	3.824	4.109
	随机符号（1%）	100.0	100.0	100.0	6.33	3.825	4.670
	随机符号（10%）	100.0	100.0	100.0	6.52	3.822	4.619
	随机符号（30%）	88.9	100.0	94.1	70.91	3.841	4.567
	延迟（100 步）	88.9	100.0	94.1	13.21	3.841	7.675
	偏置添加	84.6	91.7	88.0	14.57	3.830	3.892
	不可见噪声（90%）	100.0	100.0	100.0	6.48	3.836	7.675
	不可见噪声（95%）	100.0	100.0	100.0	6.52	3.823	7.677
	不可见噪声（99%）	100.0	100.0	100.0	6.48	3.826	7.682
GRADIENT MANIPULATION	常数（零）	100.0	100.0	100.0	6.42	3.829	3.942
	常数（一）	88.9	100.0	94.1	1.0	3.816	10.630
	随机值	100.0	100.0	100.0	1.0	3.818	9.595
	缩放（$\alpha = -1$）	0.0	0.0	0.0	N/A	3.893	3.893
	随机符号（1%）	0.0	0.0	0.0	N/A	3.990	3.982
	随机符号（10%）	0.0	0.0	0.0	N/A	3.982	3.994
	随机符号（30%）	0.0	0.0	0.0	N/A	3.944	3.933
	延迟（100 步）	100.0	100.0	100.0	7.33	3.826	10.157
	偏置添加	100.0	100.0	100.0	1.0	3.828	10.813
	不可见噪声（90%）	100.0	75.0	85.7	101.89	3.968	4.218
	不可见噪声（95%）	100.0	79.2	88.4	229.68	3.954	4.174
	不可见噪声（99%）	100.0	79.2	88.4	211.0	3.943	4.176

混合攻击。虽然此前的实验分别考察了单独的攻击类型，但在真实场景中，对手通常会在不同节点上同时采用各种攻击策略。为了评估我们的方法在更贴近现实的条件下的表现，我们在分布式训练环境中使用 Llama-3-0.6B 模型开展实验，每个阶段有 37.5% 的工作节点为恶意节点（具体而言，每个阶段有 3 个恶意工作节点和 5 个诚实工作节点，构成一个 8×16 网格）。所有攻击者同时开始实施恶意行为，并为每个攻击者随机分配一种攻击模式（激活操控或梯度操控）以及第 2 节中的一种具体攻击策略。表 2 汇总了我们在三个数据集上的实验结果。结果表明，我们的验证方法能够成功识别大多数危害显著的攻击，使验证损失指标与未受攻击的基线模型相当。这种对异构攻击向量的鲁棒性，凸显了 S E NT IN E L 在保护流水线并行大语言模型训练免受复杂对抗场景影响方面的有效性。

表 2：针对混合攻击训练 Llama-3-0.6B 的检测性能。

数据集	SENTINEL（我们的方法）					原始
	精确率 (%) ↑	召回率 (%) ↑	F1 分数 (%) ↑	检测速度 ↓	验证损失 ↓	验证损失 ↓
C4	83.7	92.3	87.8	78.14	3.831	3.821
FW	81.8	92.3	86.7	66.00	3.827	3.840
OW	91.9	87.2	89.5	52.70	3.784	3.778

表 3：C4数据集上大规模 Llama-3 训练的激活攻击检测性能。

设置	攻击	SENTINEL（我们的方法）				
		精确率。(%) ↑	召回率.(%) ↑	F1 分数 (%) ↑	检测速度 ↓	验证损失 ↓
0.6B w/ 16 × 16 Mesh	随机值	100.0	100.0	100.0	7.96	3.900
	延迟（100 步）	85.7	100.0	92.3	14.39	3.945
	偏置添加	100.0	25.6	40.8	65.15	3.981
	不可见噪声	100.0	100.0	100.0	7.96	3.898
1.2B w/ 8 × 8 Mesh	随机值	100.0	100.0	100.0	4.33	3.723
	延迟（100 步）	37.5	100.0	54.5	67.0	3.774
	偏置添加	0.0	0.0	0.0	N/A	3.738
	不可见噪声	100.0	100.0	100.0	4.33	3.727

大规模实验。 为了验证我们的方法在大规模场景下的有效性，我们在另外两种设置中开展了实验：(1) 16×16 包含 256 个工作节点的网格配置；(2) 在包含 64 个工作节点的 8×8 网格上训练更大的 1.2B 参数模型。在这两种场景中，我们均保持每个阶段有 37.5% 的恶意工作节点（对于 16×16 网格，恶意与诚实工作节点的比例为 6:10；对于 8×8 网格，该比例为 3:5），并且每个攻击轮次中有 25% 的攻击者处于激活状态。表 3 （扩展结果见表 17 ）中的结果表明，我们的验证机制能够在所有攻击场景下有效保持训练完整性，验证损失与未受攻击的基线结果相当。与 0.6B 设置相比，这些结果无需进行广泛的超参数调优即可实现，不过仍需改进，以减少 1.2B 模型上激活延迟攻击中的假阳性。

5.1 消融研究。

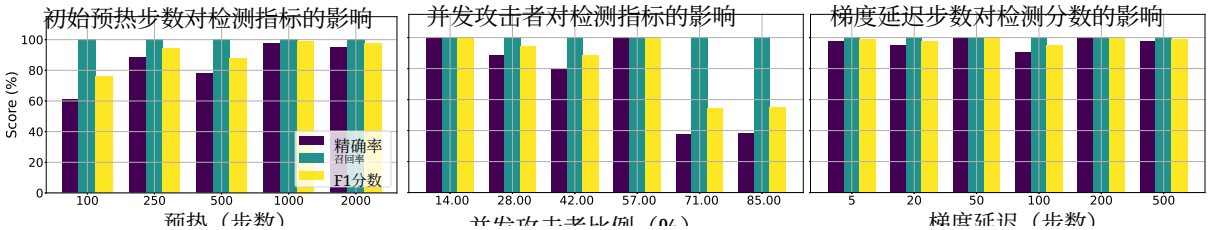
接下来，我们研究关键因素如何影响我们的验证方法检测恶意工作节点的能力。除非另有说明，我们使用最强攻击进行评估：99% 的不可见噪声激活攻击（研究 1 和 2）、100 步的延迟梯度攻击（研究 3），以及所有激活攻击的混合攻击（研究 4 和 5）；每个阶段的恶意工作节点与诚实工作节点比例始终为 3:5（恶意工作节点占 37.5%）。

初始预热期。我们的方法需要一个初始预热阶段，以确保训练已达到稳定状态。图3a 显示，尽管早期检测的召回率很高，但由于训练波动，阈值尚不够鲁棒，精确率起初较低。大约经过 1k 步后，随着验证方法建立起可靠的界限，精确率趋于稳定。

攻击者共谋。恶意工作节点协调攻击时，检测会更具挑战性。如图3b所示，我们的验证方法在恶意工作节点之间的共谋比例高达 60% 时仍然有效。超过这一阈值后，假阳性会显著增加，从而可能对精确率造成不利影响。

梯度延迟的影响。图3c 展示了我们的方法对不同梯度攻击延迟时长的鲁棒性。即使延迟极短，恶意梯度与当前合法梯度十分相似，我们的验证方法仍能保持较高的检测率。

替代架构。 为展示 SENTINEL 在不同 Transformer 架构之间的可迁移性，我们在 FineWeb-EDU（Shazeer等人，2017年）数据集上训练了两种混合专家（MoE）（MetaAI，2025年）架构，分别为 Llama-4-0.4B（DeepSeek-AI，2024年）和 DeepSeek-V3-1B（Penedo 等人，2024年）。我们假设 33% 的对手会串谋，同时发送其恶意向量。我们训练这些模型及其原始基线，直至 5000 步。为展示所选超参数的可迁移性，我们对 Llama-3-0.6B 使用相同的超参数



（a）预热步数　　（b）同时攻击者　　(c) 延迟步数

图3：关于各要素对验证性能影响的消融研究。

用于验证的实验。如表 4 所示，SENTINEL 可成功应用于这些基于混合专家的模型，展现出其在不同架构之间卓越的可迁移性。

大规模模型。 接下来，我们表明，将模型规模从中等规模扩大到大规模可以在不造成明显性能下降的情况下完成。为此，我们扩展了 Llama-3-0.6B 实验，通过增加隐藏维度和层数，使总参数量达到约 40 亿。随后，我们使用 176 个工作节点，在严格的总批量大小 96 下训练该模型，以适配我们可用的计算资源，并在 FineWeb 数据集上训练 5k 步。如表 4 所示，SENTINEL 保护训练免受中断攻击，并得到一个验证损失接近原始、未受攻击基线的模型（差值仅为 0.04），同时将检测准确率保持在约 80%。这表明我们的结论可以扩展到层更宽/更深的更大规模模型。

表4：针对混合激活攻击训练替代模型的检测性能。

架构	SENTINEL（我们的方法）				原始
	精确率 (%) ↑	召回率 (%) ↑	F1 分数 (%) ↑	验证损失 ↓	验证损失 ↓
Llama-4-0.4B	73.5	92.3	81.8	3.617	3.628
DeepSeek-V3-1B	94.6	97.2	95.9	3.421	3.393
LLAMA-3-4B	94.9	68.6	79.6	3.714	3.668

指数移动平均在验证中的影响。为展示指数移动平均中捕捉到的时间训练动态如何影响对恶意工作节点的检测，我们将本方法与一个简单版本进行比较，后者仅将提交的信号与平均值进行比较。在此实验中，我们启动一个 Llama-3-0.6B 模型的训练，同时恶意工作节点发起激活延迟攻击。为消除指数移动平均的影响，我们将 $\beta = 0$ 设为该值，这实质上意味着我们将与瞬时激活平均值进行比较。结果见表 5。如图所示，与瞬时均值进行比较不足以防范恶意工作节点。这进一步证实了时间模式对于检测流水线并行攻击的重要性。此外，我们通过将 ENTINEL的指数移动平均衰减率 $\beta_h = \beta_g$调整为 0。6和 0。99，测量 S 5对衰减率的敏感性。如表 5所示，对衰减率的敏感性并不显著。

表5：关于指数移动平均对 S ENTINEL 影响的消融实验。

参考点	度量指标			
	PR. (%) ↑	RE. (%) ↑	F1 分数(%) ↑	验证损失 ↓
平均值 ($\beta_h = \beta_g = 0$)	23.08	100.0	37.5	6.248
SENTINEL ($\beta_h = 0.9,\ \ \beta_g = 0.8$)	**100.0**	**100.0**	**100.0**	**3.826**
SENTINEL ($\beta_h = \beta_g = 0.60$)	94.7	**100.0**	97.3	3.894
SENTINEL ($\beta_h = \beta_g = 0.99$)	90.0	**100.0**	94.7	3.875

距离度量的影响。不同攻击对各种距离度量的敏感性，在激活攻击和激活梯度攻击之间有所不同，因此我们需要采用多种距离度量。正如前文所述

在第3.1节中，其他最优距离度量的选择或许能够提供统一的解决方案（例如，神经网络分类器），我们将其留待未来工作探讨。此处，为了评估各种距离度量的影响，我们采用表2中的混合攻击设置进行消融实验：在 C4数据集上训练 Llama-3-0.6B，并且每次仅使用一种距离度量来检测混合攻击。结果如表6所示。可以看出，结合所有度量指标，在应对所有激活攻击和激活梯度攻击的混合攻击时表现最佳。

表6：针对混合攻击的距离度量消融实验。

距离度量	度量指标			
	PR. (%) ↑	RE. (%) ↑	F1 分数(%) ↑	验证损失 ↓
SFR	42.9	83.3	56.6	8.882
SWD	40.0	**94.4**	56.2	6.332
归一化 L_2	75.0	75.0	75.0	10.274
绝对偏差 L_1	71.1	88.9	79.0	3.883
全部（SENTINEL）	**83.7**	92.3	**87.8**	**3.831**

随机种子的影响。由于计算资源有限且实验成本高昂，本文未报告误差线。为了证明我们方法的统计完整性，我们针对两种最具挑战性的激活攻击计算了误差线：延迟（100 步）和不可见噪声（ 99%）。我们随机选取了 5 个种子，并重复了表 1中的实验，同时随机化网络初始化和恶意工作节点选择。表 7中的结果表明了我们研究发现的统计显著性。

表 7：针对激活延迟和隐形噪声攻击，SENTINEL 性能的统计显著性分析及误差线。

攻击	度量指标			
	PR. (%) ↑	RE. (%) ↑	F1 分数 (%) ↑	验证损失 ↓
延迟（100 步）	99.2 ± 1.6	100.0 ± 0.0	99.6 ± 0.8	3.843 ± 0.008
不可见噪声（99%）	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	3.832 ± 0.004

自适应攻击。 在接下来的研究中，我们探讨一种自适应攻击，该攻击者了解 S EN TI NEL 如何验证其信号。具体而言，假设恶意工作节点会对发送至后续层的信号维护一个指数移动平均，并使用与验证器节点相同的 β。在收集到足够的指数移动平均样本后，攻击会发送偏向预定目标方向的漂移激活值/梯度，并利用过时的动量估计制造持续偏差，同时使其在统计上看起来合理。攻击者根据 $\hat{\boldsymbol{x}} = \boldsymbol{m}_{t-\delta} + \alpha \cdot (\boldsymbol{t}_{\text{drift}} - \boldsymbol{m}_{t-\delta}) / ||\boldsymbol{m}_{t-\delta}|| + \boldsymbol{\epsilon}$ 构造攻击，其中 $\boldsymbol{m}_{t-\delta}$ 是过时的指数移动平均动量， α是漂移率， $\boldsymbol{t}_{\text{drift}}$ 是预定漂移目标， $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2)$ 是用于隐蔽的高斯噪声，而 $\delta = \left\lfloor \frac{\log(0.1)}{\log(\beta)} \right\rfloor$是延迟因子。我们在使用 C4数据集训练 Llama-3-0.6B 时，针对激活值和激活梯度运行此攻击。对于激活攻击，验证器/攻击者使用 $\beta = 0.9$；对于梯度攻击，则使用 $\beta = 0.8$。

使用表1 中的设置运行此攻击（每个阶段有 25% 的恶意工作节点），我们得到了表 8中的结果。可以看出，自适应指数移动平均攻击在没有验证时可能具有破坏性；S EN TIN EL 能够完美地检测并缓解该攻击，从而验证了我们对了解我们防御的自适应攻击的韧性。这是因为攻击者的指数移动平均仅构成真实总指数移动平均的一部分，并且在假设诚实多数的前提下，这不足以中断训练。

与数据并行防御的结合。正如本文全文所阐述的，S E N T INE L 旨在验证 PP 中阶段之间传递的信号。与 PP 轴不同，我们指出，现有的拜占庭容错文献

表8：针对自适应攻击的验证性能。

MODE	SENTINEL（我们的方法）				无验证
	精确率（%） ↑	召回率（%） ↑	F1 分数（%） ↑	验证损失 ↓	验证损失 ↓
激活	100.0	100.0	100.0	3.835	7.776
梯度	100.0	100.0	100.0	3.818	7.777

仅针对梯度平均（又称全归约）期间发生的攻击。因此，这两个轴互为补充，作用于不同维度（请参阅第 A 节中的 Q0）。为展示这两个轴的互补性，我们采用两种常用的鲁棒聚合器，具体来说是 Krum（Blanchard 等人，2017年）和 Bulyan（Mhamdi 等人，2018年），来替换工作节点之间的原始全归约。随后，我们在 FineWeb 上训练 Llama-3-0.6B，使其在恶意与诚实比例为 3:5 的情况下抵御混合攻击，共训练 5k 步。如表 9 所示，在正交的数据并行轴上加入这些方法，对模型在混合攻击下的收敛没有显著影响，这支持了我们的观点：PP 和数据并行解决方案互为补充。

表9：SENTINEL 在混合攻击下训练 Llama-3-0.6B 时，面对鲁棒聚合器的表现。

方法	受攻击的训练			
	精确率(%) ↑	召回率(%) ↑	F1 分数(%) ↑	验证损失 ↓
SENTINEL +Krum	93.6	80.6	86.6	3.873
SENTINEL +Bulyan	85.3	80.6	82.9	3.883

5.2 将 SENTINEL 扩展至 SWARM 并行

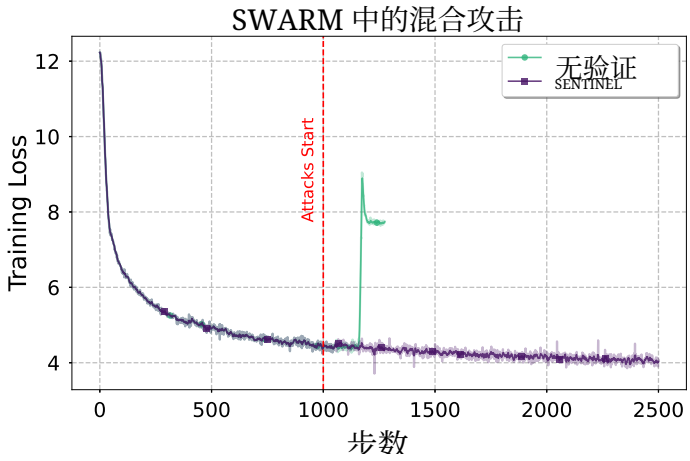
最后，我们调整 SENTINEL，以在一项真实的 SWARM（Ryabinin 等人，2023年）实验中验证工作节点信号。SWARM 并行由 Hivemind 框架（Ryabinin 和 Gusev，2020）驱动，提供了一个容错的分布式训练生态系统。它由分布在数据并行和流水线并行坐标上的工作节点组成。在每个阶段，一组工作节点处理数据批次，训练器节点负责协调，并在前向传播中随机路由激活值，在反向传播中随机路由激活梯度。从这个角度来看，与我们迄今考虑的固定设置相比，SWARM 并行类似于一个随机数据并行／流水线并行网格。如第 2.1 节所述，训练器节点在加入 SENTINEL 后，自然适合扩展为验证节点。

在本实验中，我们在包含 128 个工作节点（8×16 网格）的分布式 SWARM 配置上训练 Llama-3-0.6B 模型。我们使用 32 个具备验证能力的训练器节点，在 FineWeb-EDU（FineWeb 的精选子集）上训练模型。由于训练器不具备点对点通信，每个训练器各自维护独立的指数移动平均值，并仅在预热结束时进行一次同步。

我们针对混合随机攻击评估了鲁棒性：在每个阶段（第一个阶段和最后两个阶段除外）将 37.5% 的工作节点指定为恶意工作节点，维持 3:5 的恶意与诚实比例，并设置 15% 的串谋，使攻击者同时发动攻击。如图4，在缺乏验证的情况下，恶意工作节点的存在会显著破坏训练收敛。然而，SENTINEL 成功维持了训练的完整性，使训练能够不间断地继续。这一生产级环境中的结果证明了我们的方法在保障现实世界去中心化训练生态系统方面的实际适用性。有关 SENTINEL 在 SWARM 中的实现细节与完整结果，请参见第G。

6. 结论

本文研究了去中心化流水线并行网络中的安全漏洞，展示了恶意工作节点如何破坏流水线阶段之间交换的激活值和激活梯度。



使用 SWARM（Ryabinin 等人，2023年）在 128 个可抢占式 AWS 实例上的分布式工作节点上训练 Llama-3-0.6B 模型时的损失。工作节点采用各种激活/梯度操纵攻击来干扰训练。缺乏验证时，训练会受到干扰；而 SENTINEL 能够成功保护训练，避免发散。

为指导未来的研究，我们提出了一套训练中断攻击基准，用于评估去中心化训练安全。我们的核心贡献是 SENTINEL，这是一种轻量级的、基于动量的验证机制，它利用可信验证节点维护传输信号（激活值和激活梯度）的指数移动平均值，作为统计参考点。我们进一步开发了一种基于四分位距的自适应阈值策略，以自动校准检测灵敏度。我们通过理论分析和基于 SWARM 的去中心化训练实际集成，对该方法进行了补充。通过在数百个工作节点上分布式运行参数规模高达 4B 的模型进行大量实验，我们证明了该方法能够有效维持训练完整性，并且在各种攻击场景下始终保持较高的 F1 分数（$> 85\%$）。

局限性：虽然我们的验证方法能够有效检测所呈现的攻击类型，但未必能推广到所有可能的对抗策略。未来的工作应探索更好的自适应检测机制，以减少手动调参的需求；或许可以采用用于异常检测的神经网络（Pang等人，2021年）。此外，我们的方法针对流水线并行中特有的阶段间攻击，但去中心化训练仍易受到其他威胁，包括后门攻击（Li 等人，2024年）和成员推理（Shokri等人，2017年），这些威胁各自为真正开放的协作训练带来了不同的挑战。关于这些问题的全面讨论，请参见第B 节。

参 考 文 献

Thalaiyasingam Ajanthan、Sameera Ramasinghe、Yan Zuo、Gil Avraham 和 Alexander Long。用于异步流水线并行优化的 Nesterov方法。载于国际机器学习大会（*ICML*）论文集，2025年。

Arasu Arun、Adam St. Arnaud、Alexey Titov、Brian Wilcox、Viktor Kolobaric、Marc Brinkmann、Oguzhan Ersoy、Ben Fielding 和 Joseph Bonneau。Verde：通过经评审的委托验证机器学习程序。*CoRR*，abs/2502.19405，2025年。

Gilad Baruch、Moran Baruch 和 Yoav Goldberg。《一点就够了：绕过分布式学习的防御》。载于神经信息处理系统年会（*NeurIPS*）论文集，第 8632–8642 页，2019。

Xiao Bi、Deli Chen、Guanting Chen、Shanhuang Chen、Damai Dai、Chengqi Deng、Honghui Ding、Kai Dong、Qiushi Du、Zhe Fu、Huazuo Gao、Kaige Gao、WenjunGao、Ruiqi Ge、Kang Guan、DayaGuo,

Jianzhong Guo、Guangbo Hao、Zhewen Hao、Ying He、Wenjie Hu、Panpan Huang、Erhang Li、Guowei Li、Jiashi Li、Yao Li、Y. K. Li、Wenfeng Liang、Fangyun Lin、Alex X. Liu、Bo Liu、Wen Liu、Xiaodong Liu、Xin Liu、Yiyuan Liu、Haoyu Lu、Shanghao Lu、Fuli Luo、Shirong Ma、Xiaotao Nie、Tian Pei、Yishi Piao、Junjie Qiu、Hui Qu、Tongzheng Ren、Zehui Ren、Chong Ruan、Zhangli Sha、Zhihong Shao、Junxiao Song、Xuecheng Su、Jingxiang Sun、Yaofeng Sun、Minghui Tang、Bingxuan Wang、Peiyi Wang、Shiyu Wang、Yaohui Wang、Yongji Wang、Tong Wu、Y. Wu、Xin Xie、Zhenda Xie、Ziwei Xie、Yiliang Xiong、Hanwei Xu、R. X. Xu、Yanhong Xu、Dejian Yang、Yuxiang You、Shuiping Yu、Xingkai Yu、B. Zhang、Haowei Zhang、Lecong Zhang、Liyue Zhang、MingchuanZhang、Minghua Zhang、WentaoZhang、Yichao Zhang、Chenggang Zhao、Yao Zhao、Shangyan Zhou、Shunfeng Zhou、Qihao Zhu和Yuheng Zou。《DeepSeek 大语言模型：以长期主义为导向，扩展开源语言模型》。*CoRR*，abs/2401.02954，2024年。doi：10。48550/ARXIV。2401。02954。URLhttps://doi。org/10。48550/arXiv。2401。02954。

Battista Biggio、Blaine Nelson 和 Pavel Laskov。针对支持向量机的投毒攻击。发表于国际机器学习大会（*ICML*）论文集，2012年。

Peva Blanchard、El Mahdi El Mhamdi、Rachid Guerraoui 和 Julien Stainer。对抗环境下的机器学习：拜占庭容错梯度下降。发表于神经信息处理系统年会（*NeurIPS*）论文集，第 119–129 页，2017年。

Tom B. Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared Kaplan、Prafulla Dhariwal、Arvind Nee-lakantan、Pranav Shyam、Girish Sastry、Amanda Askell、Sandhini Agarwal、Ariel Herbert-Voss、Gretchen Krueger、Tom Henighan、Rewon Child、Aditya Ramesh、Daniel M. Ziegler、Jeffrey Wu、Clemens Winter、Christopher Hesse、Mark Chen、Eric Sigler、Mateusz Litwin、Scott Gray、Benjamin Chess、Jack Clark、Christopher Berner、Sam McCandlish、Alec Radford、Ilya Sutskever 和 Dario Amodei。语言模型是少样本学习者。发表于神经信息处理系统年会（*NeurIPS*）论文集，2020。

DeepSeek-AI。《DeepSeek-V3 技术报告》。*CoRR*，abs/2412.19437，2024年。网址：https://doi。org/10。48550/arXiv。2412。19437。

DeepSeek-AI、Daya Guo、Dejian Yang、Haowei Zhang、Junxiao Song、Ruoyu Zhang、Runxin Xu、Qihao Zhu、Shirong Ma、Peiyi Wang、Xiao Bi、Xiaokang Zhang、Xingkai Yu、Yu Wu、Z. F. Wu、Zhibin Gou、Zhihong Shao、Zhuoshu Li、Ziyi Gao、Aixin Liu、Bing Xue、Bingxuan Wang、Bochao Wu、Bei Feng、Chengda Lu、Chenggang Zhao、Chengqi Deng、Chenyu Zhang、Chong Ruan、Damai Dai、Deli Chen、Dongjie Ji、Erhang Li、Fangyun Lin、Fucong Dai、Fuli Luo、Guangbo Hao、Guanting Chen、Guowei Li、H. Zhang、Han Bao、Hanwei Xu、Haocheng Wang、Honghui Ding、Huajian Xin、Huazuo Gao、Hui Qu、Hui Li、Jianzhong Guo、Jiashi Li、Jiawei Wang、Jingchang Chen、Jingyang Yuan、Junjie Qiu、Junlong Li、J. L. Cai、Jiaqi Ni、Jian Liang、Jin Chen、Kai Dong、Kai Hu、Kaige Gao、Kang Guan、Kexin Huang、Kuai Yu、Lean Wang、Lecong Zhang、Liang Zhao、Litong Wang、Liyue Zhang、Lei Xu、Leyi Xia、Mingchuan Zhang、Minghua Zhang、Minghui Tang、Meng Li、Miaojun Wang、Mingming Li、Ning Tian、Panpan Huang、Peng Zhang、Qiancheng Wang、Qinyu Chen、Qiushi Du、Ruiqi Ge、Ruisong Zhang、Ruizhe Pan、Runji Wang、R. J. Chen、R. L. Jin、Ruyi Chen、Shanghao Lu、Shangyan Zhou、Shanhuang Chen、Shengfeng Ye、Shiyu Wang、Shuiping Yu、Shunfeng Zhou、Shuting Pan 和 S. S. Li。《DeepSeek-R1：通过强化学习激发大语言模型的推理能力》。*CoRR*，abs/2501.12948，2025年。

Arthur Douillard、Qixuang Feng、Andrei A. Rusu、Rachita Chhaparia、Yani Donchev、Adhiguna Kuncoro、Marc'Aurelio Ranzato、Arthur Szlam 和 Jiajun Shen。Diloco：语言模型的分布式低通信训练。*CoRR*，abs/2311.08105，2023年。

Arthur Douillard、Yanislav Donchev、Keith Rush、Satyen Kale、Zachary Charles、Zachary Garrett、Gabriel Teston、Dave Lacey、Ross McIlroy、Jiajun Shen、Alexandre Ramé、Arthur Szlam、Marc' Aurelio Ranzato 和 Paul Barham。《通过重叠通信实现流式 Diloco：迈向分布式免费午餐》。*CoRR*，abs/2501.18512，2025年。

Abhimanyu Dubey、Abhinav Jauhri、Abhinav Pandey、Abhishek Kadian、Ahmad Al-Dahle、Aiesha Letman、Akhil Mathur、Alan Schelten、Amy Yang、Angela Fan、Anirudh Goyal、Anthony Hartshorn、Aobo Yang、Archi Mitra、Archie Sravankumar、Artem Korenev、Arthur Hinsvark、Arun Rao、Aston Zhang、Aurélien Rodriguez、Austen Gregerson、Ava Spataru、Baptiste Rozière、Bethany Biron、Binh Tang、Bobbie Chern、Charlotte Caucheteux、Chaya Nayak、Chloe Bi、Chris Marra、Chris McConnell、Christian Keller、Christophe Touret、Chunyang Wu、Corinne Wong、Cristian Canton Ferrer、Cyrus Nikolaidis、Damien Allonsius、Daniel Song、Danielle Pintz、Danny Livshits、David Esiobu、Dhruv Choudhary、Dhruv Mahajan、Diego Garcia-Olano、Diego Perino、Dieuwke Hupkes、Egor Lakomkin、Ehab AlBadawy、Elina Lobanova、Emily Dinan、Eric Michael Smith、Filip Radenovic、Frank Zhang、Gabriel Synnaeve、Gabrielle Lee、Georgia Lewis Anderson、Graeme Nail、Grégoire Mialon、Guan Pang、Guillem Cucurell、Hailey Nguyen、Hannah Korevaar、Hu Xu、Hugo Touvron、Iliyan Zarov、Imanol Arrieta Ibarra、Isabel M. Kloumann、Ishan Misra、Ivan Evtimov、Jade Copet、Jaewon Lee、Jan Geffert、Jana Vranes、Jason Park、Jay Mahadeokar、Jeet Shah、Jelmer van der Linde、Jennifer Billock、Jenny Hong、Jenya Lee、Jeremy Fu、Jianfeng Chi、Jianyu Huang、Jiawen Liu、Jie Wang、Jiecao Yu、Joanna Bitton、Joe Spisak、Jongsoo Park、Joseph Rocca、Joshua Johnstun、Joshua Saxe、Junteng Jia、Kalyan Vasuden Alwala、Kartikeya Upasani、Kate Plawiak、Ke Li、Kenneth Heafield、Kevin Stone 等。《Llama 3 模型家族》。*CoRR*，abs/2407.21783，2024年。

Clement Fung、Chris J. M. Yoon 和 Ivan Beschastnikh。《联邦学习在 Sybil 场景中的局限性》。发表于攻击、入侵与防御研究国际研讨会（*RAID*）论文集，2020。

Aaron Gokaslan、Vanya Cohen、Ellie Pavlick 和 Stefanie Tellex。OpenWebText语料库。http://Skylion007。github。io/OpenWebTextCorpus，2019。

Eduard Gorbunov、Alexander Borzunov、Michael Diskin 和 Max Ryabinin。《大规模安全分布式训练》。发表于国际机器学习大会（*ICML*）论文集，第 7679–7739 页，2022。

Eduard Gorbunov、Samuel Horváth、Peter Richtárik 和 Gauthier Gidel。《方差缩减是拜占庭节点的解药：更优的收敛率、更弱的假设，以及锦上添花的通信压缩》。发表于国际学习表征会议（*ICLR*）论文集，2023年。

Lie He、Sai Praneeth Karimireddy 和 Martin Jaggi。通过重采样在异构数据集上进行拜占庭鲁棒学习。*CoRR*，abs/2006.09365，2020。

Wassily Hoeffding。有界随机变量之和的概率不等式。论文集 *o* *f* *WassilyHoeffding*，第 409–426 页，1994。

Jordan Hoffmann、Sebastian Borgeaud、Arthur Mensch、Elena Buchatskaya、Trevor Cai、Eliza Rutherford、Diego de Las Casas、Lisa Anne Hendricks、Johannes Welbl、Aidan Clark、Tom Hennigan、Eric Noland、Katie Millican、George van den Driessche、Bogdan Damoc、Aurelia Guy、Simon Osindero、Karen Simonyan、Erich Elsen、Jack W. Rae、Oriol Vinyals 和 Laurent Sifre。训练计算量最优的大语言模型。*CoRR*，abs/2203.15556，2022。

Yanping Huang、Youlong Cheng、Ankur Bapna、Orhan Firat、Dehao Chen、Mia Xu Chen、HyoukJoong Lee、Jiquan Ngiam、Quoc V. Le、Yonghui Wu 和 Zhifeng Chen。《GPipe：使用流水线并行高效训练巨型神经网络》。载于神经信息处理系统年会论文集（*NeurIPS*），第 103–112 页，2019 年。

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、Gianna Lengyel、Guillaume Bour、Guillaume Lample、Lélio Renard Lavaud、Lucile Saulnier、Marie-Anne Lachaux、Pierre Stock、Sandeep Subramanian、Sophia Yang、SzymonAntoniak、Teven Le Scao、Théophile Gervet、Thibaut

Lavril、Thomas Wang、Timothée Lacroix 和 William El Sayed。Mixtral 专家混合模型。*CoRR*，abs/2401.04088，2024年。

Sai Praneeth Karimireddy、Lie He 和 Martin Jaggi。利用历史信息进行拜占庭鲁棒优化。载于国际机器学习大会（*ICML*）论文集，第 5311–5319 页，2021年。

Sai Praneeth Karimireddy、Lie He 和 Martin Jaggi。通过分桶在异构数据集上进行拜占庭鲁棒学习。载于国际学习表征会议（*ICLR*）论文集，2022。

Andrej Karpathy。NanoGPT。https://github.com/karpathy/nanoGPT，2022。

Anastasia Koloskova、Nikita Doikov、Sebastian U. Stich 和 Martin Jaggi。非凸光滑函数的增量梯度收敛性。载于国际机器学习大会（*ICML*）论文集，2024年。

Alex Krizhevsky、Ilya Sutskever 和 Geoffrey E. Hinton。使用深度卷积神经网络进行 ImageNet 分类。*ACM* 通讯，60(6):84–90，2017年。

Quentin Lhoest、Albert Villanova del Moral、Yacine Jernite、Abhishek Thakur、Patrick von Platen、Suraj Patil、Julien Chaumond、Mariama Drame、Julien Plu、Lewis Tunstall、Joe Davison、Mario Šaško、Gunjan Chhablani、Bhavitvya Malik、Simon Brandeis、Teven Le Scao、Victor Sanh、Canwen Xu、Nicolas Patry、Angelina McMillan-Major、PhilippSchmid、SylvainGugger、Clément Delangue、Théo Matussière、Lysandre Debut、Stas Bekman、Pierric Cistac、Thibault Goehringer、Victor Mustar、François Lagunas、Alexander Rush 和 Thomas Wolf。《数据集：自然语言处理社区库》。发表于自然语言处理实证方法会议（*EMNLP*）论文集：系统演示，2021年11月。

Shen Li、Yanli Zhao、Rohan Varma、Omkar Salpekar、Pieter Noordhuis、Teng Li、Adam Paszke、Jeff Smith、Brian Vaughan、Pritam Damania 和 Soumith Chintala。PyTorch 分布式：加速数据并行训练的经验。*VLDB Endowment* 论文集，13(12):3005–3018，2020。

Yige Li、Hanxun Huang、Yunhan Zhao、Xingjun Ma 和 Jun Sun。BackdoorLLM：针对大语言模型后门攻击的综合基准测试。*CoRR*，abs/2408.12798，2024年。

Wanchao Liang、Tianyu Liu、Less Wright、Will Constable、Andrew Gu、Chien-Chin Huang、Iris Zhang、Wei Feng、Howard Huang、Junjie Wang、Sanket Purandare、Gokul Nadathur 和 Stratos Idreos。TorchTitan：面向生产级大语言模型预训练的一站式 PyTorch 原生解决方案。载于国际学习表征会议论文集（*ICLR*），2025年。

Yanli Liu、Yuan Gao 和 Wotao Yin。关于带动量的随机梯度下降的改进分析。载于神经信息处理系统年会论文集（*NeurIPS*），2020年。

Ilya Loshchilov 和 Frank Hutter。解耦权重衰减正则化。载于国际学习表征会议论文集（*ICLR*），2019。

Lin Lu、Chenxi Dai、Wangcheng Tao、Binhang Yuan、Yanan Sun 和 Pan Zhou。观点：探索基于流水线并行的去中心化训练的鲁棒性。载于国际机器学习大会论文集（*ICML*），2024年。

Vien Mai 和 Mikael Johansson。带动量的随机梯度方法在非光滑非凸优化中的收敛。载于国际机器学习大会（*ICML*）论文集，第 6630–6639 页，2020。

Grigory Malinovsky、Peter Richtárik、Samuel Horváth 和 Eduard Gorbunov。拜占庭鲁棒性与部分参与可以兼得：只需裁剪梯度差。载于神经信息处理系统年会（*NeurIPS*）论文集，2024年。

Brendan McMahan、Eider Moore、Daniel Ramage、Seth Hampson 和 Blaise Agüera y Arcas。从去中心化数据中高效通信地学习深度网络。载于人工智能与统计国际会议论文集（*AISTATS*），第 1273–1282 页，2017年。

MetaAI。Llama 4 家族：原生多模态人工智能创新新时代的开端，2025年。

El Mahdi El Mhamdi、Rachid Guerraoui 和 Sébastien Rouault。拜占庭环境下分布式学习的隐性脆弱性。载于国际机器学习大会论文集（*ICML*），第 3518–3527 页，2018年。

OpenAI。GPT-4 技术报告。*CoRR*，abs/2303.08774，2023年。

Guansong Pang、Chunhua Shen、Longbing Cao 和 Anton Van Den Hengel。用于异常检测的深度学习：综述。*ACM*计算调查（*CSUR*），54(2):1–38，2021年。

Adam Paszke、Sam Gross、Francisco Massa、Adam Lerer、James Bradbury、Gregory Chanan、Trevor Killeen、Zeming Lin、Natalia Gimelshein、Luca Antiga、Alban Desmaison、Andreas Köpf、Edward Z. Yang、Zachary DeVito、Martin Raison、Alykhan Tejani、Sasank Chilamkurthy、Benoit Steiner、Lu Fang、Junjie Bai 和 Soumith Chintala。PyTorch：一种命令式风格的高性能深度学习库。载于《神经信息处理系统年会（*NeurIPS*）论文集》，2019。

Guilherme Penedo、Hynek Kydlíˇcek、Loubna Ben allal、Anton Lozhkov、Margaret Mitchell、Colin Raffel、Leandro Von Werra 和 Thomas Wolf。FineWeb数据集：从网络中大规模提炼最优质的文本数据。载于《神经信息处理系统年会数据集与基准测试专题（*NeurIPS*）论文集》，2024年。

B. T. Polyak。加快迭代方法收敛的一些方法。苏联计算数学与数学物理，1964年。

Alec Radford、Jeffrey Wu、Rewon Child、David Luan、Dario Amodei、Ilya Sutskever 等。语言模型是无监督的多任务学习器。*OpenAI*博客，1(8):9，2019。

Colin Raffel、Noam Shazeer、Adam Roberts、Katherine Lee、Sharan Narang、Michael Matena、Yanqi Zhou、Wei Li 和 Peter J. Liu。借助统一的文本到文本转换器探索迁移学习的极限。机器学习研究期刊 *(JMLR)*，21:140:1–140:67，2020。

Shashank Rajput、Hongyi Wang、Zachary Charles 和 Dimitris S. Papailiopoulos。DETOX：一种基于冗余的框架，可实现更快且更鲁棒的梯度聚合。载于 Hanna M. Wallach、Hugo Larochelle、Alina Beygelzimer、Florence d'Alché-Buc、Emily B. Fox 和 Roman Garnett（编），神经信息处理系统年会论文集 *(NeurIPS)*，第 10320–10330 页，2019。

Sameera Ramasinghe、Thalaiyasingam Ajanthan、Gil Avraham、Yan Zuo 和 Alexander Long。子空间网络：通过通信高效的模型并行缩放去中心化训练。载于神经信息处理系统年会（*NeurIPS*）论文集，2025年。

Ahmad Rammal、Kaja Gruntkowska、Nikita Fedin、Eduard Gorbunov 和 Peter Richtárik。面向拜占庭鲁棒学习的通信压缩：更高效的算法与改进的收敛率。载于人工智能与统计国际会议（*AISTATS*）论文集，第 1207–1215 页，2024年。

Herbert Robbins 和 Sutton Monro。《随机逼近法》。《数学统计年刊》，第 400–407 页，1951年。

Max Ryabinin 和 Anton Gusev。利用去中心化混合专家实现大型神经网络众包训练。载于《神经信息处理系统年会论文集（*NeurIPS*）》，2020。

Max Ryabinin、Eduard Gorbunov、Vsevolod Plokhotnyuk 和 Gennady Pekhimenko。Moshpit SGD：异构不可靠设备上的通信高效去中心化训练。载于神经信息处理系统年度大会论文集（*NeurIPS*），第 18195–18211 页，2021年。

Max Ryabinin、Tim Dettmers、Michael Diskin 和 Alexander Borzunov。SWARM 并行：训练大型模型的通信效率可能出人意料地高。载于国际机器学习大会论文集（*ICML*），第 29416–29440 页，2023年。

Noam Shazeer、Azalia Mirhoseini、Krzysztof Maziarz、Andy Davis、Quoc V. Le、Geoffrey E. Hinton 和 Jeff Dean。规模惊人的大型神经网络：稀疏门控混合专家层。载于论文集 *f* 国际学习表征会议（*ICLR*），2017年。

Reza Shokri、Marco Stronati、Congzheng Song 与 Vitaly Shmatikov。针对机器学习模型的成员推理攻击。载于《*IEEE* 安全与隐私研讨会论文集》（*SP*），第 3–18 页，2017年。

Yuandong Tian、Yiping Wang、Beidi Chen 与 Simon S Du。扫描与快照：理解单层 Transformer 中的训练动态与词元组成。载于《神经信息处理系统年会论文集》（*NeurIPS*），2023年。

John W. Tukey。探索性数据分析。Addison-Wesley 行为科学丛书：定量方法。Addison-Wesley，1977年。

Stephen M. Walker II。我们所知道的关于 GPT-4 的一切，2023年。网址 https://klu.ai/blog/gpt-4-llm。于2025年5月访问。

An Yang、Baosong Yang、Beichen Zhang、Binyuan Hui、Bo Zheng、Bowen Yu、Chengyuan Li、Dayiheng Liu、Fei Huang、Haoran Wei、Huan Lin、Jian Yang、Jianhong Tu、Jianwei Zhang、Jianxin Yang、Jiaxi Yang、Jingren Zhou、Junyang Lin、Kai Dang、Keming Lu、Keqin Bao、Kexin Yang、Le Yu、Mei Li、Mingfeng Xue、Pei Zhang、Qin Zhu、Rui Men、Runji Lin、Tianhao Li、Tingyu Xia、Xingzhang Ren、Xuancheng Ren、Yang Fan、Yang Su、Yichang Zhang、Yu Wan、Yuqiong Liu、Zeyu Cui、Zhenru Zhang 与 Zihan Qiu。Qwen2.5 技术报告。*CoRR*，abs/2412.15115，2024年。

Binhang Yuan、Yongjun He、Jared Davis、Tianyi Zhang、Tri Dao、Beidi Chen、Percy Liang、Christopher Ré 和 Ce Zhang。《异构环境中基础模型的去中心化训练》。载于神经信息处理系统年会（*NeurIPS*）论文集，2022年。

Ruiqi Zhang、Spencer Frei 和 Peter L. Bartlett。经过训练的 Transformer 在上下文中学习线性模型。机器学习研究期刊（*JMLR*），25:49:1–49:55，2024年。

Yanli Zhao、Andrew Gu、Rohan Varma、Liang Luo、Chien-Chin Huang、Min Xu、Less Wright、Hamid Shojanazeri、Myle Ott、Sam Shleifer、Alban Desmaison、Can Balioglu、Pritam Damania、Bernard Nguyen、Geeta Chauhan、Yuchen Hao、Ajit Mathews 和 Shen Li。PyTorch FSDP：全分片数据并行缩放实践。*VLDB Endowment* 论文集，16(12):3848–3860，2023年。

Tongtian Zhu、Wenhao Li、Can Wang 和 Fengxiang He。DICE：去中心化学习中的数据影响级联。载于国际学习表征会议（*ICLR*）论文集，2025年。

附录

附录的组织结构如下：

- 在第A节中，我们回答若干常见问题，希望澄清在理解我们的工作及其局限性时可能产生的误解。
- 第B 节从整体视角介绍了与去中心化训练相关、但超出当前工作范围的漏洞。
- 我们提供了与SENTINEL相关工作的扩展版本，见第C节，该部分因篇幅限制而未纳入正文。
- 我们在第D节中介绍方法细节。
- 第E节介绍了我们对 SENTINEL 的理论分析，包括在宽松假设下的收敛性分析，以及随机工作节点分配可能使每个阶段都形成诚实多数的条件。
- 在 F.2 节中，我们提供了详细的实验结果。
- 最后，在 G 节中，我们逐步介绍如何将 SENTINEL与 SWARM 并行相结合（Ryabinin 等人，2023年）。我们全面研究了如何与协调 SWARM 训练的训练器节点无缝集成，随后在 128 个不可信工作节点上开展真实场景实验，训练大语言模型并采用恶意攻击。

A. 常见问题（常见问题）

Q0：数据并行（DP）和流水线并行（PP）设置之间有哪些根本区别，导致现有拜占庭容错文献不适用于本研究？ 我们清晰地比较了数据并行和流水线并行设置，并阐明现有拜占庭容错文献为何不适用于我们的工作：

表10：数据并行与流水线并行对比

方面	数据并行（先前工作）	流水线并行（我们的工作）
模型分布	每个工作节点均持有完整的模型副本	模型在工作节点之间拆分（层/阶段）
数据分布	每个工作节点处理不同的批次	同一批次流经流水线
通信模式	聚合参数梯度	依次传递的激活值/梯度
拜占庭威胁	被篡改的参数梯度	被篡改的阶段间激活值/梯度
检测目标	恶意梯度贡献	恶意激活值/梯度传输
聚合方法	鲁棒梯度聚合	每个阶段的顺序验证
文献关注点	鲁棒聚合器	不存在非平凡的先前验证工作

因此，关键区别在于：

- **先前的拜占庭容错文献**：通过为多个模型副本的参数梯度开发鲁棒聚合方法，保障 DP 轴的安全
- **我们的工作**：使用 SENTINEL 验证在串行流水线阶段之间传输的激活值和激活梯度，从而保障正交 PP 轴的安全。

因此，现有的拜占庭容错工作不适用于 PP 轴，因为：

1. **无法进行聚合**：在流水线并行中，来自不同工作节点的激活值无法聚合（它们代表不同层处理不同的数据批次）。2. **串行依赖**：每个阶段都依赖于前一阶段的输出，因此无法进行鲁棒聚合。

3. **不同的威胁模型**：恶意工作节点会篡改中间表示，而非最终参数更新。4. **验证与聚合**：我们的验证器监控通信信道，而不是聚合多方贡献。

这一根本差异解释了为何我们的威胁模型和验证方法必然不同于“典型的拜占庭容错”文献。

Q1：为什么模型所有者会将模型的一部分交给不受信任的实体进行训练？训练大语言模型所需的计算资源正变得越来越难以持续承担。正如Brown 等人(2020) 报告的那样，训练一个 GPT-3 175B 模型最多需要 3.6 千万亿次浮点运算日，按 AWS 定价总成本高达 400 万美元。如果没有大型企业提供开源模型，去中心化训练便为这类模型的训练提供了另一种解决方案：模型可以在民主化环境中开放训练，参与者则根据各自的贡献获得报酬。正如Yuan 等人(2022) 所讨论的那样，消费设备 GPU 正在全球范围内日益普及，其中许多设备未得到充分利用。“如果我们能在基础模型训练中，以去中心化开放志愿模式利用这些设备，这将成为一种**革命性的替代方案**，取代数据中心所提供的昂贵解决方案。”

Q2：假设恶意工作节点占比为 25-37.5% 是否现实？为什么有人会信任这样的系统？我们理解大家对恶意工作节点占比达到 25-37.5% 的直观担忧。不过需要指出的是，这些假设是拜占庭容错文献中的标准做法，近期该领域的研究也证明了这一点，详见表 11。重要的是，我们的工作旨在提供一种预防性安全机制。我们的目标是通过展示强大的检测能力来遏制恶意行为，而不是假定实际中必然会出现如此高的占比。如图 3b 所示，当恶意节点占比较低时，我们的检测准确率接近 100 %。最坏情况分析可确保系统在极端对抗性条件下仍具有可靠性。

表11：拜占庭容错文献中的恶意工作节点占比

参考资料	投稿平台	恶意节点（%）
Mhamdi 等人(2018年)	ICML 2018	47.37
Gorbunov 等人(2022)	ICML 2022	43.75
Blanchard 等人(2017)	NeurIPS 2017	33.00
Karimireddy 等人 (2021)	ICML 2021	30.55
Malinovsky 等人 （2024）	NeurIPS 2024	25.00
Gorbunov 等人（2023年）	ICLR 2023	20.00
Karimireddy 等人 （2022）	ICLR 2022	20.00
Rammal 等人(2024年)	AISTATS 2024	18.75

Q3：在分布式框架中集成验证节点有多实用？在部署我们算法（例如 SWARM）的真实系统中，训练器节点负责在各层之间传输激活值/激活梯度。我们提议改造这些现有的训练器节点，使其执行验证；由于它们本就能访问各层之间传递的所有信号，因此基本上可以“免费”获得这项安全功能。关键在于，训练器节点在分布式系统中承担由中心控制的角色，因为它们由网络协调器而非志愿者管理，因此在经济上可行，能够对其进行控制。由于其主要职责是协调信号传输，这些节点是轻量级的基于 CPU 的节点，成本极低。例如，考虑训练一个 4B 参数的大语言模型，其中每个层/阶段大约需要 18GB 的 GPU 显存。在 AWS 上，每个工作节点都需要一台配备 24GB 显存的 g5.2xlarge EC2 实例，费用约为每小时 1.212 美元。相比之下，训练器节点可以按每台 c5a.8xlarge 实例（32 个 vCPU，每个训练器占用 4 个 vCPU）捆绑 8 个节点，每小时费用为 1.232 美元（即每个训练器每

个训练器节点每小时）。与工作节点实例相比，这大幅降低了成本；并且由于训练器由中心控制，网络协调者在承担这些成本的同时维持安全性和可靠性，在经济和运营层面都是可行的。如果必须将训练器角色委派给志愿者，则需要适当的身份验证机制来防止恶意行为，这部分工作留待未来研究。有关在 SWARM 中实现 S哨兵的更深入讨论，请参阅第G节。

Q4：为什么假设存在一段只有诚实工作节点参与的“预热”时期？这是否有悖于分布式训练的精神？ 在分布式环境中，通常会先在受控环境中启动训练，直到训练趋于稳定，然后再允许公共工作节点加入。这种做法不仅对我们的验证方法至关重要，也能确保在公共工作节点参与之前训练保持稳定。这并不违背分布式训练的精神：设想一个用于去中心化训练的 40 层大语言模型。起初，我们在自己控制的 40 个节点上部署一个副本，但训练吞吐量有限。在最初的预热阶段之后，我们将流水线复制到另外 7 个副本中，使数据并行工作节点总数达到 320 个，其中只有 12.5% 由模型所有者控制，87.5% 是公共工作节点。请注意，在每个阶段至少保留一个可信工作节点至关重要，这不仅是为了初始预热，也是出于系统可靠性的考虑，因为志愿者节点（在我们的示例中有 280 个）可能会在运行过程中掉线。我们需要至少有一条可靠的流水线，以确保志愿者退出时训练能够持续进行。

Q5：向验证器进行 N到1 通信会造成带宽瓶颈吗？ 不会造成带宽瓶颈。在 SWARM 等模型并行实现中，各阶段之间通过训练节点进行协调，这些节点负责发送和接收激活值/梯度。由于每个训练节点都能观察到从一个阶段传递到另一个阶段的所有信号，因此我们将其用作可信的验证器节点。正如我们在第 G 节中详细讨论的那样，SWARM 利用多个训练节点逐一将数据传递到各个阶段。因此，每个训练器都可以在其他训练器向工作节点发送信号的同时，使用 S ENTINEL 独立运行验证。有关训练节点操作的技术细节，请参阅 Hivemind（Ryabinin 和 Gusev，2020）库。

Q6：验证机制如何处理假阳性？ 违规计数器配合宽恕机制，是处理暂时性异常并避免不公平地封禁诚实工作节点的巧妙方法。训练曲线见第 8 节和第 16 节的第 F 节。如图所示，训练/验证曲线表明，用 EMA 值暂时替换提交的信号，对训练动态没有明显影响。这很符合直觉：当只有少数节点经历 EMA 替换阶段时，数据并行设置中的其余工作节点会继续提交有用的信号，从而有效引导训练。

Q7：为什么训练中断攻击的有效性会因不同的攻击版本和目标而有所不同？ 在比较基于激活值的攻击与基于梯度的攻击时，有两个关键因素可以解释有效性差异：

- **幅度差异：** 激活值比激活梯度大几个数量级。因此，修改激活值的方法可以在攻击过程中实现更大的扰动幅度，从而造成更成功的破坏。
- **传播范围：** 在第 $1 \leq \ell \leq L$ 层进行激活操纵会影响所有后续层（$[\ell+1, \ell+2, \ldots, L]$）的前向传播以及所有层的反向传播。相比之下，在第 ℓ 层进行梯度操控仅会影响前序层 $[1, 2, \ldots, \ell-1]$的梯度。因此，如果攻击者位于网络中部，操纵激活值会对整个训练过程产生更广泛的影响。

问题 8：你们的收敛率与拜占庭容错文献中的著名下界相比如何？ 在我们的假设下，我们的收敛保证给出了准确的界。我们注意到一个

关键区别在于我们的设定：以往的拜占庭容错文献研究的是恶意行为者修改“参数梯度”的数据并行训练。我们的工作则关注与之正交的 流水线并行维度，在该维度中，各层“之间”共享激活值和梯度，因此需要进行验证。这两个维度相辅相成：在去中心化环境中，确保流水线并行和数据并行两个维度的安全都很重要，而本工作关注前者。因此，拜占庭容错文献中的既有保证无法直接比较。

Q9：你们的方法与Lu 等人（2024年）的先前工作相比如何？ Lu 等人（2024年）采用了一种朴素方法，即为每个工作节点分配一个重复副本以进行验证。例如，对于 320 个工作节点，需要将它们分成两组：160 个工作节点执行计算，160 个工作节点复制其工作以进行验证。虽然该方法在所有攻击上均达到 100% 的 F1 分数，但其运行吞吐量仅为真实分布式网络吞吐量的一半。我们的方案不会复制志愿者节点的工作，从而使训练速度提高一倍。从概念上讲，在 OpenWebText 上针对混合攻击，我们将得到：

表 12：与先前工作在 OpenWebText 混合攻击上的比较

方法	度量指标			
	PR. (%) ↑	RE. (%) ↑	F1 分数 (%) ↑	每秒事务数 ↑
重复工作 （Lu 等人，2024年）	100.0	100.0	100.0	6483
SENTINEL（我们的方法）	91.9	87.2	89.5	12966

Q10：从训练运行中移除节点会降低吞吐量吗？请注意，在 SWARM 等真实世界生态系统中（Ryabinin 等人，2023年），我们不再使用固定网格。工作节点可以随意加入和离开 SWARM，并且每个流水线阶段都有许多工作节点为该阶段提供服务。结合 SWARM 中的随机路由（详情请参阅第 G.3 节），可以确保工作节点达到最大利用率。因此，在每个阶段都有大量工作节点提供服务的真实世界系统中，移除工作节点不一定会降低吞吐量。

B. 使用数据并行和流水线并行的去中心化训练的脆弱性

在用于协作训练的去中心化环境中，验证参与者工作节点对于维护训练过程的完整性、安全性和整体有效性至关重要（Lu 等人，2024年）。验证是一项关键的质量控制措施，可确保每位参与者都对集体训练工作作出实质性贡献。若缺乏充分的验证机制，恶意行为者可能渗入 SWARM，从而危及其完整性。这些攻击者可能扰乱协作训练、降低训练效率，或在未真正作出贡献的情况下访问训练好的模型并从中非法获益。

下面，我们对去中心化协作训练场景中可能出现的常见恶意行为进行分类。这些类别并非互斥，因为攻击者可能同时采用多种策略。尽管如此，这种分类仍对主要威胁进行了系统概述：

- **训练中断（拒绝服务或 DoS）**：攻击者故意阻碍或中止训练过程。这可能通过丢弃关键更新、引入旨在破坏通信协议的恶意数据，或用过量或无关的提交使系统不堪重负来实现。
- **搭便车或最小努力贡献**：参与者仅贡献极少的计算 effort 或数据，却试图从集体过程中获益，例如获取最终模型、领取奖励或提升自身声誉（Zhu 等人，2025年）。常见手段包括提交无实质性更新，或策略性地保持不活跃，直至训练接近完成。

- **模型投毒与后门攻击**：恶意行为者提供对抗性更新，旨在使所得模型产生隐蔽漏洞或针对性异常行为（Li 等人，2024年）。这些后门或受损模型通常在正常运行条件下被隐藏，仅在特定的预定义场景下才会触发恶意结果。
- **隐私侵犯（数据提取或推理攻击）**：攻击者在训练过程中利用梯度、激活值或其他共享信息，从其他参与者的数据集中推断敏感或隐私信息，从而破坏机密性并危及用户隐私（Shokri等人，2017年）。
- **声誉或信用操纵**：参与者故意伪造或夸大自身贡献（例如，生成看似高质量的更新），以不正当地获取更多奖励、更高声誉或代币。这种操纵行为会损害系统的公平性，并破坏诚实贡献者之间的信任。

本文主要关注如何缓解与训练中断相关的威胁。确保识别并排除提交有害或具有破坏性更新的恶意参与者至关重要。若未能有效应对这些威胁，群体将无法实现模型收敛，也无法生成可靠且可用的最终模型。

C. 扩展相关工作

随着去中心化机器学习系统的普及，安全分布式训练受到了广泛关注。我们的工作建立在多个研究方向的基础上，同时应对大语言模型训练中流水线并行所带来的独特挑战。

去中心化训练框架。 去中心化训练已成为普及人工智能能力的一种有前景的方法。 Ryabinin 和 Gusev (2020) 推出了 Hivemind 框架，使混合专家模型能够以去中心化方式进行训练。在此基础上，Ryabinin 等人 (2021年) 提出了 Moshpit SGD，这是一种面向异构且不可靠设备的高通信效率训练算法。与此同时，Yuan 等人 (2022) 推出了 Tasklets，这是一个适用于异构环境的去中心化训练系统，能够适应不断变化的网络条件和计算能力。SWARM 并行（Ryabinin 等人，2023年）进一步改进了这一方法，通过结合流水线并行和数据并行，使模型训练规模远超以往去中心化方法所能达到的水平。虽然这些框架优先保障系统免受非恶意故障影响，但通常缺乏针对对抗性参与者的防护；在开放的去中心化训练环境中，这是一个关键漏洞。

拜占庭容错分布式训练。分布式学习中的拜占庭容错已在联邦学习（McMahan 等人，2017年）和数据并行训练（Li 等人，2020）的背景下得到广泛研究。Blanchard 等人（2017年）提出了 Krum，这是首个用于分布式随机梯度下降的拜占庭容错聚合规则，能够抵御受损工作节点实施的任意梯度操纵。此后又出现了更为成熟的方法，包括 Bulyan（Mhamdi 等人，2018年）、基于中位数的聚合（Baruch 等人，2019）以及基于裁剪的方法（He 等人，2020；Malinovsky 等人，2024年）。

一种截然不同的方法称为 CENTEREDCLIP，由 Karimireddy 等人（2021年）提出。该方法利用历史梯度信息检测异常更新——其理念与我们的基于动量的验证相似，但专门应用于梯度聚合。Rammal 等人近期的研究Rammal 等人（2024年）表明，通信压缩可以与拜占庭鲁棒学习有效结合，在维持安全保证的同时提高收敛率。

尽管这些方法提供了强有力的理论保证，但它们主要针对工作节点独立计算完整梯度的场景，因此并不适用于 SWARM 这类在各阶段之间传递中间激活值的流水线并行配置。此外，这些方法通常涉及比较不同工作节点之间的梯度，这将需要跨

阶段进行参数复制，这与流水线并行的目标相矛盾，即支持训练过大而无法放入单个设备的模型。

流水线并行架构中的安全性。 与其他分布式范式相比，针对流水线并行训练的安全性考量受到的关注有限。 Lu 等人 (2024年)最近发表了一篇立场论文，探讨基于流水线并行的去中心化训练所面临的鲁棒性挑战，并强调基于激活值的攻击是一个关键问题。然而，他们的工作主要着眼于识别漏洞，而未提出全面的验证方案。Lu 等人Lu 等人(2024年)和Rajput 等人(2019)提出的基于冗余的方法原则上可以调整后用于去中心化流水线并行场景。然而，这些方法会带来显著的计算开销（因为需要在多个工作节点上重复计算），从而大幅削弱去中心化训练的可扩展性优势。

D. 基于动量的验证详情

D.1 基于动量的验证算法

本节将详细介绍我们在去中心化训练中用于工作节点验证的算法。算法 1概述了该场景下端到端的验证机制。我们按照训练的进程，依时间顺序介绍该算法。验证节点负责执行所有验证操作，而工作节点仅负责在前向传播期间计算激活值，并在反向传播期间计算各自的梯度。算法2和 3分别详细说明前向与反向传播的验证流程。算法 4介绍了我们缓解级联效应的方法，如第3.1节所述。最后，算法5 说明了我们为该方法中的各项度量指标设定自适应四分位距阈值的方法。

D.2 级联效应的处理

与传统联邦学习（McMahan 等人，2017年）方法相比，流水线并行具有独特的架构特征。一个关键挑战是我们所说的“级联效应”，它仅出现在流水线并行中。在前向传播期间，单个节点提交恶意激活值，就可能污染所有后续激活值，进而导致下游验证节点错误地将良性节点标记为恶意节点（见图 5a）。如图 6b所示，反向传播期间也会出现类似现象。级联效应可能会显著提高误报率，因此解决这一挑战至关重要。

为缓解这一问题，我们的方法（详见第 3.1 节）实现了节点间通信协议。具体而言，验证节点会维护一份“受污染”列表，跟踪被判定为可能存在恶意行为的上游节点，并将这些信息传达给后续验证器，以防它们更新指数移动平均值，进而错误地将受攻击者影响的下游节点标记为异常。在反向传播期间，与受损节点具有相同数据并行秩的所有节点都会接收梯度动量，而非实际梯度。在整个验证过程中，工作节点会以稳定的速度持续处理数据，确保网络中没有节点察觉到异常的行为模式。

级联效应会沿两个传播方向显现，因为恶意行为既可以针对激活信号，也可以针对梯度信号。我们通过两种相应的缓解策略来应对这一双向漏洞：

- 当激活值受到损害时，所有受影响的节点都会接收梯度动量（见图 5）；
- 当反向传播过程中出现恶意行为时，所有下游节点都会切换为激活梯度（见图 6）。

虽然另一种方法可以是发送零向量作为梯度，但这实际上会导致停滞

算法 1 面向 SWARM 并行的基于动量的验证**要求：** 参数 $\beta_h, \beta_g \in (0,1)$，违规阈值 c，宽限期 $T_{\text{forgiveness}}$，度量指标集合 s $\mathcal{M}$

1: 初始化 $\boldsymbol{m}_0^{(s)}(\boldsymbol{h}) = \mathbf{0}, \boldsymbol{m}_0^{(s)}(\boldsymbol{g}) = \mathbf{0}, v_r^{(s)} = 0, B_s = \emptyset, \mathcal{H}_l^{(s)} = \emptyset, \mathcal{G}_l^{(s)} = \emptyset$ 对于所有 s, r *2:* 初始化 $\mathcal{T} = \emptyset$// 初始化全局污染集合 *3:* // 预热阶段以建立基线统计数据 *4: for* $t = 1$ 到 T_{warmup} *do 5: for* $s \in \{1, 2, \ldots, p\}$ *do 6:* 从所有工作节点收集 $\boldsymbol{h}_t^{(s,r)}$ 和 $\boldsymbol{g}_t^{(s,r)}$ $r \in \{1, \ldots, d\}$ *7:* 计算 $\Gamma_t^{(s,r)} = \Omega(\boldsymbol{h}_t^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{h}))$ $\forall r$，添加到 $\mathcal{H}_l^{(s)}$ *8:* 计算 $\Gamma_t^{(s,r)} = \Omega(\boldsymbol{g}_t^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{g}))$ $\forall r$，添加到 $\mathcal{G}_l^{(s)}$ *9:* 使用公式 *(1)* 更新动量 $\boldsymbol{m}_t^{(s)}(\boldsymbol{h})$ 和 $\boldsymbol{m}_t^{(s)}(\boldsymbol{g})$ *10: end for 11: end for 12:* // 带验证的主训练阶段 *13: for* $t = T_{\text{warmup}} + 1$ 到 T_{total} *do 14:* $\mathcal{T}_t = \emptyset$ // 为当前迭代初始化污染集合 *15:* // 步骤 *1*：前向传播与激活验证 *16: for* $s \in \{1, \ldots, p\}$ *do 17:* $\mathcal{T}(s)_t \leftarrow$激活验证$(s, t, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{h}), \mathcal{H}_l^{(s)}, B_s, v_r^{(s)}, \mathcal{M}, c, T_{\text{forgiveness}}, \mathcal{T}_t)$ *18:* $\mathcal{T}_t \leftarrow \mathcal{T}_t \cup \mathcal{T}(s)_t$ // 累积受污染工作节点 *19:* $R_{\text{clean}} = \{r : (t, s, r) \notin \mathcal{T}_t\}$ *20:* $\boldsymbol{m}_t^{(s)}(\boldsymbol{h}) = \beta_h \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{h}) + (1 - \beta_h) \frac{1}{|R_{\text{clean}}|} \sum_{r \in R_{\text{clean}}} \boldsymbol{h}_t^{(s,r)}$ *21: end for 22: for* $s \in \{p, p-1, \ldots, 1\}$ *do 23:* // 步骤 *2*：反向传播与梯度验证 *24:* $\mathcal{T}(s)_t \leftarrow$梯度验证$(s, t, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{g}), \mathcal{G}_l^{(s)}, B_s, v_r^{(s)}, \mathcal{M}, c, T_{\text{forgiveness}}, \mathcal{T}_t)$ *25:* $\mathcal{T}_t \leftarrow \mathcal{T}_t \cup \mathcal{T}(s)_t$ // 累积受污染工作节点 *26:* // 步骤 *3*：针对受污染工作节点的梯度替换 *27:* $\boldsymbol{g}_t^{(s)} \leftarrow$梯度替换$(s, t, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{g}), \boldsymbol{g}_t^{(s)}, \mathcal{T}_t)$ *28:* $R_{\text{clean}} = \{r : (t, s, r) \notin \mathcal{T}_t\}$ *29:* $\boldsymbol{m}_t^{(s)}(\boldsymbol{g}) = \beta_g \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{g}) + (1 - \beta_g) \frac{1}{|R_{\text{clean}}|} \sum_{r \in R_{\text{clean}}} \boldsymbol{g}_t^{(s,r)}$ *30: end for 31:* $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_t$ // 跨迭代次数累积受污染条目 *32: end for 33:* 返回：$\mathcal{T}$ // 返回完整的受污染工作节点-阶段-迭代元组集合

算法 2 激活验证 **要求：** $s, t, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{h})$ $\mathcal{H}_l^{(s)}, B_s, v_r^{(s)}, \mathcal{M}, c, T_{\text{forgiveness}}, \mathcal{T}_t$ **1:** $\mathcal{T}(s)_t = \emptyset$ // 为当前阶段初始化污染集合 **2:** 截断 $\mathcal{H}_l^{(s)} \leftarrow \mathcal{H}_l^{(s)}[-l:]$ 结束 **3:** 从所有 $r \in \{1, \ldots, d\} \setminus B_s$ 收集 $\boldsymbol{h}_t^{(s,r)}$ **4: for** $r \in \{1, \ldots, d\} \setminus B_s$ 不在 $\mathcal{T}_t$ **do 5:** 计算度量指标：$\Gamma_t^{(s,r,i)} = \Omega_i(\boldsymbol{h}_t^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{h}))$ 对于 $i \in \mathcal{M}$ **6: if** $\exists i \in \mathcal{M}:\ |\Gamma_t^{(s,r,i)} - q_2^{(s,i)}| \geq k^{(s,i)}$ *tukey*$(q_3^{(s,i)} - q_1^{(s,i)})$ **then 7:** $v_r^{(s)} \leftarrow v_r^{(s)} + 1$ // 递增违规计数器 **8:** $\mathcal{T}(s)_t \leftarrow \mathcal{T}(s)_t \cup \{(t, s, r)\}$ // 在当前阶段标记为受污染 **9: if** $v_r^{(s)} \geq c$ **or** $\Gamma_t^{(s,r,i)} \gg k^{(s,i)}$ *tukey*$(q_3^{(s,i)} - q_1^{(s,i)})$ **then 10:** $B_s \leftarrow B_s \cup \{r\}$ // 封禁工作节点 **11:** 通知阶段 $s' > s$ 以标记受影响的小批次 **12: end if 13: else 14:** 将 $\Gamma_t^{(s,r,i)}$ 添加到 $\mathcal{H}_l^{(s,i)}\ \forall i \in \mathcal{M}$ **15:** $v_r^{(s)} \leftarrow \max(0, v_r^{(s)} - 1)$ 若 $T_{\text{forgiveness}}$ 连续干净步数 **16: end if 17: end for 18:** 更新四分位距统计量并使用算法 5 调整 $k^{(s,i)}$ *tukey* $\forall i \in \mathcal{M}$ **19: if** $|\mathcal{T}(s)_t| > 0.5 \cdot (d - |B_s|)$ **then 20:** $\mathcal{T}(s)_t \leftarrow \emptyset$ // 若超过 50% 被标记则清除（自然漂移，见 D.3 节） **21: end if 22:** 返回：$\mathcal{T}(s)_t$ // 返回此阶段的受污染工作节点

算法3 梯度验证 Require: $s, t, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{g})$ $\mathcal{G}_l^{(s)}, B_s, v_r^{(s)}, \mathcal{M}, c, T_{\text{forgiveness}}, \mathcal{T}_t$1:
$\mathcal{T}(s)_t = \emptyset$ // 初始化当前阶段的污染集合 2: 截断 $\mathcal{G}_l^{(s)} \leftarrow \mathcal{G}_l^{(s)}[-l:\text{end}]$
3: 污染$_{\text{downstream}} = \{r : (t, s', r) \in \mathcal{T}_t$ 对于某个 $s' > s\}$ // 在

下游阶段

4: 对于 $r \in \{1, \ldots, d\} \setminus B_s$ 执行 *5:* 如果 $r \in$ 受污染$_{\text{downstream}}$ 则 *6:* $\mathcal{T}(s)_t \leftarrow \mathcal{T}(s)_t \cup \{(t, s, r)\}$ // 标记为受下游污染 *7:* 否则 *8:* 从工作节点收集 $\boldsymbol{g}_t^{(s,r)}$ *r9:* 计算度量指标：对 $i \in \mathcal{M}$执行 $\Gamma_t^{(s,r,i)} = \Omega_i(\boldsymbol{g}_t^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{g}))$ *10:* 如果 $\exists i \in \mathcal{M}$ ： $|\Gamma_t^{(s,r,i)} - q_2^{(s,i)}| \geq k^{(s,i)}$ *Tukey*$(q_3^{(s,i)} - q_1^{(s,i)})$ 则 *11:* $v_r^{(s)} \leftarrow v_r^{(s)} + 1$ // 增加违规计数器 *12:* $\mathcal{T}(s)_t \leftarrow \mathcal{T}(s)_t \cup \{(t, s, r)\}$ // 标记为受污染 *13:* 如果 $v_r^{(s)} \geq c$ 或 $\Gamma_t^{(s,r,i)} \gg k^{(s,i)}$ *Tukey*$(q_3^{(s,i)} - q_1^{(s,i)})$ 则 *14:* $B_s \leftarrow B_s \cup \{r\}$ // 禁用工作节点 *15:* 通知各阶段 $s' < s$， 以标记受影响的小批次 *16:* 结束如果 *17:* 否则 *18:* 将 $\Gamma_t^{(s,r,i)}$ 添加到 $\mathcal{G}_l^{(s,i)}$ $\forall i \in \mathcal{M}$ *19:* 如果连续 $T_{\text{forgiveness}}$ 步均干净，则$v_r^{(s)} \leftarrow max(0, v_r^{(s)} - 1)$ *20:* 结束如果 *21:* 结束如果 *22:* 结束循环 *23:* 使用算法 *5* 更新四分位距统计量，并调整 $k^{(s,i)}$ *Tukey*$\forall i \in \mathcal{M}$ *24:* 如果 $|\mathcal{T}(s)_t| > 0.5 \cdot (d - |B_s|)$ 则 *25:* $\mathcal{T}(s)_t \leftarrow \emptyset$ // 如果超过 *50%* 被标记，则清除（ *naturalshift*，参见 *D.3* 节） *26:* 结束如果 *27:* 返回： $\mathcal{T}(s)_t$ // 返回此阶段的受污染工作节点

算法 4 梯度替换

要求： $s, t, \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{g})\ \boldsymbol{g}_t^{(s)}, \mathcal{T}_t$

1: **对于** $r \in \{1, \ldots, d\}$，**执行**
2: **如果** $(t, s, r) \in \mathcal{T}_t$，**则**
3: $\boldsymbol{g}_t^{(s,\ r)} \leftarrow \boldsymbol{m}_{t-1}^{(s)}(\boldsymbol{g})$ // 用动量替换梯度
4: **结束 if**
5: **结束 for**
6: **返回：** $\boldsymbol{g}_t^{(s)}$ // 返回更新后的梯度

在受影响的流水线中进行训练。[2] 我们将针对受污染片段探索合适梯度信号的工作留待未来开展。

D.3 关于自然分布偏移

除恶意攻击外，由于数据特征或模型动态不断演变，训练过程中也可能自然发生合法的数据分布偏移（Tian et al.，2023年；Zhang et al.，2024年）。在此情况下，同一流水线阶段的多个工作节点可能会同时表现出通常会触发恶意检测的统计偏差，尽管所有节点均在诚实地运行。

为区分自然分布偏移与协同攻击，我们在验证器层面采用基于共识的方法。当某一流水线阶段超过 50% 的节点被标记为潜在恶意节点时，验证器会将其归因于自然分布偏移，而非恶意行为。该阈值利用了诚实多数假设：涉及超过半数节点的协同攻击将违反我们的安全模型，因此此类情形无法与合法的系统范围变化区分开来。

检测到自然分布偏移后，系统将作出如下响应：

1. 训练正常继续，不启用恶意缓解协议。2. 第 D.2 节所述的级联效应机制不会被激活。3. 节点更新其指数移动平均统计量，以适应新的数据分布。

这种共识机制可确保合理的分布变化不会导致不必要的验证开销或训练中断。然而，如何在异步训练环境中完善这一方法（Ajanthan et al.，2025年），以应对节点可能在不同时间发生分布偏移的情况，仍是未来研究的重要方向。

[2] 我们将在第 G 节展示，在我们的 SWARM 实现中，这一选择不会影响收敛，因为 SWARM 采用随机路由。

算法 5自适应四分位距阈值调整

要求： 阶段 s 和度量指标 $i \in \mathcal{M}$ 的历史窗口 $\mathcal{H}_l^{(s,i)}$ （或梯度的历史窗口 $\mathcal{G}_l^{(s,i)}$ ），初始乘数 k_0、目标假阳性率 α、增长因子 $\gamma_g > 1$、收缩因子 $\gamma_s < 1$、最大迭代次数 $N_{\max}$、最小距离乘数 Λ

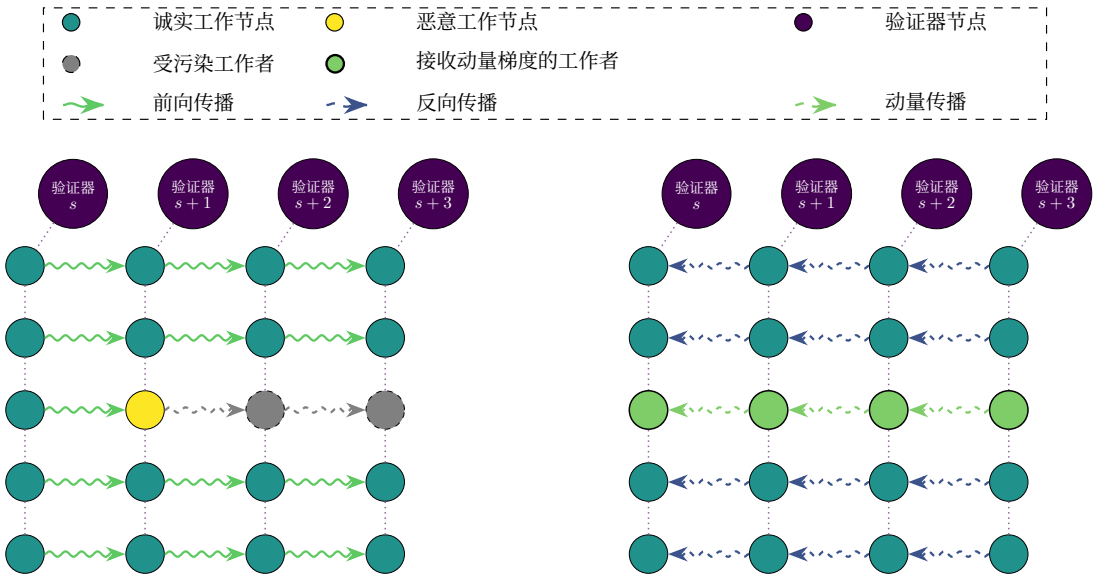
// 计算初始统计量 q_1， q_2， $q_3 \leftarrow$ 第 25、50、75 百分位数（ $\mathcal{H}_l^{(s,i)}$ ） 四分位距 $\leftarrow \max(q_3 - q_1, \ \epsilon)$ // 使用较小的值确保四分位距非零 ϵ $k \leftarrow k_0$ // 使用前一个乘数值进行初始化 $\tau_{\text{lower}} \leftarrow q_2 - k \cdot$ 四分位距 $\tau_{\text{upper}} \leftarrow q_2 + k \cdot$ 四分位距 FP-rate $\leftarrow$ 在 $\mathcal{H}_l^{(s,i)}$ 之外的比例 $[\tau_{\text{lower}}, \tau_{\text{upper}}]$

// 如果假阳性率过高，则扩大阈值 迭代 $\leftarrow 0$ **当**FP-rate $> \alpha$ 且迭代 $< N_{\max}$ **时执行** $k \leftarrow k \cdot \gamma_g$ // 增大乘数 $\tau_{\text{lower}} \leftarrow q_2 - k \cdot$ 四分位距 $\tau_{\text{upper}} \leftarrow q_2 + k \cdot$ 四分位距 FP-rate $\leftarrow$ 的比例 $\mathcal{H}_k^{(s,i)}$ 在 $[\tau_{\text{lower}}, \tau_{\text{upper}}]$之外 迭代 $\leftarrow$ 迭代 $+ 1$ **循环结束**

// 假阳性率过低时收紧阈值 迭代 $\leftarrow 0$ **当** 假阳性率 $\ll \alpha$ 且迭代 $< N_{\max}$ **执行** $k \leftarrow k \cdot \gamma_s$ // 缩小乘数 $\tau_{\text{lower}} \leftarrow q_2 - k \cdot$ 四分位距 $\tau_{\text{upper}} \leftarrow q_2 + k \cdot$ 四分位距 假阳性率 $\leftarrow$ 超出 $\mathcal{H}_l^{(s,i)}$ 的比例 $[\tau_{\text{lower}}, \tau_{\text{upper}}]$ 迭代 $\leftarrow$ 迭代 $+ 1$

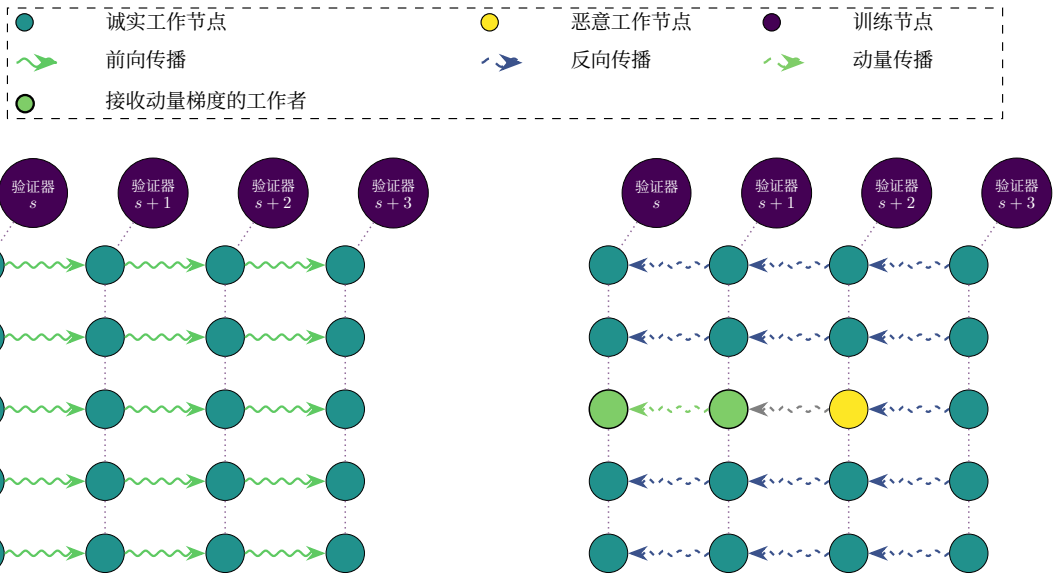
结束循环

// 根据度量指标类型强制设置与中位数的最小距离（可选） $\lambda \leftarrow \Lambda[i]$ // 获取当前度量指标的乘数 $d_{\min} \leftarrow |q_2| \cdot \lambda$ // 最小阈值距离 $\tau_{\text{lower}} \leftarrow$ 最小值$(\tau_{\text{lower}}, q_2 - d_{\min})$ $\tau_{\text{upper}} \leftarrow$ 最大值$(\tau_{\text{upper}}, q_2 + d_{\min})$ **返回：** $\tau_{\text{lower}}, \tau_{\text{upper}}, k$



(a) 前向传播。 (b) 反向传播。

图5：分布式训练期间处理受损工作节点的验证协议。在前向传播期间，阶段 $s+1$ 上的工作节点被检测为可能已受损（以黄色表示）。验证器节点继续将激活值转发至后续阶段，而不向下游工作节点发出警报，以避免中断流水线。在反向传播期间，验证器节点不转发受损工作节点计算出的梯度，而是以梯度动量值替代，以维持训练稳定性。通信通过相邻流水线阶段之间的验证器节点进行，但为清晰起见，图中显示的是工作节点之间的直接箭头。



(a) 前向传播。 (b) 反向传播。

图6：在梯度传播过程中处理受损工作节点的验证协议。在反向传播期间，阶段 $s+2$ 中的工作节点被检测为可能已受损（以黄色显示）。为防止受污染的梯度继续传播，验证节点会用梯度动量值替代所有前序阶段（$s+1$、 s等）中工作节点的梯度动量值，而不是转发损坏的梯度。这既能确保训练稳定性，又能维持流水线运行，同时不向下游工作节点透露受损情况。

E. 理论保证及其证明

E.1 收敛性分析

在本节中，我们给出完整的收敛性分析。需要注意的是，我们所推导的界绝非可能达到的最紧界。相反，我们的目标是在基于动量的验证与训练动态之间建立数学联系。

E.1.1 恶意检测界

我们的首要目标是确定恶意工作节点在不被检测的情况下所能引入的最大扰动的界。我们首先分析动量平滑如何影响激活向量中的全局偏差。[3]

引理 E.1 (动量平滑限制全局偏差)。 设阶段 s 和迭代 t 的激活向量动量更新为

$$\boldsymbol{m}_t^{(s)} = \beta_h \, \boldsymbol{m}_{t-1}^{(s)} + (1-\beta_h)\left(\frac{1}{d}\sum_{r=1}^{d} \boldsymbol{h}_t^{(s,r)}\right), \qquad 0 \le \beta < 1, \tag{8}$$

其中， d 表示每个阶段的工作节点副本数量。假设：

1. 工作节点中有 $\gamma_s < \frac{1}{2}$ 的比例为恶意工作节点，其中 $B_s \subset \{1,2,\dots,d\}$ 表示恶意工作节点的子集。*2.* 恶意工作节点添加一个向量扰动 $\boldsymbol{\delta}_t^{(s,r)}$ ，满足

$$\|\boldsymbol{\delta}_t^{(s,r)}\| \le \varepsilon. \tag{9}$$

那么，恶意扰动导致的动量偏差满足

$$\boxed{\|\Delta \boldsymbol{m}_t^{(s)}\| \le \gamma_s \, \varepsilon}, \tag{10}$$

其中， $\Delta \boldsymbol{m}_t^{(s)}$ 是使用恶意扰动计算的动量与仅使用未受扰动的（诚实）激活值计算的动量之间的差值。

证明。 设 $H_s = \{1,2,\dots,d\} \setminus B_s$ 表示阶段 s 的诚实工作节点集合。由于恶意工作节点所占比例至多为 γ_s：

$$|B_s| = \gamma_s d, \qquad |H_s| = (1-\gamma_s)d. \tag{11}$$

我们可以将每个工作节点的激活向量表示为

$$\boldsymbol{h}_t^{(s,r)} = \boldsymbol{h}_{t,\text{nom}}^{(s,r)} + \boldsymbol{e}_t^{(s,r)}, \quad \text{where} \quad \boldsymbol{e}_t^{(s,r)} = \begin{cases} \boldsymbol{\delta}_t^{(s,r)}, & r \in B_s, \\ \mathbf{0}, & r \in H_s. \end{cases} \tag{12}$$

标称（未扰动）平均激活可表示为：

$$\bar{\boldsymbol{h}}_t^{(s)} = \frac{1}{d}\sum_{r=1}^{d} \boldsymbol{h}_{t,\text{nom}}^{(s,r)}. \tag{13}$$

[3]尽管我们针对激活操纵提出了理论，但我们的结果同样很容易推广到梯度操控。

对于观测到的（已扰动）平均值，我们有：

$$\begin{aligned}\hat{\boldsymbol{h}}_t^{(s)} &= \frac{1}{d}\sum_{r=1}^{d}\big(\boldsymbol{h}_{t,\mathrm{nom}}^{(s,r)} + \boldsymbol{e}_t^{(s,r)}\big) \\ &= \bar{\boldsymbol{h}}_t^{(s)} + \frac{1}{d}\sum_{r\in B_s}\boldsymbol{\delta}_t^{(s,r)}.\end{aligned} \tag{14}$$

对于动量项，我们可以写为：

$$\begin{aligned}\boldsymbol{m}_{t,\mathrm{obs}}^{(s)} &= \beta_h \boldsymbol{m}_{t-1,\mathrm{obs}}^{(s)} + (1-\beta_h)\hat{\boldsymbol{h}}_t^{(s)}, \\ \boldsymbol{m}_{t,\mathrm{nom}}^{(s)} &= \beta_h \boldsymbol{m}_{t-1,\mathrm{nom}}^{(s)} + (1-\beta_h)\bar{\boldsymbol{h}}_t^{(s)}.\end{aligned} \tag{15}$$

第 t 次迭代时的动量偏差为：

$$\begin{aligned}\Delta\boldsymbol{m}_t^{(s)} &= \boldsymbol{m}_{t,\mathrm{obs}}^{(s)} - \boldsymbol{m}_{t,\mathrm{nom}}^{(s)} \\ &= \beta_h(\boldsymbol{m}_{t-1,\mathrm{obs}}^{(s)} - \boldsymbol{m}_{t-1,\mathrm{nom}}^{(s)}) + (1-\beta_h)\big(\hat{\boldsymbol{h}}_t^{(s)} - \bar{\boldsymbol{h}}_t^{(s)}\big) \\ &= \beta_h \Delta\boldsymbol{m}_{t-1}^{(s)} + (1-\beta_h)\frac{1}{d}\sum_{r\in B_s}\boldsymbol{\delta}_t^{(s,r)}.\end{aligned} \tag{16}$$

假设 $\Delta\boldsymbol{m}_0^{(s)} = \mathbf{0}$，我们可以求解这个递推关系：

$$\Delta\boldsymbol{m}_t^{(s)} = (1-\beta_h)\sum_{j=1}^{t}\beta_h^{t-j}\frac{1}{d}\sum_{r\in B_s}\boldsymbol{\delta}_j^{(s,r)}. \tag{17}$$

取范数并应用三角不等式，可得：

$$\begin{aligned}\|\Delta\boldsymbol{m}_t^{(s)}\| &\le (1-\beta_h)\sum_{j=1}^{t}\beta_h^{t-j}\frac{1}{d}\sum_{r\in B_s}\|\boldsymbol{\delta}_j^{(s,r)}\| \\ &\le (1-\beta_h)\sum_{j=1}^{t}\beta_h^{t-j}\frac{|B_s|}{d}\varepsilon \\ &= (1-\beta_h)\gamma_s\varepsilon\sum_{j=1}^{t}\beta_h^{t-j} \\ &= (1-\beta_h)\gamma_s\varepsilon\frac{1-\beta_h^t}{1-\beta_h} \\ &\le (1-\beta_h)\gamma_s\varepsilon\frac{1}{1-\beta_h} \\ &= \gamma_s\varepsilon.\end{aligned} \tag{18}$$

这就证明了所述界限。对于只考虑最近一次迭代影响的情况（等价于初始化 $\boldsymbol{m}_{t-1,\mathrm{obs}}^{(s)} = \boldsymbol{m}_{t-1,\mathrm{nom}}^{(s)}$），我们有：

$$\begin{aligned}\|\Delta\boldsymbol{m}_t^{(s)}\| &= \left\|(1-\beta)\frac{1}{d}\sum_{r\in B_s}\boldsymbol{\delta}_t^{(s,r)}\right\| \\ &\le (1-\beta)\frac{|B_s|}{d}\varepsilon \\ &= (1-\beta)\gamma_s\varepsilon.\end{aligned} \tag{19}$$

□

定理 E.1确立了基于动量的平滑的一项关键性质：它能够自然地减弱恶意扰动的影响。这种衰减与恶意工作节点 γ_s所占比例成正比，表明恶意联盟规模越小，对全局状态的影响就越小。这一结果对于理解系统遏制恶意影响的能力至关重要。

基于这一　　基础，我们现在分析这些有界扰动如何影响我们的检测统计量　　：

引理 E.2（检验统计量偏差）。 对于度量函数 Ω，假设检测器计算

$$\Gamma_t^{(s,r)} = \left\|\Omega\big(\boldsymbol{h}_t^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}\big) - \Omega_{\text{ref}}^{(s)}\right\| \tag{20}$$

其中， $\Omega^{(s)}$ 参考 是由受信任的训练节点计算得出的参考统计量（例如，我们基于中位数的参考统计量）。
假设对于每个工作节点副本 $r \in \{1, 2, \ldots, d\}$:

- *Activation perturbation:* $\|\boldsymbol{\delta}_t^{(s,r)}\| \le \varepsilon$.
- *Momentum update:* $\boldsymbol{m}_t^{(s)} = \beta_h \boldsymbol{m}_{t-1}^{(s)} + (1-\beta_h)\frac{1}{d}\sum_{r=1}^{d} \boldsymbol{h}_t^{(s,r)}$ *with at most a fraction* γ_s *malicious workers (from Theorem E.1)*
$$\|\Delta \boldsymbol{m}_t^{(s)}\| \le \gamma_s \varepsilon. \tag{21}$$
- Ω的利普希茨连续性：对于任意输入 $\boldsymbol{x}$、 $\boldsymbol{y}$ 及扰动 $\boldsymbol{\delta}_x$、 $\boldsymbol{\delta}_y$:
$$\left\|\Omega(\boldsymbol{x}+\boldsymbol{\delta}_x, \boldsymbol{y}+\boldsymbol{\delta}_y) - \Omega(\boldsymbol{x}, \boldsymbol{y})\right\| \le L_\Omega \left(\|\boldsymbol{\delta}_x\| + \|\boldsymbol{\delta}_y\|\right), \tag{22}$$
其中， L_Ω 是 Ω的利普希茨常数。

定义（可能已知的）基线差距如下:

$$\delta^{\text{base}} = \Omega\big(\boldsymbol{h}_{t,\text{honest}}^{(s,r)}, \boldsymbol{m}_{t-1,\text{honest}}^{(s)}\big). \tag{23}$$

于是，检验统计量满足:

$$\boxed{\Gamma_t^{(s,r)} \le \delta_{\text{avg}}^{\text{base}} + L_\Omega \varepsilon + L_\Omega \gamma_s \varepsilon} \tag{24}$$

证明。 对于带有恶意扰动 $\boldsymbol{\delta}_t^{(s,r)}$ 的激活向量，利用 Ω的利普希茨性质，可得:

$$\begin{aligned}\left\|\Omega(\boldsymbol{h}_t^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}) - \Omega(\boldsymbol{h}_{t,\text{honest}}^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)})\right\| &\le L_\Omega \|\boldsymbol{\delta}_t^{(s,r)}\| \\ &\le L_\Omega \varepsilon.\end{aligned} \tag{25}$$

根据定理 E.1，我们知道动量向量偏差有界于 $\|\Delta \boldsymbol{m}_{t-1}^{(s)}\| \le \gamma_s \varepsilon$。因此，再次应用利普希茨性质:

$$\begin{aligned}\left\|\Omega(\boldsymbol{h}_{t,\text{honest}}^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}) - \Omega(\boldsymbol{h}_{t,\text{honest}}^{(s,r)}, \boldsymbol{m}_{t-1,\text{honest}}^{(s)})\right\| &\le L_\Omega \|\Delta \boldsymbol{m}_{t-1}^{(s)}\| \\ &\le L_\Omega \gamma_s \varepsilon.\end{aligned} \tag{26}$$

现在，我们可以利用三角不等式分解检验统计量：

$$\begin{aligned}
\Gamma_t^{(s,r)} &= \left\|\Omega(\boldsymbol{h}_t^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}) - \Omega_{\text{ref}}^{(s)}\right\| \\
&= \Big\|\Omega(\boldsymbol{h}_t^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}) - \Omega(\boldsymbol{h}_{t,\text{honest}}^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}) \\
&\quad + \Omega(\boldsymbol{h}_{t,\text{honest}}^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}) - \Omega(\boldsymbol{h}_{t,\text{honest}}^{(s,r)}, \boldsymbol{m}_{t-1,\text{honest}}^{(s)}) \\
&\quad + \Omega(\boldsymbol{h}_{t,\text{honest}}^{(s,r)}, \boldsymbol{m}_{t-1,\text{honest}}^{(s)}) - \Omega_{\text{ref}}^{(s)}\Big\| \\
&\le \left\|\Omega(\boldsymbol{h}_t^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}) - \Omega(\boldsymbol{h}_{t,\text{honest}}^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)})\right\| \\
&\quad + \left\|\Omega(\boldsymbol{h}_{t,\text{honest}}^{(s,r)}, \boldsymbol{m}_{t-1}^{(s)}) - \Omega(\boldsymbol{h}_{t,\text{honest}}^{(s,r)}, \boldsymbol{m}_{t-1,\text{honest}}^{(s)})\right\| \\
&\quad + \left\|\Omega(\boldsymbol{h}_{t,\text{honest}}^{(s,r)}, \boldsymbol{m}_{t-1,\text{honest}}^{(s)}) - \Omega_{\text{ref}}^{(s)}\right\| \\
&\le L_\Omega \varepsilon + L_\Omega \gamma_s \varepsilon + \delta^{\text{base}}
\end{aligned} \tag{27}$$

如果检测器补偿（或忽略）基线差距 δ^{base}，并在 $\Gamma_t^{(s,r)} > \tau$时发出警报，则仅可归因于恶意扰动的额外偏差为：

$$\Gamma_{\text{pert}} \le L_\Omega \varepsilon + L_\Omega \gamma_s \varepsilon = L_\Omega (1 + \gamma_s)\varepsilon, \tag{28}$$

因此，只要满足以下条件，恶意工作节点就可以保持不被检测到：

$$L_\Omega (1 + \gamma_s)\varepsilon \le \tau, \quad \Longrightarrow \quad \boxed{\varepsilon \le \frac{\tau}{L_\Omega (1 + \gamma_s)}} \tag{29}$$

至此，证明完毕。 □

定理 E.2 给出了恶意扰动下检验统计量偏差的关键界。该界取决于两个关键因素：（1）利普希茨常数 L_Ω（检验函数的）以及（2）恶意工作节点所占比例 γ_s。其实际含义是，恶意工作节点只有在其扰动幅度满足以下条件时才能保持不被检测到：

$$\varepsilon \le \frac{\tau}{L_\Omega (1 + \gamma_s)} \tag{30}$$

这建立了检测阈值 τ 与最大不可检测扰动幅度之间的直接关系。该方程表明调节 τ 如何影响安全-性能权衡：阈值越低，我们就能提供更强的安全保证，但可能以假阳性增加为代价。这凸显了为检验统计量设置适当阈值的重要性。

E.1.2 梯度扰动分析

在确定了不可检测激活扰动的界限后，我们分析这些扰动如何在网络中传播并影响参数梯度。这项分析对于理解其对训练动态的影响至关重要。

引理 E.3 (逐阶段 Lipschitz 常数)。 假设阶段 s 的副本 r 实现了一个映射

$$\boldsymbol{h}^{(s,r)} = f_s\big(\boldsymbol{h}^{(s-1,r)}; \boldsymbol{\theta}^{(s)}\big)$$

其雅可比矩阵满足

$$\|\partial_{\boldsymbol{\theta}} f_s\| \le L_{\boldsymbol{\theta}}^{(s)}, \qquad \|\partial_{\boldsymbol{h}} f_s\| \le L_f^{(s)}.$$

那么，阶段 s的参数梯度服从

$$\|\nabla^{agg}_{\boldsymbol{\theta}^{(s)}}\mathcal{L}(\boldsymbol{\theta})\| \le L^{(s)}_{\boldsymbol{\theta}}\ \frac{1}{d}\sum_{r=1}^{d}\|\boldsymbol{g}^{(s,r)}\|,$$

其中， $\boldsymbol{g}^{(s+1,r)}$ 是相对于激活 $\boldsymbol{h}^{(s,r)}$的梯度。

证明。 损失 $\mathcal{L}(\boldsymbol{\theta})$仅通过阶段映射的复合依赖于 $\boldsymbol{\theta}^{(s)}$ ：

$$\begin{aligned}\boldsymbol{h}^{(s,r)} &= f_s(\boldsymbol{h}^{(s-1,r)};\boldsymbol{\theta}^{(s)})\\ \boldsymbol{h}^{(s+1,r)} &= f_{s+1}(\boldsymbol{h}^{(s,r)};\boldsymbol{\theta}^{(s+1)})\\ &\vdots\\ \boldsymbol{h}^{(p,r)} &= f_p(\boldsymbol{h}^{(p-1,r)};\boldsymbol{\theta}^{(p)})\end{aligned} \tag{31}$$

随后是读出 $\mathcal{L}_{\text{head}}(\boldsymbol{h}^{(p,r)})$。
应用链式法则可得

$$\nabla_{\boldsymbol{\theta}^{(s,r)}}\mathcal{L}(\boldsymbol{\theta}) = \partial_{\boldsymbol{\theta}} f_s\big(\partial_{\boldsymbol{h}} f_{s+1}\big)\cdots\big(\partial_{\boldsymbol{h}} f_p\big)\nabla_{\boldsymbol{h}^{(p,r)}}\mathcal{L}_{\text{head}}. \tag{32}$$

这里，每个 $\partial_{\boldsymbol{\theta}} f_s$ 均在$(\boldsymbol{h}^{(s-1,r)},\boldsymbol{\theta}^{(s)})$处求值，每个 $\partial_{\boldsymbol{h}} f_j$ 均在$(\boldsymbol{h}^{(j-1,r)},\boldsymbol{\theta}^{(j)})$处求值。定义

$$\boldsymbol{g}^{(s+1,r)} := \big(\partial_{\boldsymbol{h}} f_{s+1}\big)\cdots\big(\partial_{\boldsymbol{h}} f_p\big)\ \nabla_{\boldsymbol{h}^{(p,r)}}\mathcal{L}_{\text{head}}, \tag{33}$$

从而 $\boldsymbol{g}^{(s+1,r)}$ 恰好是反向传播期间进入阶段 s 的副本 r 的激活梯度。全归约操作在应用这些梯度之前，会聚合阶段 s 的 d 个副本的所有梯度，即:

$$\nabla^{\text{agg}}_{\boldsymbol{\theta}^{(s)}}\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{d}\sum_{r=1}^{d}\nabla_{\boldsymbol{\theta}^{(s,r)}}\mathcal{L}(\boldsymbol{\theta}) \tag{34}$$

取欧几里得范数，应用三角不等式，并利用算子范数的次可乘性，可得

$$\begin{aligned}\|\nabla^{\text{agg}}_{\boldsymbol{\theta}^{(s)}}\mathcal{L}(\boldsymbol{\theta})\| &\le \frac{1}{d}\sum_{r=1}^{d}\|\nabla_{\boldsymbol{\theta}^{(s,r)}}\mathcal{L}(\boldsymbol{\theta})\|\\ &\le \frac{1}{d}\sum_{r=1}^{d}\|\partial_{\boldsymbol{\theta}} f_s\|\ \|\boldsymbol{g}^{(s+1,r)}\|\\ &\le L^{(s)}_{\boldsymbol{\theta}}\ \frac{1}{d}\sum_{r=1}^{d}\|\boldsymbol{g}^{(s+1,r)}\|.\end{aligned} \tag{35}$$

注意，式（34）中的步骤遵循如下事实：利普希茨常数假设在数据分布上是一致的。

□

定理 E.3 刻画了各阶段的参数梯度对激活扰动的依赖程度。利普希茨常数 $L^{(s)}_{\boldsymbol{\theta}}$ 和 $L^{(s)}_{f}$ 量化了这种关系，为理解梯度敏感性奠定了基础。这些特定于阶段的利普希茨常数很重要，因为它们揭示了模型中哪些阶段最容易受到恶意操纵的影响。常数较大的阶段会更强烈地放大扰动，因此会成为攻击者的首要目标，也是需要优先加强监控的环节。

基于这些利普希茨性质，我们现在精确量化激活扰动如何转化为梯度扰动：

引理E.4（参数梯度对激活扰动的敏感性）。 设阶段 s 中的一个预期诚实副本的激活为 $\boldsymbol{h}^{(s,r)}$。假设恶意工作节点将其替换为 $\boldsymbol{h}^{(s,r)}+\boldsymbol{\delta}$。若激活扰动足够小，使输入激活的变化 $\boldsymbol{\delta}$ 仅通过局部雅可比矩阵扰动 $\boldsymbol{g}^{(s+1,r)}$，则聚合后的 阶段 s参数梯度的变化满足

$$\|\Delta\nabla^{agg}_{\boldsymbol{\theta}^{(s)}}\mathcal{L}(\boldsymbol{\theta})\| := \|\nabla^{agg}_{\boldsymbol{\theta}^{(s)}}\mathcal{L}\big(\boldsymbol{\theta}\mid \boldsymbol{h}^{(s,r)}+\boldsymbol{\delta}\big) - \nabla^{agg}_{\boldsymbol{\theta}^{(s)}}\mathcal{L}\big(\boldsymbol{\theta}\mid \boldsymbol{h}^{(s,r)}\big)\| \le \frac{G_s}{d}\,\|\boldsymbol{\delta}\|,$$

其中 $G_s := L^{(s)}_{\boldsymbol{\theta}}\Big(\prod_{j\ge s+1} L^{(j)}_f\Big)$。更一般地，若一组 B_s 恶意副本各自注入范数至多为 $\|\boldsymbol{\delta}\|$的扰动，则

$$\boxed{\|\Delta\nabla^{agg}_{\boldsymbol{\theta}^{(s)}}\| \le \gamma_s\cdot G_s\cdot\varepsilon.} \tag{36}$$

证明。 我们先考虑单个副本 r 位于阶段 s。对于该副本，根据定理 E.3 可以写为：

$$\nabla_{\boldsymbol{\theta}^{(s,r)}}\mathcal{L}(\boldsymbol{\theta}) = \partial_{\boldsymbol{\theta}} f_s\,\boldsymbol{g}^{(s+1,r)},\quad \text{s.t. }\ \boldsymbol{g}^{(s+1,r)} = \Big(\prod_{j=s+1}^{p}\partial_{\boldsymbol{h}} f_j\Big)\nabla_{\boldsymbol{h}^{(p,r)}}\mathcal{L}_{\text{head}}.$$

因此，假设 在小输入扰动下对梯度信号的变化进行线性化，可得 ：

$$\delta\boldsymbol{g}^{(s+1,r)} = \Big(\prod_{j=s+1}^{p}\partial_{\boldsymbol{h}} f_j\Big)\boldsymbol{\delta}.$$

因此，对于副本 r，我们可以将参数梯度的敏感性写为：

$$\Delta\nabla_{\boldsymbol{\theta}^{(s,r)}}\mathcal{L}(\boldsymbol{\theta}) = (\partial_{\boldsymbol{\theta}} f_s)\,\delta\boldsymbol{g}^{(s+1,r)} = \partial_{\boldsymbol{\theta}} f_s\Big(\prod_{j=s+1}^{p}\partial_{\boldsymbol{h}} f_j\Big)\,\boldsymbol{\delta}.$$

利用次可乘性和利普希茨界，我们有

$$\begin{aligned}\|\Delta\nabla_{\boldsymbol{\theta}^{(s)}}\mathcal{L}(\boldsymbol{\theta})\| &\le \|\partial_{\boldsymbol{\theta}} f_s\|\Big(\prod_{j=s+1}^{p}\|\partial_{\boldsymbol{h}} f_j\|\Big)\|\boldsymbol{\delta}\| \\ &\le L^{(s)}_{\boldsymbol{\theta}}\Big(\prod_{j\ge s+1} L^{(j)}_f\Big)\|\boldsymbol{\delta}\| \\ &= G_s\|\boldsymbol{\delta}\|\end{aligned} \tag{37}$$

由于阶段更新使用聚合梯度，我们可以写成

$$\Delta\nabla^{\text{agg}}_{\boldsymbol{\theta}^{(s)}}\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{d}\,\Delta\nabla_{\boldsymbol{\theta}^{(s,r)}}\mathcal{L}(\boldsymbol{\theta}).$$

因此， $\|\Delta\nabla^{\text{agg}}_{\boldsymbol{\theta}^{(s)}}\mathcal{L}(\boldsymbol{\theta})\| \le \frac{G_s}{d}\|\boldsymbol{\delta}\|$. 如果有 $|B_s|$ 个副本被破坏，我们将有

$$\begin{aligned}\|\Delta\nabla^{\text{agg}}_{\boldsymbol{\theta}^{(s)}}\mathcal{L}(\boldsymbol{\theta})\| &\le \frac{1}{d}\sum_{r\in B_s}\|\Delta\nabla_{\boldsymbol{\theta}^{(s,r)}}\mathcal{L}(\boldsymbol{\theta})\| \\ &\le \frac{|B_s|}{d}\cdot G_s\cdot\|\boldsymbol{\delta}\| \\ &= \gamma_s\cdot G_s\cdot\|\boldsymbol{\delta}\|,\end{aligned} \tag{38}$$

证明至此完成。 □

定理E.4 提供了激活扰动及其对参数梯度的影响之间的关键联系。放大因子 G_s 表示阶段 s 的扰动如何在反向传播过程中通过网络传播。该因子取决于局部参数梯度敏感度（$L_{\boldsymbol{\theta}}^{(s)}$）以及后续阶段的激活梯度敏感度乘积（$\prod_{j\geq s+1} L_f^{(j)}$）。这一结果对于流水线并行训练中抵御恶意工作节点具有重要意义：

1. 较早的阶段（较低的 s）通常具有更大的放大因子，因为扰动必须经过更多后续阶段传播。2. 参数量较大或激活模式复杂的阶段，可能具有更高的单独 Lipschitz 常数。3. 恶意工作节点造成的影响比例会因全归约操作的平均效应而降低，这由 γ_s 因子体现。

结合我们的检测界，现在可以确定未被检测到的恶意工作节点所能引起的最大参数梯度扰动：

$$\|\Delta\nabla^{\text{agg}}_{\boldsymbol{\theta}^{(s)}}\mathcal{L}(\boldsymbol{\theta})\| \leq \gamma_s \cdot G_s \cdot \frac{\tau}{L_\Omega(1+\gamma_s)} := \zeta \tag{39}$$

该界将检测阈值与梯度扰动直接联系起来，这对于我们的收敛性分析至关重要。

E.1.3 扰动梯度下的收敛

在确定梯度扰动的界之后，我们现在分析这些扰动如何影响动量随机梯度下降的收敛性质。我们考虑一般的非凸损失函数，但我们的结果也可以轻松扩展到强凸情形。

定理 E.5（在扰动和噪声作用下，凸与非凸情形中满足光滑性条件的动量随机梯度下降的收敛性）**.** 考虑以下平衡动量更新：

$$\begin{aligned}\boldsymbol{v}_{t+1} &= \beta\boldsymbol{v}_t + (1-\beta)\boldsymbol{g}_t,\\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \eta\boldsymbol{v}_{t+1}\end{aligned} \tag{40}$$

其中 $\boldsymbol{g}_t = \nabla\mathcal{L}(\boldsymbol{\theta}_t) + \boldsymbol{\zeta}_t + \boldsymbol{\xi}_t$，其李雅普诺夫函数为 $\Psi_t = \mathcal{L}(\boldsymbol{\theta}_t) + c\|\boldsymbol{v}_t\|^2$，其中某个常数为 $c > 0$。假设：

1. $\mathcal{L}$是 L-光滑的，但可能非凸 *2.* $\mathcal{L}$有下界 $\mathcal{L}^*$*3.* $\beta \in [0,1)$ 是动量参数 *4.* $\eta > 0$ 是学习率 *5.* $\boldsymbol{\zeta}_t$ 是确定性扰动，最大扰动范数为 $\|\boldsymbol{\zeta}_t\| \leq \zeta$，而 $\boldsymbol{\xi}_t$ 是零均值噪声，其满足 $\mathbb{E}[\|\boldsymbol{\xi}_t\|^2] \leq \sigma^2$

对于任意正常数 ε_1、ε_2、ε_3，在给定过去信息 $\mathcal{F}_t$ 的条件下，我们有：

$$\mathbb{E}[\Psi_{t+1}|\mathcal{F}_t] \leq \Psi_t - \alpha\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2 + C_1\|\boldsymbol{v}_t\|^2 + C_2\|\boldsymbol{\zeta}_t\|^2 + D\sigma^2 \tag{41}$$

其中，这些常数定义如下：

$$\begin{aligned}
\alpha &= \eta(1-\beta)\left(1-\varepsilon_2-\frac{\beta}{4\varepsilon_1(1-\beta)}-\frac{2}{\eta}\left(\frac{\eta^2 L}{2}+c\right)\left(1-\beta+\frac{\beta}{4\varepsilon_1}\right)\right)\\
C_1 &= \left(\eta\beta\varepsilon_1+\left(\frac{\eta^2 L}{2}+c\right)\beta\left(\beta+2(1-\beta)(\varepsilon_1+\varepsilon_3)\right)-c\right)\\
C_2 &= \left(\frac{\eta(1-\beta)}{4\varepsilon_2}+2\left(\frac{\eta^2 L}{2}+c\right)(1-\beta)\left(1-\beta-\frac{\beta}{4\varepsilon_3}\right)\right)\\
D &= \left(\frac{\eta^2 L}{2}+c\right)(1-\beta)^2.
\end{aligned} \tag{42}$$

如果我们为 ε_1、ε_2、ε_3 选择合适的值，使得 $\alpha>0$ 和 $C_1<0$ 成立，并假设 $\boldsymbol{v}_0=\mathbf{0}$，那么该算法在期望意义下收敛到一个驻点的邻域：

$$\boxed{\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2]\le\frac{\mathcal{L}_0-\mathcal{L}^*}{\alpha T}+\frac{C_2\zeta^2+D\sigma^2}{\alpha},} \tag{43}$$

其中，$\mathcal{L}_0:=\mathcal{L}(\boldsymbol{\theta}_0)$ 是我们在初始化时的损失值。

证明。 我们首先使用受（Liu 等人，2020；Mai 和 Johansson，2020）启发的李雅普诺夫势函数 $\Psi_t=\mathcal{L}(\boldsymbol{\theta}_t)+c\|\boldsymbol{v}_t\|^2$分析单步进展。

损失项的演化。 根据 $\mathcal{L}$ 的 L 光滑性，我们有：

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}_{t+1}) &\le \mathcal{L}(\boldsymbol{\theta}_t)+\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\boldsymbol{\theta}_{t+1}-\boldsymbol{\theta}_t\rangle+\frac{L}{2}\|\boldsymbol{\theta}_{t+1}-\boldsymbol{\theta}_t\|^2\\
&= \mathcal{L}(\boldsymbol{\theta}_t)-\eta\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\boldsymbol{v}_{t+1}\rangle+\frac{\eta^2 L}{2}\|\boldsymbol{v}_{t+1}\|^2
\end{aligned} \tag{44}$$

现在，展开 $\Psi_{t+1}-\Psi_t=\left[\mathcal{L}(\boldsymbol{\theta}_{t+1})-\mathcal{L}(\boldsymbol{\theta}_t)\right]+c\left[\|\boldsymbol{v}_{t+1}\|^2-\|\boldsymbol{v}_t\|^2\right]$，我们有：

$$\begin{aligned}
\Psi_{t+1}-\Psi_t &= \left[\mathcal{L}(\boldsymbol{\theta}_{t+1})-\mathcal{L}(\boldsymbol{\theta}_t)\right]+c\left[\|\boldsymbol{v}_{t+1}\|^2-\|\boldsymbol{v}_t\|^2\right]\\
&= -\eta\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\boldsymbol{v}_{t+1}\rangle+\frac{\eta^2 L}{2}\|\boldsymbol{v}_{t+1}\|^2+c\|\boldsymbol{v}_{t+1}\|^2-c\|\boldsymbol{v}_t\|^2\\
&= -\eta\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\boldsymbol{v}_{t+1}\rangle+\left(\frac{\eta^2 L}{2}+c\right)\|\boldsymbol{v}_{t+1}\|^2-c\|\boldsymbol{v}_t\|^2.
\end{aligned} \tag{45}$$

代入 $\boldsymbol{v}_{t+1}=\beta\boldsymbol{v}_t+(1-\beta)\boldsymbol{g}_t$ (Polyak，1964年)，于是我们得到：

$$\begin{aligned}
\Psi_{t+1}-\Psi_t &\le -\eta\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\boldsymbol{v}_{t+1}\rangle+\left(\frac{\eta^2 L}{2}+c\right)\|\boldsymbol{v}_{t+1}\|^2-c\|\boldsymbol{v}_t\|^2\\
&= -\eta\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\beta\boldsymbol{v}_t+(1-\beta)\boldsymbol{g}_t\rangle+\left(\frac{\eta^2 L}{2}+c\right)\|\beta\boldsymbol{v}_t+(1-\beta)\boldsymbol{g}_t\|^2-c\|\boldsymbol{v}_t\|^2\\
&= -\eta\beta\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\boldsymbol{v}_t\rangle-\eta(1-\beta)\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\boldsymbol{g}_t\rangle\\
&\quad+\left(\frac{\eta^2 L}{2}+c\right)\left(\beta^2\|\boldsymbol{v}_t\|^2+(1-\beta)^2\|\boldsymbol{g}_t\|^2+2\beta(1-\beta)\langle\boldsymbol{v}_t,\boldsymbol{g}_t\rangle\right)-c\|\boldsymbol{v}_t\|^2
\end{aligned} \tag{46}$$

接下来，我们对各项分别进行界定。

界定 $-\eta\beta\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t)\ \boldsymbol{v}_t\rangle$。 利用参数为 $\varepsilon_1>0$ 的杨氏不等式，我们得到：

$$-\eta\beta\langle\boldsymbol{v}_t,\nabla\mathcal{L}(\boldsymbol{\theta}_t)\rangle\le\eta\beta\varepsilon_1\|\boldsymbol{v}_t\|^2+\frac{\eta\beta}{4\varepsilon_1}\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2. \tag{47}$$

界定 $-\eta(1-\beta)\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t)\ \boldsymbol{g}_t\rangle$。 由于 $\boldsymbol{g}_t=\nabla\mathcal{L}(\boldsymbol{\theta}_t)+\boldsymbol{\zeta}_t+\boldsymbol{\xi}_t$，我们可以写为：

$$\begin{aligned}-\eta(1-\beta)\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\boldsymbol{g}_t\rangle&=-\eta(1-\beta)\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\nabla\mathcal{L}(\boldsymbol{\theta}_t)+\boldsymbol{\zeta}_t+\boldsymbol{\xi}_t\rangle\\&\le-\eta(1-\beta)\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2-\eta(1-\beta)\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\boldsymbol{\zeta}_t\rangle\\&\quad-\eta(1-\beta)\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\boldsymbol{\xi}_t\rangle\\&\le-\eta(1-\beta)\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2+\eta(1-\beta)\varepsilon_2\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2+\frac{\eta(1-\beta)}{4\varepsilon_2}\|\boldsymbol{\zeta}_t\|^2\\&\quad-\eta(1-\beta)\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t),\boldsymbol{\xi}_t\rangle\end{aligned} \tag{48}$$

界定 $\|\boldsymbol{g}_t\|^2$。 对于这一项，我们写为：

$$\begin{aligned}\|\boldsymbol{g}_t\|^2&=\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)+\boldsymbol{\zeta}_t+\boldsymbol{\xi}_t\|^2\\&=\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)+\boldsymbol{\zeta}_t\|^2+\|\boldsymbol{\xi}_t\|^2+2\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t)+\boldsymbol{\zeta}_t,\boldsymbol{\xi}_t\rangle\\&\le2\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2+2\|\boldsymbol{\zeta}_t\|^2+\|\boldsymbol{\xi}_t\|^2+2\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t)+\boldsymbol{\zeta}_t,\boldsymbol{\xi}_t\rangle,\end{aligned} \tag{49}$$

界定 $\langle\boldsymbol{v}_t,\ \boldsymbol{g}_t\rangle$。 展开 $\boldsymbol{g}_t=\nabla\mathcal{L}(\boldsymbol{\theta}_t)+\boldsymbol{\zeta}_t+\boldsymbol{\xi}_t$，可得：

$$\begin{aligned}\langle\boldsymbol{v}_t,\boldsymbol{g}_t\rangle&=\langle\boldsymbol{v}_t,\nabla\mathcal{L}(\boldsymbol{\theta}_t)+\boldsymbol{\zeta}_t+\boldsymbol{\xi}_t\rangle\\&=\langle\boldsymbol{v}_t,\nabla\mathcal{L}(\boldsymbol{\theta}_t)\rangle+\langle\boldsymbol{v}_t,\boldsymbol{\zeta}_t\rangle+\langle\boldsymbol{v}_t,\boldsymbol{\xi}_t\rangle\\&\le\varepsilon_1\|\boldsymbol{v}_t\|^2+\frac{1}{4\varepsilon_1}\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2+\varepsilon_3\|\boldsymbol{v}_t\|^2+\frac{1}{4\varepsilon_3}\|\boldsymbol{\zeta}_t\|^2+\langle\boldsymbol{v}_t,\boldsymbol{\xi}_t\rangle\end{aligned} \tag{50}$$

合并各项。 对式（46）取条件期望，并代入前面的界，我们得到

$$\begin{aligned}\mathbb{E}[\Psi_{t+1}|\mathcal{F}_t]-\Psi_t\le{}&\eta\beta\varepsilon_1\|\boldsymbol{v}_t\|^2+\frac{\eta\beta}{4\varepsilon_1}\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2\\&-\eta(1-\beta)\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2+\eta(1-\beta)\varepsilon_2\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2+\frac{\eta(1-\beta)}{4\varepsilon_2}\|\boldsymbol{\zeta}_t\|^2\\&+\left(\frac{\eta^2L}{2}+c\right)\left(\beta^2\|\boldsymbol{v}_t\|^2+2(1-\beta)^2\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2+2(1-\beta)^2\|\boldsymbol{\zeta}_t\|^2+(1-\beta)^2\sigma^2\right)\\&2\left(\frac{\eta^2L}{2}+c\right)\beta(1-\beta)\left((\varepsilon_1+\varepsilon_3)\|\boldsymbol{v}_t\|^2+\frac{1}{4\varepsilon_1}\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2+\frac{1}{4\varepsilon_3}\|\boldsymbol{\zeta}_t\|^2\right)-c\|\boldsymbol{v}_t\|^2\end{aligned} \tag{51}$$

合并同类项并设定界限。 代入所有界限并合并同类项后，我们得到：

$$\begin{aligned}\mathbb{E}[\Psi_{t+1}|\mathcal{F}_t]-\Psi_t\le{}&\\&-\eta(1-\beta)\left(1-\varepsilon_2-\frac{\eta\beta}{4\varepsilon_1\eta(1-\beta)}-\frac{2}{\eta}\left(\frac{\eta^2L}{2}+c\right)\left(1-\beta+\frac{\beta}{4\varepsilon_1}\right)\right)\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2\\&+\left(\eta\beta\varepsilon_1+\left(\frac{\eta^2L}{2}+c\right)\beta\left(\beta+2(1-\beta)(\varepsilon_1+\varepsilon_3)\right)-c\right)\|\boldsymbol{v}_t\|^2\\&+\left(\frac{\eta(1-\beta)}{4\varepsilon_2}+2\left(\frac{\eta^2L}{2}+c\right)(1-\beta)\left(1-\beta-\frac{\beta}{4\varepsilon_3}\right)\right)\|\boldsymbol{\zeta}_t\|^2\\&+\left(\frac{\eta^2L}{2}+c\right)(1-\beta)^2\sigma^2\end{aligned} \tag{52}$$

定义以下常数：

$$\begin{aligned}
\alpha &= \eta(1-\beta)\left(1-\varepsilon_2-\frac{\beta}{4\varepsilon_1(1-\beta)}-\frac{2}{\eta}\left(\frac{\eta^2 L}{2}+c\right)\left(1-\beta+\frac{\beta}{4\varepsilon_1}\right)\right)\\
C_1 &= \left(\eta\beta\varepsilon_1+\left(\frac{\eta^2 L}{2}+c\right)\beta\left(\beta+2(1-\beta)(\varepsilon_1+\varepsilon_3)\right)-c\right)\\
C_2 &= \left(\frac{\eta(1-\beta)}{4\varepsilon_2}+2\left(\frac{\eta^2 L}{2}+c\right)(1-\beta)\left(1-\beta-\frac{\beta}{4\varepsilon_3}\right)\right)\\
D &= \left(\frac{\eta^2 L}{2}+c\right)(1-\beta)^2
\end{aligned} \tag{53}$$

建立收敛性。对于收敛性，我们可以设置变量，使得 $\alpha > 0$和 $C_1 < 0$。期望意义下的单步进展变为：

$$\mathbb{E}[\Psi_{t+1}|\mathcal{F}_t] \leq \Psi_t - \alpha\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2 + C_1\|\boldsymbol{v}_t\|^2 + C_2\|\boldsymbol{\zeta}_t\|^2 + D\sigma^2 \tag{54}$$

当 $C_1 < 0$时，含有 $\|\boldsymbol{v}_t\|^2$ 的项有助于收敛。取完整期望并从 $t = 0$求和至 $T-1$：

$$\begin{aligned}
\sum_{t=0}^{T-1}\mathbb{E}[\alpha\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2] &\leq \mathbb{E}[\Psi_0] - \mathbb{E}[\Psi_T] + \sum_{t=0}^{T-1}(C_2\|\boldsymbol{\zeta}_t\|^2 + D\sigma^2])\\
&\leq \mathbb{E}[\Psi_0] - \mathcal{L}^* + T(C_2\zeta^2 + D\sigma^2)
\end{aligned} \tag{55}$$

其中我们使用了 $\mathcal{L}^* \leq \mathcal{L}(\boldsymbol{\theta}_t)$，并舍去了含有 $C_1 < 0$的负项。代入 $\Psi_0 = \mathcal{L}_0 + c\|v_0\|^2 = \mathcal{L}_0$并除以 αT，得到：

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2] \leq \frac{\mathcal{L}_0 - \mathcal{L}^*}{\alpha T} + \frac{C_2\zeta^2 + D\sigma^2}{\alpha}. \tag{56}$$

因此，平均平方梯度范数收敛到一个邻域，该邻域由扰动幅度 ζ^2 和噪声方差 σ^2决定。这一结果表明，在非凸光滑情形下，带有扰动和噪声的动量随机梯度下降会收敛到驻点的一个邻域。

□

定理2为非凸设定下的动量随机梯度下降提供了收敛保证，这对于大语言模型等深度学习应用尤其重要。我们不再保证收敛到最优点的一个邻域，而是对平均梯度范数提供保证；这是衡量非凸优化的标准指标。该界直接取决于扰动幅度，这表明即使在非凸设定下，通过有效的检测机制控制恶意扰动，对于确保收敛到驻点也至关重要。

E.1.4 统一分析：检测-收敛关系

现在，我们将结果统一起来，建立检测阈值与收敛保证之间的全面关系。这一统一视角为我们提供了关于安全-性能权衡的明确指导。

定理 E.6（恶意工作节点下分布式训练的收敛保证）。设一个分布式训练系统包含 p 个阶段，每个阶段有 d 个工作节点副本，其中阶段 s 有 $\gamma_s < \frac{1}{2}$ 比例的工作节点是恶意的。使用阈值为 τ-的检验统计量检测器进行基于动量的验证，并假设我们将使用参数为 $\beta \in [0,1$ 的平衡动量*SGD*），学习率为 $\eta > 0$，用于

优化。
假设：

1. 检验统计量函数 Ω是利普希茨连续的，其常数为 L_Ω；*2.* 每个阶段 s 实现一个映射 $\boldsymbol{h}^{(s,r)} = f_s\left(\boldsymbol{h}^{(s-1,r)};\ \boldsymbol{\theta}^{(s)}\right)$，其雅可比矩阵界为 $\|\partial_{\boldsymbol{\theta}} f_s\| \le L_{\boldsymbol{\theta}}^{(s)}$，且 $\|\partial_{\boldsymbol{h}} f_s\| \le L_{(s)} f$；*3.* 损失函数 $\mathcal{L}$ 是 L-光滑的，且下界为 $\mathcal{L}^*$；*4.* 随机梯度包含零均值噪声，其方差上界为 σ^2。

于是，以下结果成立：

- **检测　　规避界**。 恶意工作节点要想不被检验统计量检测器发现：

$$\varepsilon \le \frac{\tau}{L_\Omega(1+\gamma_s)}. \tag{57}$$

- **参数梯度扰动**。 阶段 s 未被检测到的恶意工作节点所能造成的最大参数梯度扰动为：

$$\|\Delta\nabla_{\boldsymbol{\theta}^{(s)}}^{agg}\| \le \gamma_s \cdot G_s \cdot \frac{\tau}{L_\Omega(1+\gamma_s)} \tag{58}$$

其中，$G_s = L_{\boldsymbol{\theta}}^{(s)}\left(\prod_{j\ge s+1} L_f^{(j)}\right)$ 表示扰动的放大因子。

- **收敛界**。 在最大未检测扰动下，并假设ζ：$= \gamma_s \cdot G_s \cdot {}_{L_\Omega}{}^{\tau}{}_{(1+\gamma_s)}$，对于非凸损失，我们有：

$$\boxed{\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2] \le \frac{\mathcal{L}_0-\mathcal{L}^*}{\alpha T} + \frac{C_2\zeta^2 + D\sigma^2}{\alpha}} \tag{59}$$

其中常数的定义与定理*E.5*相同。

证明。W　我们通过关联定理 E.1 和 E.4 以及定理 E.5 来证明该定理　.　□

E.1.5 Recovering Well-known Lower-bounds for SGD Convergence from Theorem E.6

为评估我们的收敛定理是否有效，我们考察它能否推广到常见的非凸优化界。考虑不含恶意扰动的普通随机梯度下降：令 $\beta = 0$（将动量随机梯度下降简化为随机梯度下降），$\zeta = 0$（无恶意扰动），并将其代入式（42）中的系数，其中 $c \to 0^+$：

$$\begin{aligned} \alpha &\approx \mathcal{O}(\eta) \\ D &\approx \mathcal{O}(\eta^2 L), \end{aligned} \tag{60}$$

其中 η 是我们的学习率（见公式（40））。

将这些系数代入我们的收敛定理，可得：

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\mathcal{L}(\theta_t)\|^2] \leq \frac{\mathcal{L}_0-\mathcal{L}^*}{\alpha T}+\frac{C_2\zeta^2+D\sigma^2}{\alpha} \tag{61}$$

$$\overset{(a)}{\leq} \frac{\mathcal{L}_0-\mathcal{L}^*}{\alpha T}+\frac{D\sigma^2}{\alpha} \tag{62}$$

$$\overset{(b)}{\leq} \mathcal{O}\left(\frac{\mathcal{L}_0-\mathcal{L}^*}{\eta T}+\eta L\sigma^2\right) \tag{63}$$

其中，(a) 假设没有扰动，(b) 使用推导出的常数。

这与经典的随机梯度下降界一致，出自Koloskova 等人(2024年)：

> **“随机梯度下降，例 3.1** 由于 $\sigma^2 \leq \tau\sigma_{\text{SGD}}^2$ （见表 2），并利用 $\tau = \Theta(1/L\gamma)$，定理 5.1 中的收敛率可转化为
>
> $$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\nabla f(x_t)\|^2 \leq \mathcal{O}\left(\frac{F_0}{\gamma T}+L\gamma\sigma_{\text{SGD}}^2\right),$$
>
> 其中 $\gamma \leq \frac{1}{8\sqrt{3}L}$ [，且 $F_0 = f(\mathbf{x}_0) - f^*$]。这恢复了非凸函数上随机梯度下降的经典收敛率（忽略常数因子）。”

该界与我们根据收敛定理推导出的结果完全一致。

E.1.6 理论启示

我们的统一分析揭示了设计可容忍恶意行为的 SWARM 验证的若干重要启示：

- **安全性与收敛性权衡**：检测阈值 τ 通过其对最大未检测扰动的影响，直接影响收敛保证。较低的阈值可提供更强的安全保证，但可能增加假阳性，并因不必要的工作节点排除而可能减缓收敛。

- **拜占庭比例影响**：如式（58）所示，最大参数偏差与 $\frac{\gamma_s}{1+\gamma_s}$成正比，其中 γ_s 表示阶段 s的恶意工作节点比例。该单调递增函数表明，更大的恶意比例允许更严重的参数偏差，凸显了维持诚实多数的重要性。

- **检测器灵敏度**：检测器 Ω 的利普希茨常数 L_Ω 越大，参数梯度偏差就越小。从实际角度看，采用更敏感的检测函数可以压缩恶意工作节点进行未检测扰动的空间，从而限制其潜在影响。

- **阶段脆弱性**：放大因子 G_s 值越高的阶段越容易受到恶意扰动的影响。在典型的神经网络架构中，这通常意味着较早的层（会影响所有后续计算）具有更高的脆弱性。这表明，应优先配置安全资源，以便更密切地监控这些关键阶段。

- **作为抵御恶意行为的鲁棒性手段的动量**：较高的动量值 β 会通过提高历史梯度估计的权重，自然地减小逐迭代扰动的影响。这提供了一种内在的鲁棒性，可对显式检测机制形成补充。因此，最优动量值不仅取决于优化动态，还取决于安全性考量。

- **自适应检测阈值**：我们的分析表明，可根据各阶段的放大因子 G_s，为每个阶段分别设置最优的检测阈值。放大因子较高的阶段

应采用更严格的阈值，以确保模型在各阶段都具有一致的收敛保证。我们将此留待未来工作。

E.2 诚实多数保证的证明

本节给出第 3.2 节中定理 3.2的完整证明。

引理 1（诚实多数保证）。 考虑我们的分布式训练系统，其中有 p 个流水线阶段，每个阶段由 d 个工作节点构成副本。设 b 为恶意工作节点总数，$\epsilon \in$ （0，1）为一个较小的正常数。若工作节点通过随机方式分配到各阶段，且

$$\boxed{b \leq \frac{dp}{2} - p\sqrt{\frac{d}{2}\ln\left(\frac{p}{\epsilon}\right)},} \tag{64}$$

则以至少 $1-\epsilon$ 的概率，每个流水线阶段中的恶意工作节点都严格少于 $d/2$ 个。

换言之，定理 3.2 建立了恶意工作节点初始池与以下条件之间的联系：向各阶段进行“随机工作节点分配”时，能够满足 SENTINEL 所需的逐阶段诚实多数假设。这一点很重要，因为我们通常事先并不知道某个工作节点是否为拜占庭节点；而且，由于通常采用随机工作节点分配来分配工作节点，我们需要确保诚实多数假设得以保留。

证明。 令 B_s 表示阶段 s的恶意工作节点集合。我们对恶意节点的分配建模如下：每个工作节点以概率 $q = b/n = b/(dp)$ 成为恶意节点，且这一事件与其他节点相互独立。

对于任意阶段 s，当阶段被随机分配给工作节点时，恶意节点的数量 $|B_s|$ 服从参数为 d 和 q的二项分布：

$$|B_s| \sim \text{Binomial}(d, q), \quad \mathbb{E}[|B_s|] = qd. \tag{65}$$

我们的目标是确保，以高概率，每个阶段 s 都 $|B_s| < d/2$。根据 Hoeffding 不等式（Hoeffding，1994）对相互独立的伯努利随机变量之和的结论，有

$$\Pr\left[|B_s| - \mathbb{E}[|B_s|] \geq t\right] \leq \exp\left(-\frac{2t^2}{d}\right). \tag{66}$$

令 $t = d/2 - qd = d(1/2 - q)$，可得

$$\Pr\left[|B_s| \geq d/2\right] \leq \exp\left(-2d\left(\frac{1}{2} - q\right)^2\right). \tag{67}$$

对所有 p 个阶段应用并集界，至少一个阶段中恶意工作节点占多数的概率上界为

$$\Pr\left[\exists s : |B_s| \geq d/2\right] \leq p \cdot \exp\left(-2d\left(\frac{1}{2} - q\right)^2\right). \tag{68}$$

要使该概率至多为 ϵ，我们需要

$$p \cdot \exp\left(-2d\left(\frac{1}{2} - q\right)^2\right) \leq \epsilon. \tag{69}$$

对两边取自然对数并求解 q，得到

$$\frac{1}{2} - q \geq \sqrt{\frac{\ln(p/\epsilon)}{2d}}. \tag{70}$$

代入 $q = b/n = b/(dp)$并求解 b，我们得到允许的最大恶意 s
节点：

$$b_{\max} = \left(\frac{1}{2} - \sqrt{\frac{\ln(p/\epsilon)}{2d}}\right) dp = \frac{dp}{2} - p\sqrt{\frac{d}{2}\ln\left(\frac{p}{\epsilon}\right)}. \tag{71}$$

因此，如果恶意节点总数 b 至多为 $b_{\max}$，则以至少 $1-\epsilon$的概率，所有流水线阶段的恶意工作节点数都将严格少于 $d/2$ 。 □

这一理论结果表明，随着系统在每个阶段扩展更多副本，它对对抗性工作节点的鲁棒性会不断增强，并趋近于最多容忍一半工作节点为恶意的理论极限。

F. 扩展实验结果

本节提供支持主要发现的全面实验细节。我们首先介绍实验设置和超参数，随后给出扩展版的结果。

F.1 详细实验设置

实验基础设施。 为了模拟异构分布式环境，我们基于 `TorchTitan` (Liang 等人，2025年）框架开展实验，该框架构建于 `PyTorch` (Paszke 等人，2019）.[4]我们的设置采用流水线并行，其中每个 Transformer 层对应一个流水线阶段，数据并行副本则作为各阶段内的工作节点。我们使用 1 至 3 个计算节点，在三种规模下开展实验；每个节点配备 8 个 英伟达 A100-SXM4-40GB 图形处理器，总工作节点数分别为 64、128 和 256。消融研究使用 8 个 英伟达 A100-SXM4-80GB图形处理器进行。对于使用 Llama-3-4B 的大规模实验，我们使用了 8 个 英伟达 H100-HBM3-80GB图形处理器，并启用了激活检查点。

模型与训练配置。 我们使用不同的架构配置评估了仅解码器 Llama-3（Dubey 等，2024年）模型。完整的训练超参数见表 13。此外，我们还在 NanoGPT（Karpathy，2022）上评估了我们方法的性能，将其作为具有代表性的 GPT-2（Radford 等，2019）架构；同时还在 Llama-4-0.4B（MetaAI，2025年）和 DeepSeek-V3- 1B（DeepSeek-AI，2024年）上进行了评估，将它们作为具有代表性的混合专家模型。除非另有说明，我们对所有模型配置均训练 5000 步。

验证设置。 SENTINEL部署专用验证器节点，拦截阶段间通信，并监控前向激活和反向激活梯度是否存在异常行为。对于 $c = 5$ 次违规，我们将工作节点永久封禁。此外，我们实现了自适应四分位距阈值机制来检测异常值，其参数详见表 14。[5] 这些检测阈值通过在每种配置上开展初步实验进行经验校准，但进一步的超参数调优可能会带来更好的效果。

正如论文中所指出的，我们假设每个模型的第一层和最后两层均由诚实节点操作。这一假设至关重要，原因有二：

[4]我们按照 Karpathy (2022) 的实现，将 NanoGPT 集成到 `TorchTitan`中。 [5]对于基于 MoE 的模型实验，我们采用与 Llama-3-0.6B 模型相同的设置。

1. 这些层控制着前向传播和反向传播过程中的主要数据流与梯度流，对训练稳定性至关重要；2. 对于试图通过数据投毒/后门攻击（Li 等人，2024年）（通过输入层）或标签操纵（Biggio 等人，2012年；Fung 等人，2020）（通过输出层）破坏模型完整性的对手而言，这些层也是关键攻击面。

若没有这一诚实节点假设，恶意行为者便可通过直接篡改输入或输出，轻易绕过中间验证。

数据集。 我们在三个大规模文本语料库上进行实验：CommonCrawl（C4）（Raffel 等人，2020）、FineWeb（FW）（Penedo 等人，2024年）以及 OpenWebText（OW）（Gokaslan 等人，2019），均通过 Hugging Face Datasets 流式 API（Lhoest 等人，2021年）获取。为支持模型评估，我们构建了一个包含 10 万个样本的留出验证集，该验证集在整个过程中始终与训练数据分离。在验证阶段，我们从该验证集中为每个数据并行工作节点采样包含 160 个样本的批次用于评估。

Table 13: Model architectures and training configuration.

Model Configuration	NanoGPT-0.25B	Llama-3-0.6B	Llama-3-1.2B	Llama-3-4B	Llama-4-0.4B	DeepSeek-V3-1B
Model Architecture						
Parameters	278,364,672	574,391,296	1,224,247,296	3,984,989,184	443,957,760	967,989,760
Hidden Dimensions	768	1024	2048	3072	512	1024
Number of Layers (Stages)	12	16	8	22	16	15
Attention Heads	12	32	32	32	32	16
Key-Value Heads	–	8	8	8	8	–
Number of Experts	–	–	–	–	8	8
Number of Shared Experts	–	–	–	–	1	2
Top-K (for MoE)	–	–	–	–	1	6
RoPE Theta	–	500000	500000	500000	500000	
FFN Dimension Multiplier	–	–	1.3	1.3	1.3	
Multiple Of	–	–	1024	1024	1024	
RoPE Theta	–	500000	500000	500000	500000	
Distributed Setup						
Data Parallel Dimension	8	8 (16 for 16×16 mesh)	8	8	8	8
Pipeline Parallel Dimension	12	16	8	22	16	15
Total Workers	96	128 (256 for 16×16 mesh)	64	176	128	90
Optimizer						
Type	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
Learning Rate	6e-4	6e-4	6e-4	6e-4	4e-3	2.2e-4
Epsilon	1e-8	1e-8	1e-8	1e-8	1e-15	1e-8
Learning Rate Scheduler						
Warmup Steps	100	100	100	500	600	600
Decay Ratio	0.8	0.8	0.8	0.8	–	0.8
Decay Type	Linear	Linear	Linear	Linear	Linear	Cosine
Minimum LR	0.0	0.0	0.0	0.0	0.1	0.1
Training						
Worker Batch Size	8	12	10	12	16	16
Global Batch Size	64	96 (192 for 16×16 mesh)	80	96	128	128
Sequence Length	1024	1024	1024	1024	1024	1024
Gradient Clipping	1.0	1.0	1.0	1.0	1.0	1.0

Table 14: Hyper-parameters of the proposed verification approach (Alg. 1).

Parameter	NanoGPT		Llama-3					
	8×12 Mesh, 0.25B		8×16 Mesh, 0.6B		16×16 Mesh, 0.6B		8×8 Mesh, 1.2B	
	Activation	Gradient	Activation	Gradient	Activation	Gradient	Activation	Gradient
Momentum Beta (β)	0.90	0.80	0.90	0.80	0.90	0.80	0.99	0.85
Initial IQR Multiplier (k_0)	1.5	3.0	1.5	3.0	1.5	3.0	3.0	3.0
Target FP-Rate (α)	0.01	0.001	0.0001	0.001	0.01	0.01	0.001	0.05
Adaptive Max Iterations ($N_{\max}$)	10	10	10	10	10	10	10	10
Adaptive Grow Factor (γ_g)	1.1	1.01	1.1	1.01	1.1	1.01	1.25	1.01
Adaptive Shrink Factor (γ_s)	0.9	0.99	0.9	0.99	0.9	0.99	0.95	0.99
Adaptive Min Distance Multiplier (Λ)	0.35	0.15	0.15	0.05	0.25	0.05	0.2	0.05
Adaptive Epsilon (ε)	0.01	0.001	0.001	0.00005	0.0001	0.0001	0.01	0.0001

F. 2 详细实验结果

流水线并行中的激活与梯度攻击分析。在第5节所介绍的部分攻击基础上，我们现在提供第2 节中所有激活与梯度攻击的全面验证结果，以完整刻画其行为。表1展示了 C4数据集上的完整结果，而表15 和16 分别展示了 FineWeb 和 OpenWebText 数据集上的性能。

我们的全面评估揭示了若干重要见解。首先，在分布式流水线并行训练中，激活操纵构成了与梯度操控同等重要的威胁，证实在拜占庭容错系统中必须谨慎考虑这两类攻击向量。其次，当攻击逃避检测时，其偏离基线原始训练的程度仍可忽略不计，这直接支持了我们在定理3.1中提出的理论分析。这种一致性在三个数据集上均成立，展现了我们方法在多样化训练场景中的鲁棒性与通用性。

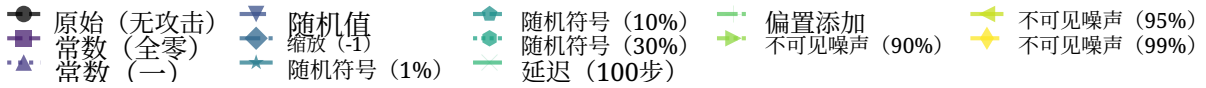
值得特别关注的是，以 1.0 的检测速度检测到的攻击。这表明，尽管我们在第 3.1 节提出了宽容策略，这些攻击仍会造成足够显著的偏差，因此有必要立即排除恶意工作节点。图 7 至 10 中的训练和验证损失曲线进一步说明，我们的指数移动平均验证方法能够有效控制恶意行为，并在整个训练过程中将性能维持在接近原始基线的水平。这些结果共同验证了我们提出的验证框架的有效性和泛化能力。

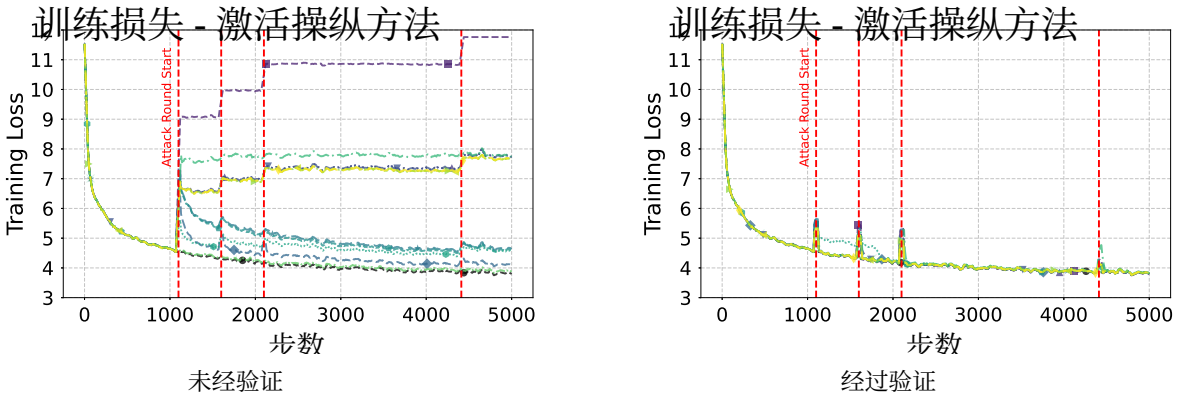
表15：Llama-3-0.6B 在 FineWeb 数据集上的攻击检测性能。所示度量指标包括精确率、召回率、F1分数（均为百分比）、平均检测速度（以迭代次数计）和验证损失。

模式攻击		SENTINEL（我们的方法）					无验证
		Pr. (%) ↑	Re. (%) ↑	F1 分数 (%) ↑	检测速度 ↓	验证损失 ↓	验证损失 ↓
-	无（原始方法）	100.0	100.0	100.0	N/A	3.818	3.840
Activation Manipulation	常数（零）	100.0	100.0	100.0	6.43	3.819	11.761
	常数（一）	100.0	100.0	100.0	6.61	3.814	7.793
	随机值	96.0	100.0	98.0	6.46	3.831	7.793
	缩放（$\alpha = -1$）	100.0	100.0	100.0	6.29	3.827	4.121
	随机符号（1%）	100.0	100.0	100.0	6.42	3.825	4.693
	随机符号（10%）	51.1	100.0	67.6	3.58	3.829	4.716
	随机符号（30%）	92.3	100.0	96.0	7.5	3.826	4.564
	延迟（100 步）	85.7	100.0	92.3	14.83	3.832	7.692
	偏置添加	86.4	79.2	82.6	10.68	3.828	3.898
	不可见噪声（90%）	100.0	100.0	100.0	6.38	3.824	7.709
	不可见噪声（95%）	100.0	100.0	100.0	6.29	3.824	7.713
	不可见噪声（99%）	100.0	100.0	100.0	6.33	3.829	7.712
Gradient Manipulation	常数（零）	100.0	100.0	100.0	6.33	3.815	3.949
	常数（一）	92.3	100.0	96.0	1.14	3.824	10.726
	随机值	100.0	100.0	100.0	1.38	3.831	9.359
	缩放（$\alpha = -1$）	0.0	0.0	0.0	N/A	3.888	3.901
	随机符号（1%）	0.0	0.0	0.0	N/A	3.989	3.999
	随机符号（10%）	0.0	0.0	0.0	N/A	3.990	4.004
	随机符号（30%）	0.0	0.0	0.0	N/A	3.941	3.966
	延迟（100 步）	100.0	100.0	100.0	7.29	3.817	10.017
	偏置添加	100.0	100.0	100.0	1.14	3.828	10.573
	不可见噪声（90%）	100.0	75.0	85.7	52.94	3.954	4.212
	不可见噪声（95%）	100.0	75.0	85.7	140.89	3.959	4.217
	不可见噪声（99%）	100.0	75.0	85.7	209.56	3.949	4.197

表16：Llama-3-0.6B 在 OpenWebText 数据集上的攻击检测性能。所示度量指标包括精确率、召回率、F1 分数（均为百分比）、平均检测速度（以迭代次数计）和验证损失。

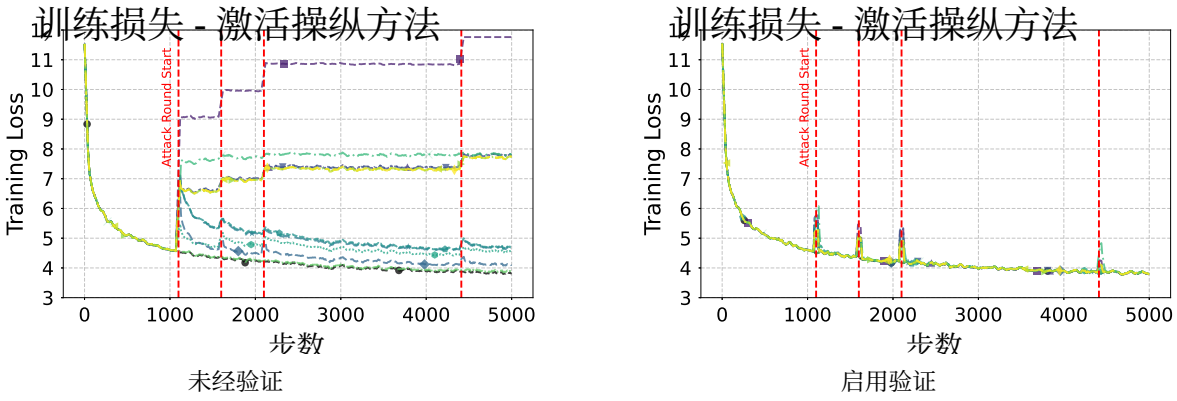
模式攻击		SENTINEL（我们的方法）					无验证
		PR. (%) ↑	RE. (%) ↑	F1 分数 (%) ↑	检测速度 ↓	验证损失 ↓	验证损失 ↓
-	无（原始方法）	100.0	100.0	100.0	N/A	3.773	3.778
ACTIVATION MANIPULATION	常数（零）	100.0	100.0	100.0	6.33	3.773	11.761
	常数（一）	100.0	100.0	100.0	6.29	3.779	7.820
	随机值	100.0	100.0	100.0	6.29	3.777	7.821
	缩放（$\alpha = -1$）	100.0	100.0	100.0	6.29	3.776	4.016
	随机符号（1%）	100.0	100.0	100.0	6.21	3.774	4.614
	随机符号（10%）	100.0	100.0	100.0	6.29	3.779	4.578
	随机符号（30%）	96.0	100.0	98.0	25.62	3.779	4.524
	延迟（100 步）	88.9	100.0	94.1	10.74	3.790	7.701
	偏置添加	91.7	91.7	91.7	14.0	3.782	3.843
	不可见噪声（90%）	100.0	100.0	100.0	6.25	3.777	7.677
	不可见噪声（95%）	100.0	100.0	100.0	6.5	3.783	7.674
	不可见噪声（99%）	100.0	100.0	100.0	6.29	3.780	7.678
GRADIENT MANIPULATION	常数（零）	100.0	100.0	100.0	5.7	3.775	3.908
	常数（一）	100.0	100.0	100.0	1.0	3.774	10.722
	随机值	100.0	100.0	100.0	1.0	3.780	9.611
	缩放（$\alpha = -1$）	0.0	0.0	0.0	N/A	3.847	3.860
	随机符号（1%）	12.1	16.7	14.0	1516.33	3.990	3.957
	随机符号（10%）	5.3	4.2	4.7	2646.0	3.967	3.952
	随机符号（30%）	0.0	0.0	0.0	N/A	3.901	3.906
	延迟（100 步）	100.0	100.0	100.0	7.09	3.771	11.508
	偏置添加	100.0	100.0	100.0	1.0	3.777	10.947
	不可见噪声（90%）	100.0	87.5	93.3	92.95	3.877	4.178
	不可见噪声（95%）	100.0	91.7	95.7	136.42	3.828	4.098
	不可见噪声（99%）	100.0	87.5	93.3	64.42	3.858	4.123





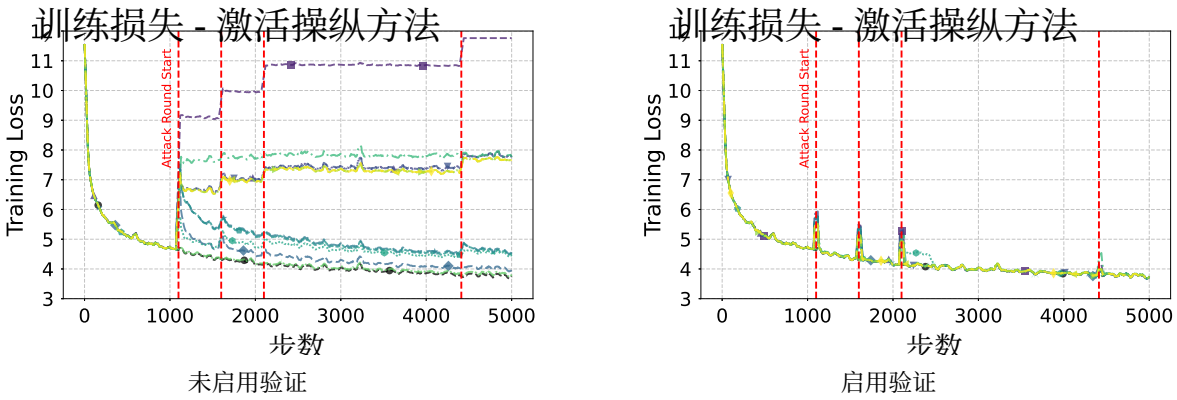
未经验证　　经过验证

(a) C4数据集



未经验证　　启用验证

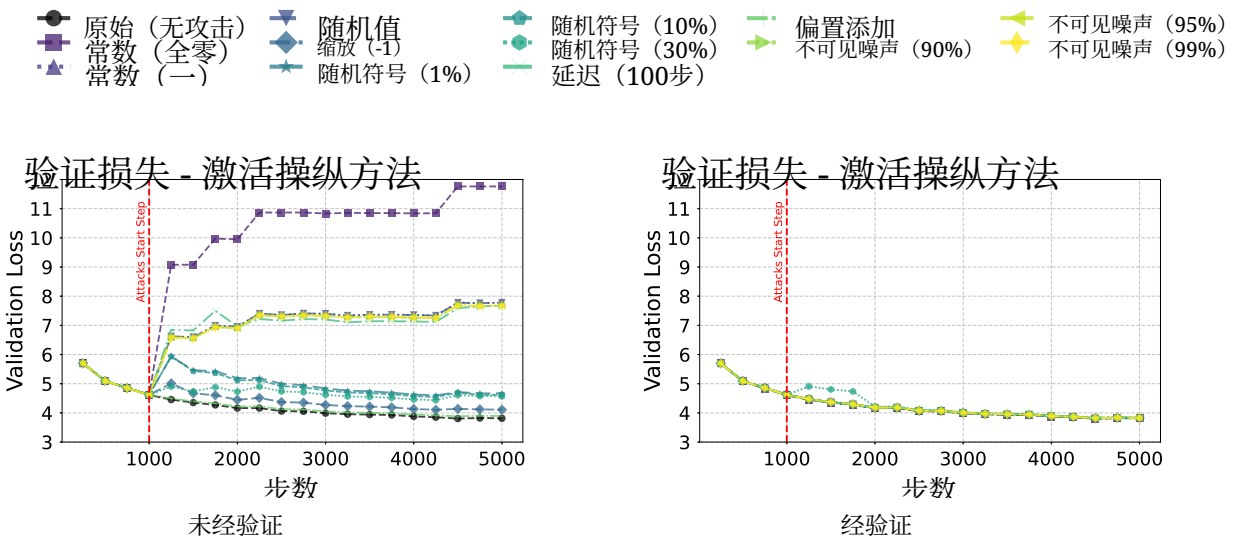
(b) FineWeb 数据集



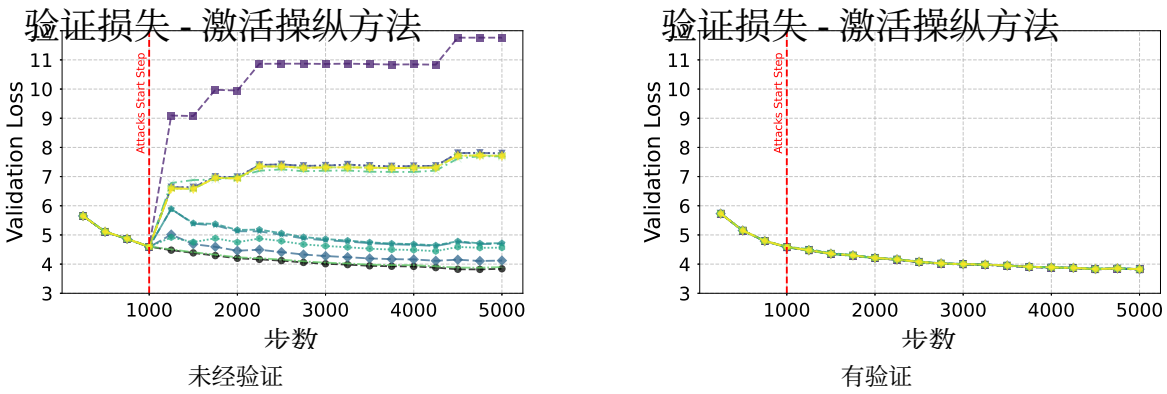
未启用验证　　启用验证

(c) OpenWebText 数据集

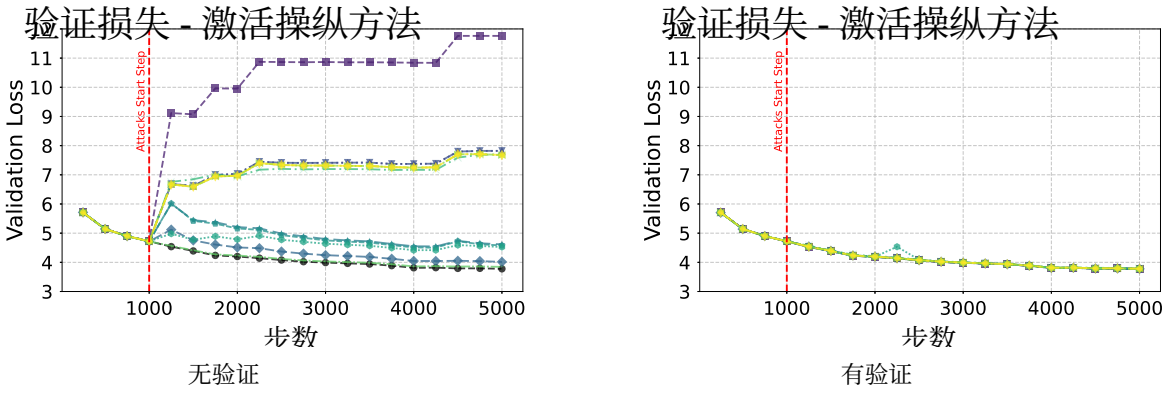
图 7：**训练损失**对比了我们提出的验证机制与基线原始训练在遭受**激活操纵攻击**时的表现。我们使用三个数据集（C4、FineWeb 和 OpenWebText）在 Llama-3-0.6B 上进行了评估。虚线表示攻击起始点，此时 6 个随机选出的节点开始在协同拜占庭攻击中提交对抗性激活。我们的验证方法能够保持稳定收敛，而基线方法在攻击下则遭受显著性能下降。



（a）C4数据集

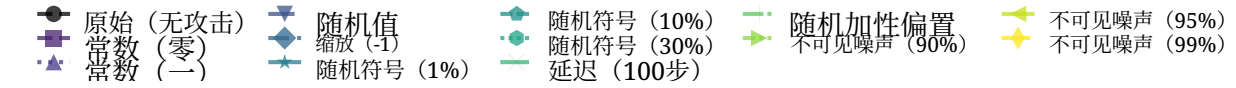


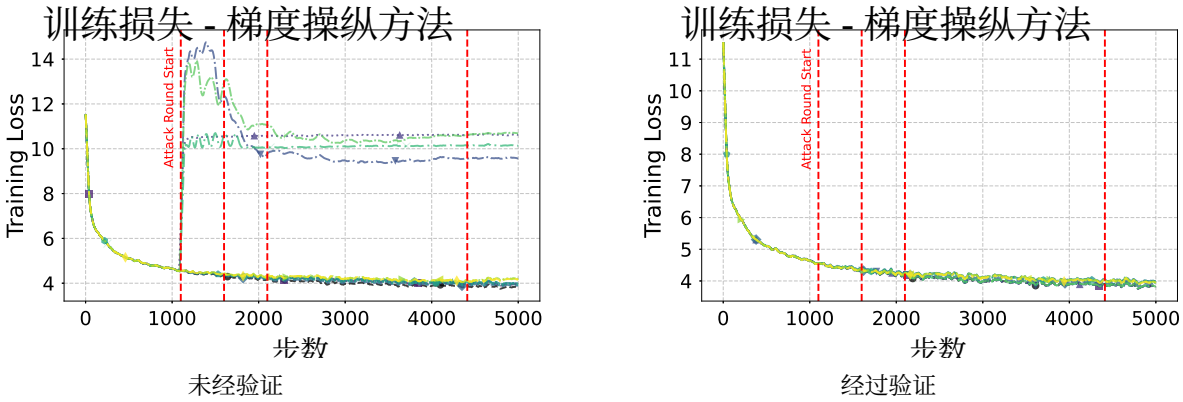
(b) FineWeb 数据集



(c) OpenWebText 数据集

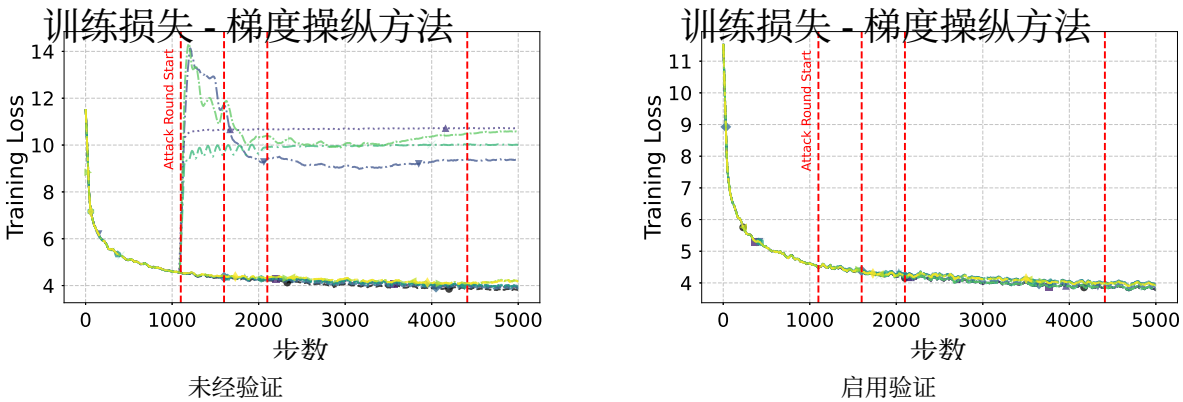
图 8：**验证损失** 对比我们的验证机制与基线原始训练在**激活操纵攻击**下的表现。我们在三个数据集（C4、FineWeb 和 OpenWebText）上评估 Llama-3-0.6B。红色虚线标记从预热到攻击阶段的过渡，此时拜占庭节点开始提交对抗性激活。我们的验证方法保持稳定的验证性能，而在没有验证的情况下，基线在攻击后出现显著下降。





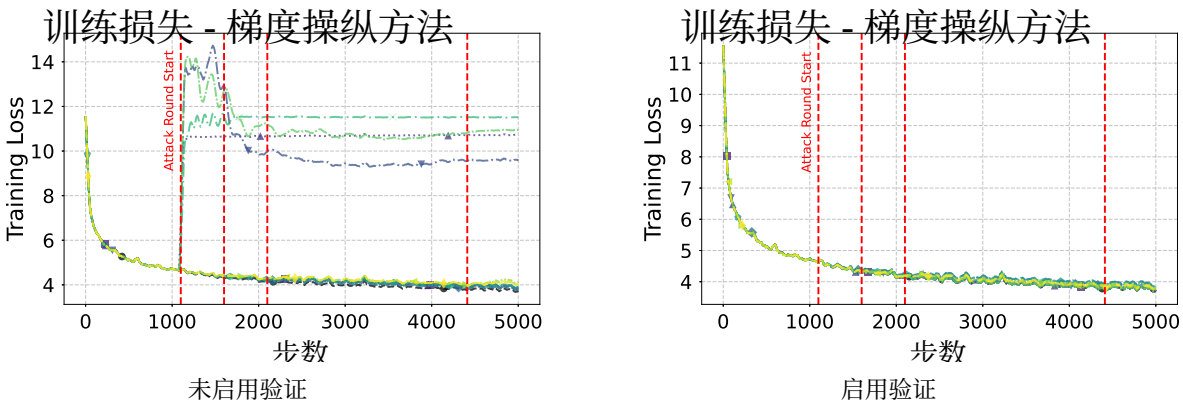
未经验证　　经过验证

(a) C4数据集



未经验证　　启用验证

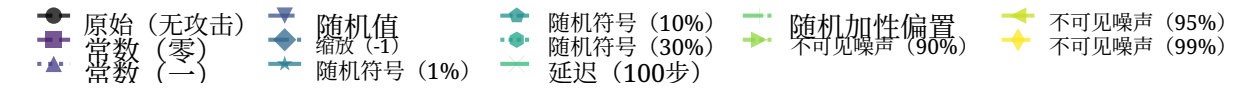
(b) FineWeb 数据集

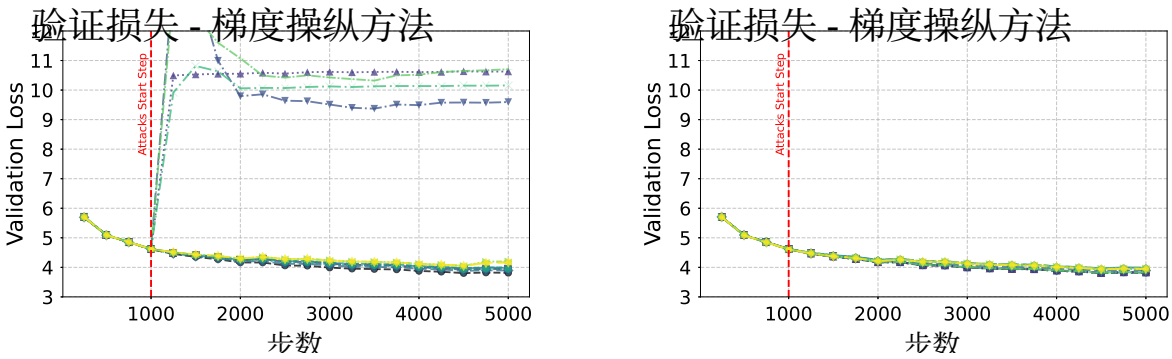


未启用验证　　启用验证

(c) OpenWebText 数据集

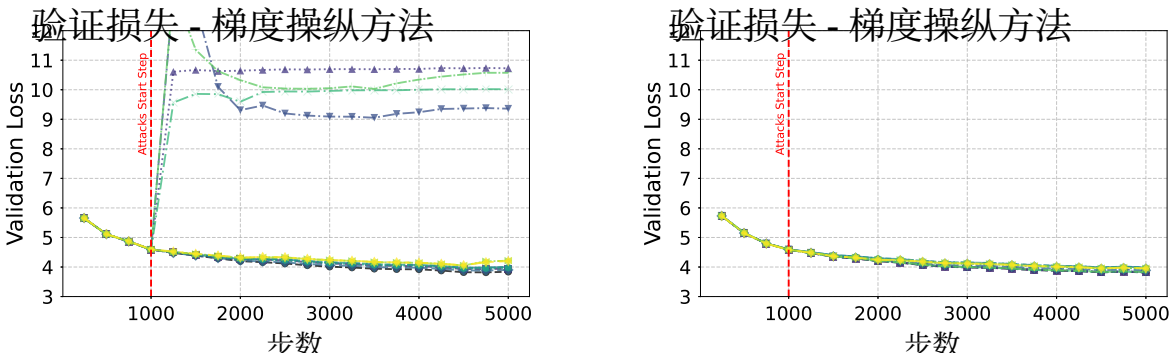
图 9：**训练损失** 对比了我们提出的验证机制与基线原始训练在**梯度操纵攻击**下的表现。我们使用三个数据集（C4、FineWeb 和 OpenWebText）对 Llama-3-0.6B 进行评估。竖直虚线表示攻击起始点，此时 6 个随机选出的节点开始在协同拜占庭攻击中提交对抗梯度。我们的验证方法能够保持稳定收敛，而基线在攻击下则遭受显著性能下降。





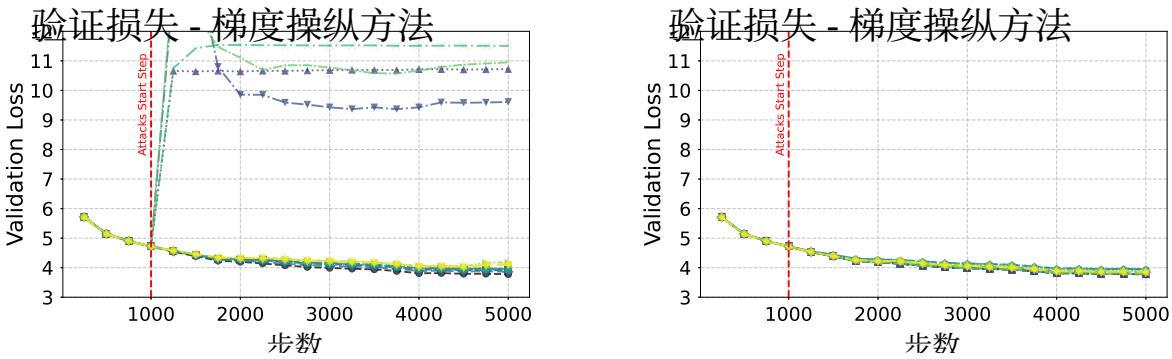
无验证　　有验证

(a) C4数据集



无验证　　启用验证

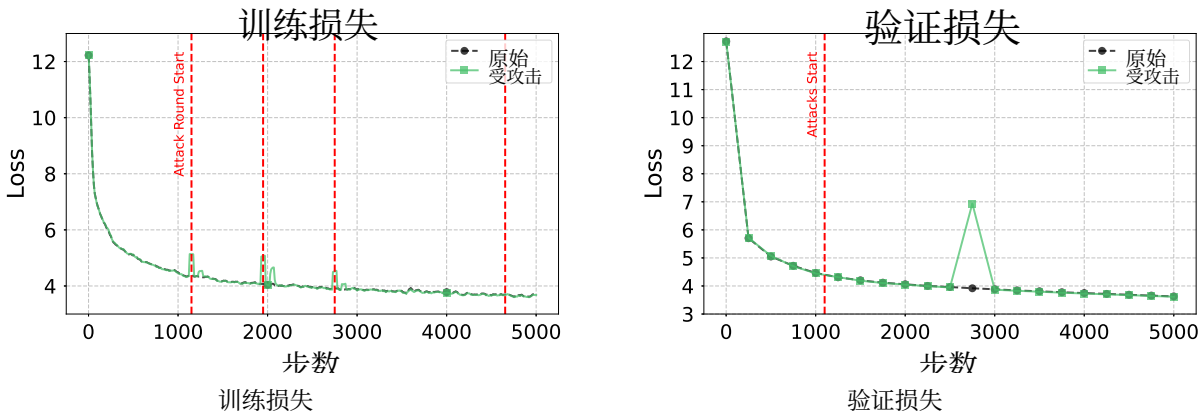
(b) FineWeb 数据集



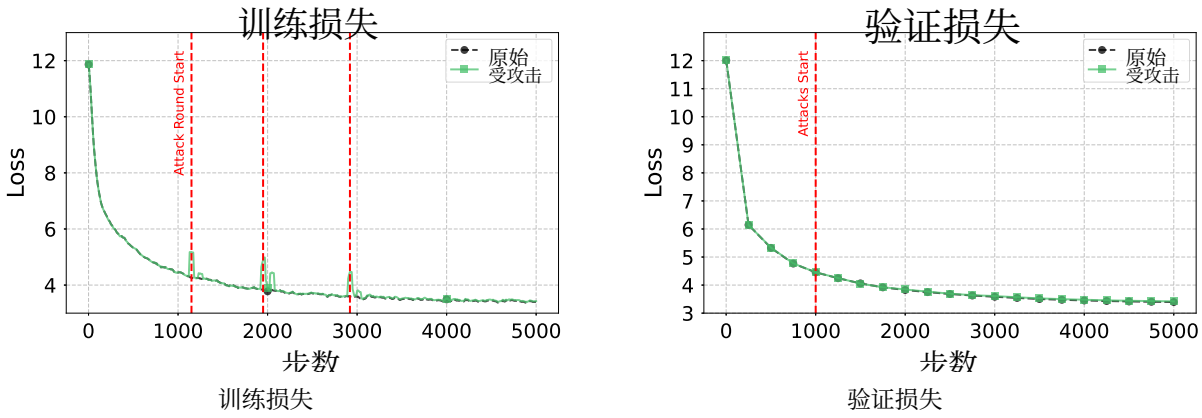
未启用验证　　启用验证

(c) OpenWebText 数据集

图 10：**验证损失** 对比了我们的验证机制与基线原始训练在**梯度操纵攻击**下的表现。我们在三个数据集（C4、FineWeb 和 OpenWebText）上评估了 Llama-3-0.6B。红色虚线标记了从预热阶段过渡到攻击阶段的时刻，此时拜占庭节点开始提交对抗梯度。我们的验证方法能够维持稳定的验证性能，而未采用该方法时，基线模型在遭受攻击后则出现了显著退化。



(a) Llama-4-0.4B

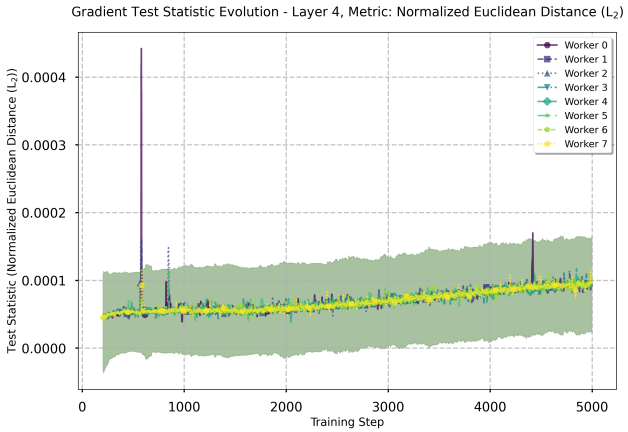


(b) DeepSeek-V3-1B

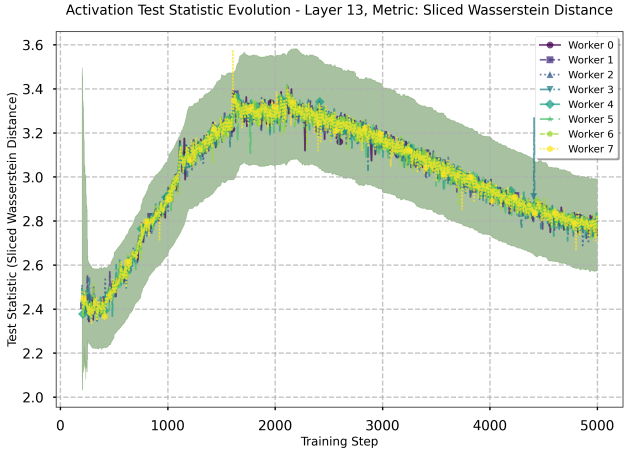
图11：采用SENTINEL训练的基于混合专家的模型在 FineWeb-EDU 数据集上经过 5k 次迭代的训练和验证损失演变。混合激活攻击场景假设每个流水线阶段中有 37.5% 的拜占庭节点，其中随机选取 30% 的节点，在随机采样的迭代中执行随机选择的攻击。每个节点使用不同的激活操纵方法（如第 2.1 节所述）。

基于 MoE 的模型的训练损失和验证损失演变。 在第 5 节中，我们讨论了 SENTINEL如何扩展到 Llama-4 和 DeepSeek-V3 等基于 MoE 的模型。这里，我们展示了整个训练过程中的训练损失和验证损失演变。如图 11所示，在一轮攻击开始后，训练损失开始呈现较高的值。然而，验证损失仍像原始基线一样持续下降，且性能与其相当。

自适应偏差界的演变。我们的方法采用基于四分位距的自适应阈值机制（见算法5），为每个受监测指标动态调整可接受的偏差界限。为展示该方法的有效性，我们呈现了 Llama-3-0.6B 模型中两个代表性层的自适应阈值变化，以及各工作节点记录的相应偏差。图12a展示了梯度延迟攻击下的结果，而图12b 展示了激活随机符号攻击期间的行为。结果表明，在良性训练阶段，自适应界限能够有效涵盖诚实工作节点的正常运行行为。关键的是，当恶意工作节点发起攻击时，自适应界限使验证节点能够立即检测并标记异常行为，从而在分布式训练过程中提供强大的防护，抵御对抗性干扰。



（a）梯度延迟攻击下第 4 层梯度的归一化 L_2 距离变化。工作节点 0 在第 4411 次迭代发起攻击，并立即被标记。



（b）第 13 层激活值的切片 Wasserstein 距离变化在随机符号攻击（1%）下。工作节点 7 和 3 分别在第 1600 次和第 4411 次迭代发起攻击，并立即被标记。

图 12：自适应偏差界和工作节点统计信息的演变。所提出的阈值机制能够适应自然分布变化，同时检测拜占庭行为。

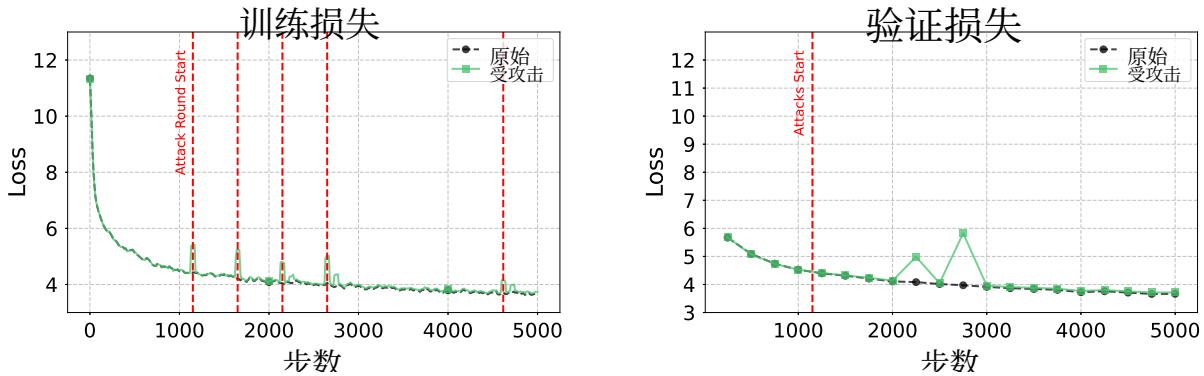
大规模实验结果。在第5节中，我们讨论了将实验扩展到另外两种设置：（1）一个包含 256 个工作节点的 16×16 网格拓扑，以及（2）一个包含 64 个工作节点的网格上的 1.2B 参数模型 8×8。此处我们展示了两种配置在所测试的所有攻击类型和恶意比例下的完整结果。表17列出了各实验条件下的详细性能指标，包括验证损失、精确率、召回率和 F1 分数。全面的实验结果证实了我们的主要发现：所提出的验证方法能够有效检测试图扰乱分布式训练的恶意行为者。

此外，我们在表 4中证明，可以轻松扩展模型规模，而 SENTINEL 在这些设置下仍能成功运行。图 13 展示了训练过程中训练损失和验证损失的变化。这表明，我们的方法能够缓解恶意工作节点试图使模型发散的影响，并使训练得以继续，其表现与原始训练相似。

更长时间的训练与替代架构。我们的方法自然引出了两个问题：在更长时间的训练方案下，我们的验证方法是否仍然稳定；以及它能否推广到我们最初实验之外的其他 Transformer 架构。为了解决这些问题，我们进行了扩展

表 17：在 C4 数据集上进行大规模 Llama-3 训练时的攻击检测性能。所示度量指标包括精确率、召回率、F1分数（均以百分比表示）、平均检测速度（以迭代次数表示）和验证损失。对于所有实验，我们假设每个阶段有 37.5% 的拜占庭工作节点（因此，对于 16×16 网格，每个阶段有 6 个恶意工作节点和 10 个诚实工作节点；而对于 8×8 网格，其比例为 3:5。）

设置	模式攻击		SENTINEL（我们的方法）				
			Pr. (%) ↑	Re. (%) ↑	F1 分数 (%) ↑	检测速度 ↓	验证损失 ↓
0.6B on 16 × 16 Mesh	GradientActivation	随机值	100.0	100.0	100.0	7.96	3.900
		延迟（100 步）	85.7	100.0	92.3	14.39	3.945
		偏置添加	100.0	25.6	40.8	65.15	3.981
		不可见噪声（99%）	100.0	100.0	100.0	7.96	3.898
	GradientActivation	随机值	100.0	100.0	100.0	124.08	3.895
		延迟（100 步）	100.0	100.0	100.0	9.05	3.890
		偏置添加	100.0	100.0	100.0	1.69	3.894
		不可见噪声（99%）	98.7	93.6	96.0	14.27	3.915
1.2B on 8 × 8 Mesh	-	无（原始方法）	100.0	100.0	100.0	N/A	3.723
	GradientActivation	随机值	100.0	100.0	100.0	4.33	3.723
		延迟（100 步）	37.5	100.0	54.5	67.0	3.774
		偏置添加	0.0	0.0	0.0	N/A	3.738
		不可见噪声（99%）	100.0	100.0	100.0	4.33	3.727
	GradientActivation	随机值	100.0	100.0	100.0	1.0	3.726
		延迟（100 步）	100.0	100.0	100.0	1.0	3.722
		偏置添加	0.0	0.0	0.0	N/A	3.805
		不可见噪声（99%）	100.0	100.0	100.0	9.2	3.725



训练损失　　验证损失

图 13：使用 SENTINEL在 FineWeb 数据集上训练 Llama-3-4B 5k 次迭代的训练损失和验证损失演变。混合激活攻击场景假设每个流水线阶段均有 37.5% 的拜占庭节点，其中随机选取的 20% 节点会在随机抽取的一次迭代中执行随机选择的攻击。每个节点采用不同的激活操纵方法（如第 2.1 节所述）。

对两种不同模型进行评估：NanoGPT-0.25B 和 Llama-3-0.6B，每种模型训练 30,000 次迭代。根据大语言模型训练中既有的缩放定律（7× 参数量），这一训练时长相当于 NanoGPT-0.25B 处理约 2B 个代币（5× 参数量），以及 Llama-3-0.6B 处理约 3B 个代币（Hoffmann 等人，2022）。

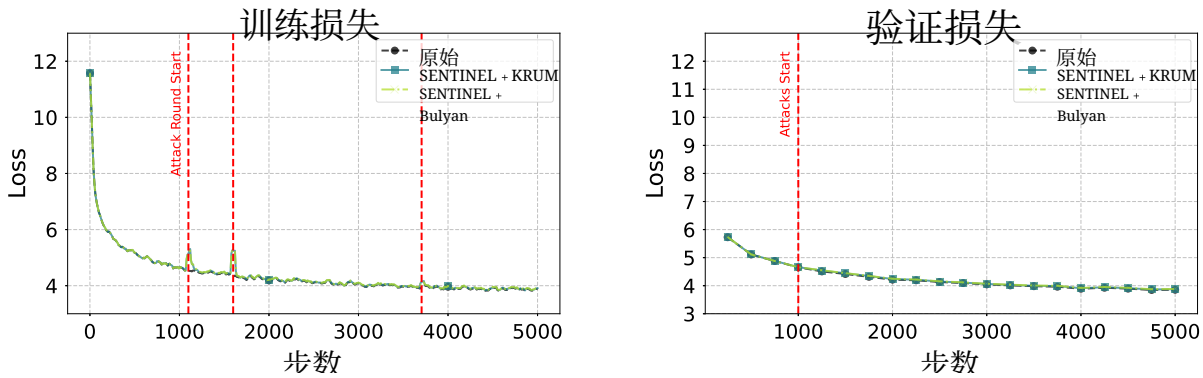
我们通过模拟一个具有挑战性的对抗环境来评估本方法的鲁棒性：在每个 Transformer 阶段，50% 的节点为恶意节点。在每个攻击轮次，一个随机选出的恶意节点会从表1 中随机选择一种攻击并执行，且假设“无串谋”。表18 总结了两种架构在这一延长训练期间的检测性能。

我们的结果表明，两种模型架构和全部三个数据集均表现出一致的稳定性，并在攻击检测上取得了较高的F1分数（$> 81\%$）。重要的是，我们观察到，在所有成功检测到的攻击类型中，检测速度的中位数为 5.0 次迭代，这与我们可接受的违规次数阈值完全一致。这表明，我们可以在预定义的容忍窗口内检测出大多数具有显著训练影响的攻击。即使某些攻击未被检测到，它们对训练收敛的影响也可忽略不计，如图15和16所示。在整个训练过程中，两种模型在我们的验证方法下的验证损失都紧密跟随原始基线，从而印证了定理3.1 中的理论分析，并证明了该方法的架构灵活性与长期稳定性。

数据并行防御方法与 SENTINEL 的互补性。在全文中，我们详细阐述了先前拜占庭容错文献提出的数据并行领域防御方法（例如，参见（Gorbunov 等人，2022；Malinovsky 等人，2024年；Mhamdi 等人，2018年））如何与 SENTINEL 互为补充。SENTINEL 通过验证前向与反向传播期间传输的激活值和激活梯度，保障流水线并行维度的安全。为展示这种互补性，我们将本方法与两种用于防御参数梯度攻击的知名鲁棒聚合技术结合。具体而言，我们使用 Krum（Blanchard 等人，2017年）和 Bulyan（Mhamdi 等人，2018年），而非原始梯度平均。我们的目标是表明，这些方法在与 SENTINEL 正交的维度上运行，因此不会对 SENTINEL 的运作产生负面影响。图 14 展示了针对混合攻击训练的 Llama-3-0.6B 模型的训练损失和验证损失。如图所示，**采用鲁棒聚合方法不会干扰 SENTINEL 的运作**，而且本方法能够确保收敛率与原始的未受攻击基线一致。

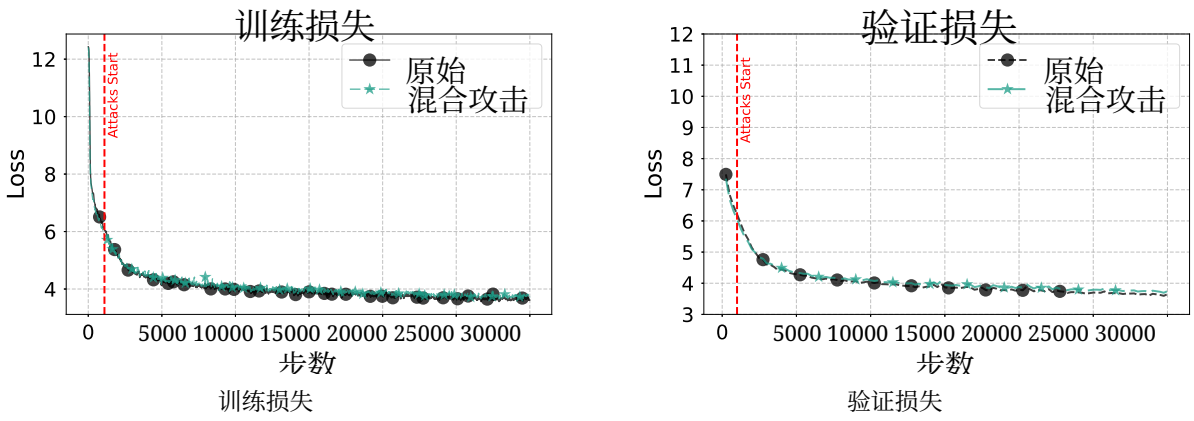
表18：针对混合攻击训练 NanoGPT-0.25B 和 Llama-3-0.6B 3万次迭代的检测性能。

模型	数据集	SENTINEL（我们的方法）						原始
		Pr. (%) ↑	Re. (%) ↑	F1 分数 (%) ↑	中位速度 ↓	平均速度 ↓	验证损失 oss ↓	验证损失 ↓
NanoGPT	C4	91.2	86.1	88.6	5.0	44.07	3.747	3.650
	FW	91.2	86.1	88.6	5.0	62.13	3.752	3.731
	OW	73.3	91.7	81.5	5.0	320.43	3.571	3.531
Llama-3	C4	76.0	86.4	80.9	5.0	263.51	3.357	3.347
	FW	88.6	88.6	88.6	5.0	108.73	3.465	3.459
	OW	78.0	88.6	83.0	5.0	44.06	3.316	3.313

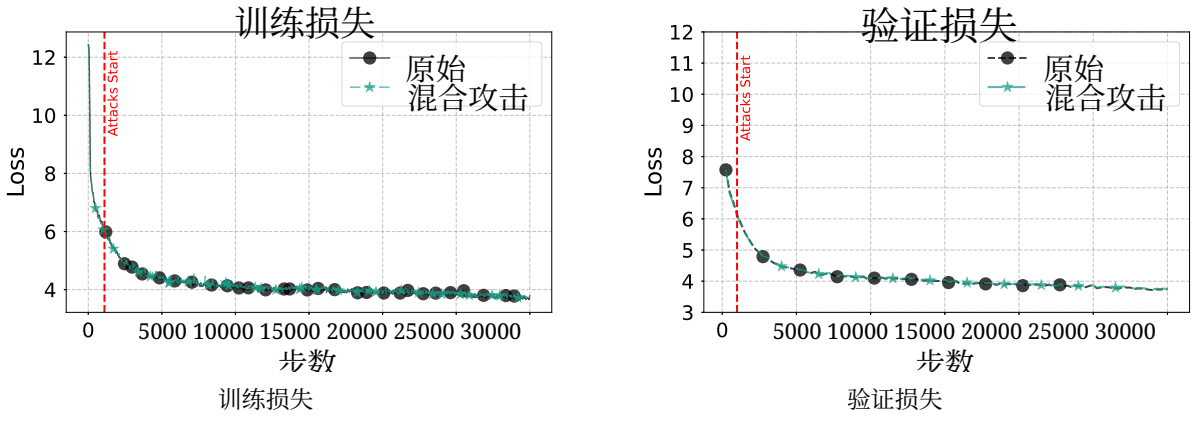


训练损失　　验证损失

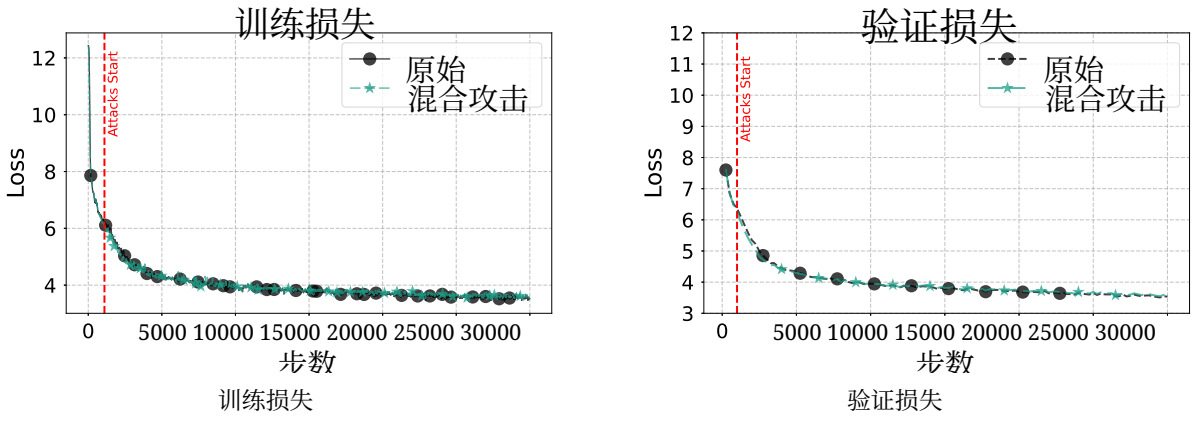
图 14：在采用鲁棒聚合方法进行梯度平均的情况下，Llama-3-0.6B 在 FineWeb 数据集上训练 5k 次迭代的训练和验证损失演变。混合攻击场景假设每个流水线阶段都有 37.5% 的拜占庭节点，其中 33% 的随机选定节点在随机抽取的迭代中执行随机选定的攻击。每个节点采用不同的操纵方法（如第 2.1 节所述）。



（a）C4数据集

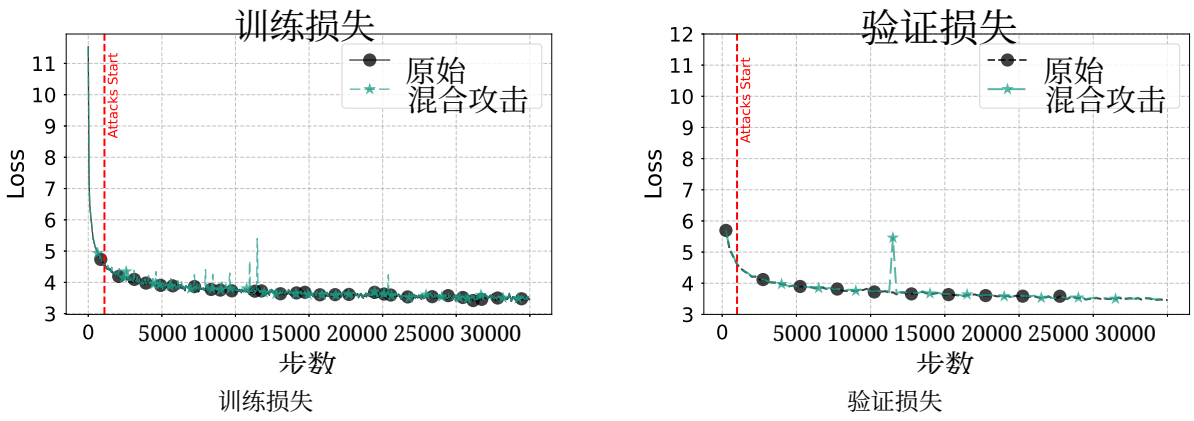


(b) FineWeb 数据集

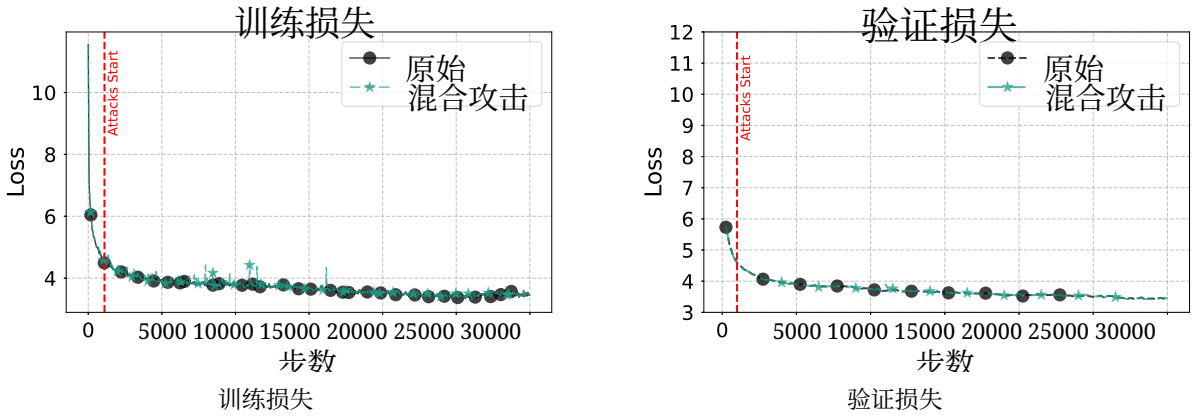


(c) OpenWebText 数据集

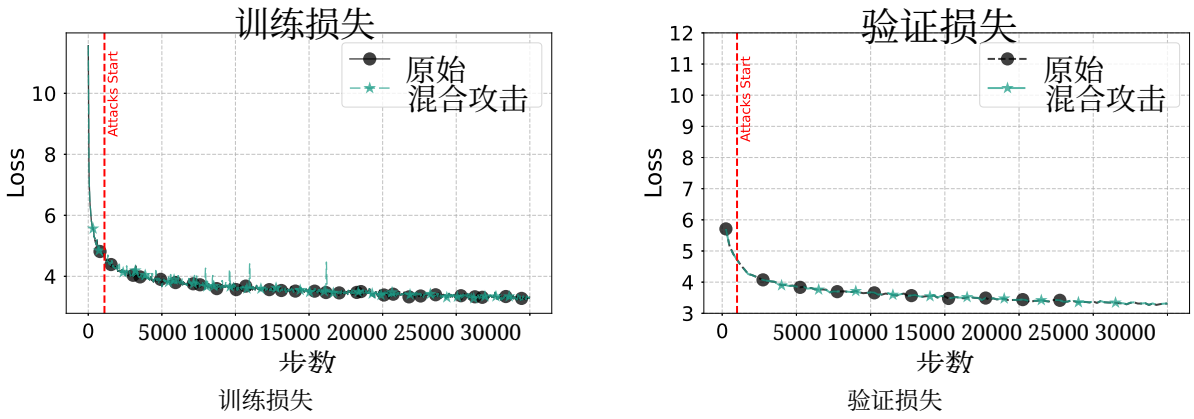
图 15：在 C4、FineWeb 和 OpenWebText 数据集上，**NanoGPT-0.25B**模型经过 3 万次迭代的训练和验证损失演变。混合攻击场景假设每个流水线阶段有 50% 的拜占庭节点，其中随机选择的一个节点在随机抽取的迭代中执行随机选择的攻击。每个节点具有不同的模式（激活或梯度）和操纵方法（如第2.1节所述）。



(a) C4 数据集



(b) FineWeb 数据集



(c) OpenWebText 数据集

图 16：在 3万次迭代中，**Llama-3-0.6B** 模型在 C4、FineWeb 和 OpenWebText 数据集上的训练和验证损失演变。混合攻击场景假设每个流水线阶段有 50% 的拜占庭节点，其中随机选择一个节点，在随机抽取的迭代中执行随机选定的攻击。每个节点具有不同的模式（激活或梯度）和操纵方法（如第 2.1 节所述）。

G. SENTINEL 在实际环境中的应用：使用 SWARM 并行验证去中心化大语言模型训练

本节将详细介绍 SENTINEL 向 SWARM 并行（Ryabinin 等人，2023年）的适配，并展示其在大语言模型去中心化训练中的能力。我们首先概述 SWARM 的运行机制。随后，我们将展示我们的验证机制与分布式 SWARM 训练中所采用的通信高效压缩技术的兼容性。我们将说明 SENTINEL 如何利用 SWARM 现有的训练器节点架构与其基础设施集成。此外，我们还将分析在 SWARM 的随机连线机制存在时验证所发挥的关键作用；该机制引入了超出传统流水线并行之外的额外故障模式。最后，我们将介绍 SWARM 实验的完整实验设置和详细结果。

G.1 SWARM 并行概述

如第 5.2 节简要讨论的那样，SWARM 并行可以视为一种随机数据并行/流水线并行训练方法。总体而言，每个工作节点会被随机分配模型中的一个层/阶段进行处理（流水线并行轴），而其他工作节点则针对同一阶段处理不同的微批次（数据并行轴）。与 `torch.distributed` (Li 等人，2020；Paszke 等人，2019）等分布式训练框架所使用的传统固定网格不同，SWARM 并行采用工作节点池的形式，每个阶段都有可用的工作节点来处理传往后续阶段的数据。

在这种随机环境中，训练协调由所谓的训练器节点负责。每个训练器节点负责端到端地处理一个微批次的数据。具体而言，每个训练器节点接收一个互不重叠的数据微批次，并将其发送给阶段 0 的工作节点。工作节点处理完数据后，训练器接收结果并将其转发到下一阶段。训练器会持续这一过程，直到该微批次在前向传播中经过所有层。随后，训练器按相反顺序遍历各阶段，开始反向传播。采用这种架构后，工作节点无需保存用于反向传播的中间激活值，因为训练器会保留所有必要信息，并能在反向传播期间将这些信息发送给不同的工作节点。之后，工作节点在本地累积参数梯度，并在与同一阶段所有现有工作节点执行全归约操作后更新参数。

训练节点使用分布式哈希表（DHT）来路由其微批次。由于各阶段的设备处理数据的速度不同，训练器会在分布式哈希表中维护工作节点负载和吞吐量信息，以便所有其他训练器知道何时发送信号，以及每个阶段的哪个工作节点处理数据最有效率。这一机制是 SWARM 中随机连线的核心，也使其区别于固定网格数据并行/流水线并行方法（Ryabinin 等人，2023年）。如图所示，SWARM 系统能够容纳许多训练节点，因为每个节点只处理一个微批次，而且每个阶段通常都有多个设备能够处理数据批次。SWARM 的示意图请参见图17a。算法6还提供了 SWARM 中训练节点工作方式的伪代码。

G.2 子空间压缩

常规 SWARM 并行支持去中心化训练，但由于带宽要求较高，训练速度较慢（Ryabinin 等人，2023年）。为降低这一要求并加快训练，Ramasinghe 等人（2025年）近期的研究表明，借助无损压缩，即使速度低至 60 Mbps，也有望进行训练。具体而言，Ramasinghe 等人（2025年）指出，大多数预训练 Transformer 的前馈层和注意力投影层已呈现低秩性质。他们利用这一发现，基于共享子空间构建了一种无损压缩机制，从而加快在流水线并行中层与层之间传递激活值和梯度的速度。

简单来说，给定一个由阶段 s的工作节点处理过的信号 $\boldsymbol{h}^{(s)} = f_s(\boldsymbol{h}^{(s-1)})$，他们不再传输 $\boldsymbol{h}^{(s)} \in \mathbb{R}^{b\times n\times d}$，而是使用子空间压缩矩阵 $\boldsymbol{U}_k$ 对其进行压缩，将其投影到一个

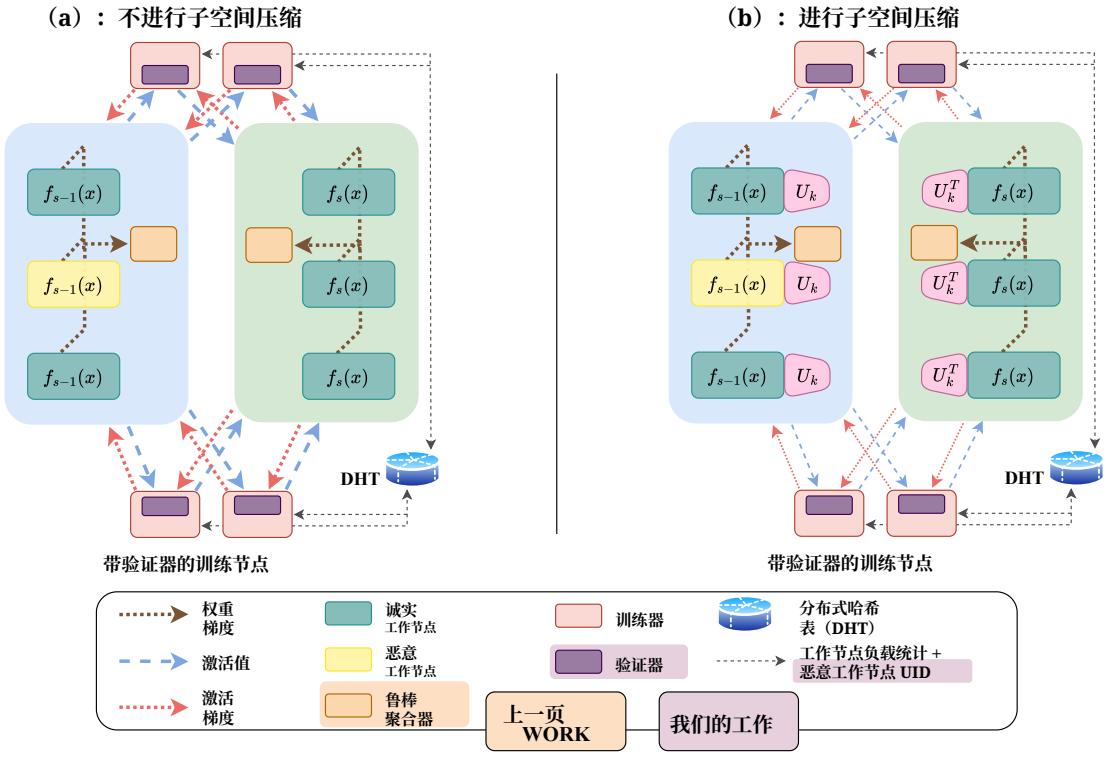


图 17：(a) SWARM（Ryabinin 等人，2023年）并行化利用流水线并行和数据并行来训练神经网络。与传统的数据并行／流水线并行方法相比，SWARM 利用训练节点随机地将输入数据的微批次路由经过模型层。训练节点借助分布式哈希表（DHT）传递工作节点负载／吞吐量信息，从而提高 SWARM 内的设备利用率。(b) 子空间压缩（Ramasinghe 等人，2025年）通过在工作节点设备的边界处加入无损压缩，补充 SWARM，以加快激活值和激活梯度与训练节点之间的通信。SENTINEL 可适配这两种设置并实现工作节点验证。它自然地部署在训练节点内，由训练节点协调工作节点之间的信号传输。

k维子空间，其中 $k \ll d$ (Ramasinghe 等人，2025年)：

$$\boldsymbol{h}^{(s)}_{\text{compressed}} = (\boldsymbol{h}^{(s)} - \text{PE} - \mathbf{T}_{\text{fixed}}[t_{1:n}, :])\boldsymbol{U}_k. \tag{72}$$

这里，PE 表示位置嵌入，$\mathbf{T}_{\text{fixed}}[t_{1:n}, :]$ 是固定的词元嵌入。在第 $s+1$ 层，接收方使用（Ramasinghe 等人，2025年）对其进行解压缩：

$$\boldsymbol{h}^{(s)}_{\text{recovered}} = \boldsymbol{h}^{(s)}_{\text{compressed}}\boldsymbol{U}_k^T + \text{PE} + \mathbf{T}_{\text{fixed}}[t_{1:n}, :] = \boldsymbol{h}^{s} \tag{73}$$

这种专为训练基于Transformer的大语言模型而设计的压缩方案，可支持在带宽低至 60 Mbps 的情况下跨四个地理区域进行训练。因此，我们会将其纳入逼真的 SWARM 集成中。有关采用子空间压缩的 SWARM 概览，请参见图 17b。

G.2.1 SENTINEL 与子空间压缩的兼容性

为展示我们提出的方法与子空间压缩的兼容性，我们将这种压缩算法集成到固定网格 TorchTitan 实现中。我们假设恶意工作节点会篡改压缩信号，因为这些信号会发送给后续工作节点。

算法 6 SWARM 训练节点（Ryabinin 等人，2023年）

```
class TrainerNode:
    def __init__(self, dht, stage_uids):
        self.dht = dht
        self.stages = [RemoteExpert(uid, dht) for uid in stage_uids]

    def forward(self, input_batch):
        """Process microbatch through all pipeline stages"""
        hidden = input_batch
        for stage in self.stages:
            hidden = stage.process(hidden)
        return hidden

class RemoteExpert:
    def __init__(self, stage_uid, dht):
        self.stage_uid = stage_uid
        self.dht = dht

    def process(self, data):
        """Route data to available worker at this stage"""
        worker = self.dht.find_available_worker(self.stage_uid)
        result = worker.forward(data)
        return result
```

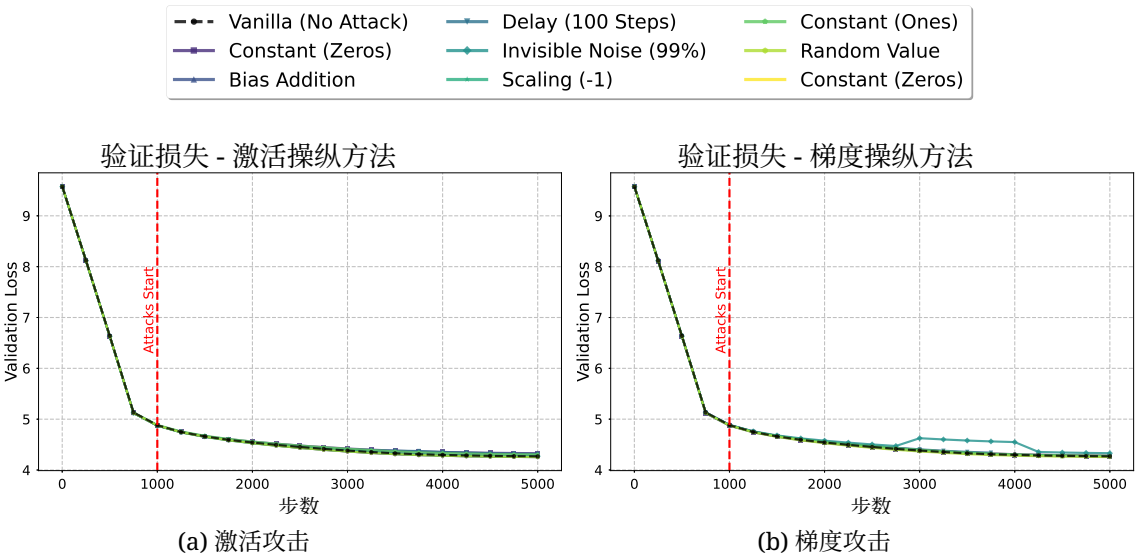
具体而言，我们使用传输信号 $\boldsymbol{h}^{(s)}$ 压缩后 $\in \mathbb{R}^{b\times n\times k+1}$ 的前 k 个维度，因为最后一个维度对应固定的词元嵌入层，而它们在不同批次之间可能存在显著差异。具体来说，SENTINEL 使用 $\boldsymbol{h}^{(s)}$ 压缩后 $[:,:,:-1]$ 进行基于指数移动平均（EMA）的激活验证（梯度验证也类似，使用 $\boldsymbol{g}^{(s)}$ 压缩后 $[:,:,:-1]$）。算法的其余部分保持不变。

基于这些假设，我们采用表中给出的设置，在 FineWeb-EDU 数据集上训练 16层 Llama-3-0.6B。13。唯一的区别是，在这些实验中，我们使用本地批次大小 8 和 8 个梯度累积步数，从而使每个优化器步的目标批量大小达到 512。对于子空间压缩，我们使用压缩因子 25，实现 96% 的压缩率。正如表中的结果所示，19，SENTINEL可以轻松适配此设置，并展现出对各种攻击的高效检测能力。我们还在图中绘制了这些攻击对应的验证损失。18，结果表明，所有情况下训练均未中断。

G.3 SENTINEL 与 SWARM 的集成

既已介绍 SWARM 的背景，以及子空间压缩如何作为一种互补因素，使受限带宽下的去中心化训练成为可能，下面我们将详细说明如何将 SENTINEL 集成到这一切实可行、可投入生产的去中心化训练生态系统中。

如前所述，训练器节点天生适合担任验证者角色。正如第A Q3 节所述，训练节点仅执行基于 CPU 的操作，且成本仅为工作节点成本的一小部分，因此很容易获得。因此，利用此类节点进行训练协调并非不切实际。此外，数据完整性在很大程度上取决于训练节点是否可信；若将这一角色委托给不受信任的参与方，就会引发其他重大问题，包括后门攻击和隐私攻击。因此，使用受信任的训练节点既经济可行，也在逻辑上至关重要。



(a) 激活攻击　　(b) 梯度攻击

图18：采用子空间压缩训练 Llama-3-0.6B 模型时的验证损失演变（Ramasinghe 等人，2025年）在各种激活/梯度操纵攻击下的情况。尽管激活/梯度经过压缩，SENTINEL 仍能成功防止相对原始训练损失发生发散。

表19：使用子空间压缩训练 Llama-3-0.6B 时，不同攻击模式下的攻击检测性能（Ramasinghe 等人，2025年）。所示度量指标包括精确率、召回率、F1分数（均为百分比）以及步数为5000时的验证损失。在所有情形下，每个阶段的恶意与诚实比例均为3:5。

模式攻击		检测性能			训练
		Pr. (%) ↑	召回率（%）↑	F1 分数（%）↑	验证损失 ↓
Activation	常数（零）	100.00	100.00	100.00	4.2490
	常数（一）	100.00	100.00	100.00	4.2486
	随机值	100.00	100.00	100.00	4.2999
	偏置加法（常数）	100.00	75.00	85.71	4.2484
	偏置加法（随机）	0.00	0.00	0.00	4.3248
	延迟（100 步）	100.00	100.00	100.00	4.2677
Gradient	常数（零）	100.00	100.00	100.00	4.2470
	常数（一）	100.00	100.00	100.00	4.2639
	随机值	100.00	100.00	100.00	4.2726
	偏置加法（常数）	80.00	100.00	88.89	4.2751
	偏置加法（随机）	100.00	100.00	100.00	4.2608
	延迟（100 步）	100.00	100.00	100.00	4.2567
	混合模式	96.97	88.89	92.75	4.2779

为了在训练器节点中加入 SENTINEL 验证，我们采用一种机制，让训练器维护其在不同阶段分发的信号的指数移动平均。具体而言，每个训练器都会存储所有层输出的指数移动平均，因为它会端到端地处理这些输出。具体来说，训练器 i 维护：

$$\{\boldsymbol{m}_{t,i}^{(s)} \mid 1 \le s \le p\} \tag{74}$$

其中

$$\boldsymbol{m}_{t,i}^{(s)}(\boldsymbol{h}) = \beta_h \boldsymbol{m}_{t-1,i}^{(s)}(\boldsymbol{h}) + (1-\beta_h)\boldsymbol{h}_{t,i}^{(s,r)}, \tag{75}$$

是步骤 t 和阶段 s 的动量。梯度指数移动平均也保留了一组类似的状态，但为简洁起见我们将其省略。利用这些指数移动平均状态，每个训练器都可以在工作节点处理信号的同时运行 SENTINEL 验证。

算法7 带有 SENTINEL 验证的 SWARM 训练节点

```
class SENTINELTrainerNode:
    def __init__(self, dht, stage_uids):
        self.dht = dht
        self.stages = [RemoteExpert(uid, dht) for uid in stage_uids]
        self.ema_detector = EMADetector()
        self.banned_workers = set()

    def forward(self, input_batch):
        """Process microbatch with Sentinel verification"""
        hidden = input_batch
        for i,  stage in enumerate(self.stages):
            # Process data at current stage
            hidden = stage.process(hidden)

            # Sentinel verification
            is_suspicious = self.ema_detector.update_and_detect(i, hidden)

            if is_suspicious:
                worker_uid = stage.get_last_worker_uid()
                self._report_violation(worker_uid)

        return hidden

    def _report_violation(self, worker_uid):
        """Report violation to DHT for global coordination"""
        violations = self.dht.get(f"violations_{worker_uid}", default=0)
        self.dht.store(f"violations_{worker_uid}", violations + 1)

        if violations + 1 > MAX_VIOLATIONS:
            self.dht.store(f"banned_{worker_uid}", True)
            self.banned_workers.add(worker_uid)
```

在 SWARM 中实现 S ENTINEL的一项挑战是如何在训练器之间传递恶意行为信息。由于每个训练器都负责维护独立的指数移动平均，并独立验证随机选取的工作节点，这可能会加剧恶意参与者造成的影响。为解决这一问题，我们利用分布式哈希表，在训练器之间轻量级地传递有关恶意工作节点的信息。训练器不再各自独立跟踪违规计数器或封禁工作节点，而是通过分布式哈希表协作，共同增加违规计数器或封禁工作节点。简而言之，我们通过工作节点在分布式哈希表中的唯一标识符（唯一标识符）跟踪其违规次数；如果违规次数超过允许值，最先发现这一情况的训练器就会封禁该工作节点。同样，对于发生暂时违规后表现良好的工作节点，我们会定期检查，并通过 SENTIN EL 中讨论的宽容策略减少其违规次数。使 S ENTINEL在 SWARM 中运行得更好的唯一细微改动，是将受污染的梯度替换为零张量；我们发现这样能让训练更加稳定。Alg. ENTINEL中给出了 S 与 SWARM 集成的伪代码，见 Alg. 7（为简洁起见，省略反向传播）。

G.3.1 随机布线下验证的重要性

在 SWARM 中实现 SENTINEL 时，我们发现一个有趣的现象：经过高度校准的检测阈值对于消除大量假阳性至关重要，这是由 PP 中的“级联效应”与 SWARM 中的“随机连线”相互作用所致。正如 Ryabinin 等人 (2023年) 所述，随机连线有助于 SWARM 提高设备利用率并减少空闲时间。然而，考虑到对抗性参与者，这种行为可能产生不利影响。具体而言，如果训练器未能检测出或标记恶意参与者，它不仅会使用损坏信号更新自身的指数移动平均，其他训练器也会面临风险，因为在通过 SWARM 的随机连线路由微批次时，它们最终也会使用该恶意工作节点。因此，如果恶意工作节点未被检测到，它就可能污染所有训练器的指数移动平均，导致它们将该阶段中的所有诚实工作节点都标记为恶意。在固定网格中（如图 19a 所示），最坏情况下，由于第 D.2 节讨论的“级联效应”，我们可能会错误地将单条流水线中的所有工作节点都标记出来。然而，在 SWARM 中，由于“随机连线”，我们可能会错误地指控同一阶段的所有工作节点；若再考虑“级联效应”，检测算法的假阳性率可能会很高。我们将对这一有趣的相互作用作进一步研究，并留待未来工作。这一现象已在图 19b 中详细说明。

G.4 详细实验结果

为评估我们将 SENTINEL 集成到 SWARM 后在包括子空间压缩在内的真实分布式训练条件下的表现，我们开展了全面实验，测试其检测能力以及系统在各种攻击场景下的鲁棒性。实验设置旨在验证所提方法在真实条件下的有效性。

G.4.1 实验设置

分布式架构。 我们使用一个包含 16 个 Transformer 层的 Llama-3-0.6B 模型，并将其划分为 16 个流水线并行阶段以用于 SWARM。每个阶段使用 8 个并行工作节点同时处理微批次，共计 128 个工作节点。为模拟真实的分布式环境，每个工作节点都部署在单独的 AWS 实例上，且与其他节点没有直接连接。我们根据计算需求使用异构实例类型：使用配备 NVIDIA A10G GPU 的 g5.2xlarge 实例运行嵌入层和第一个 Transformer 层；使用配备 NVIDIA A10G GPU 的 g5.4xlarge 实例运行最后一个 Transformer 层和投影头；使用配备 NVIDIA T4 GPU 的 g4dn.2xlarge 实例运行所有中间 Transformer 层。

训练器基础设施由分布在 4 台 `c6a.8xlarge` CPU 实例上的 32 个并行训练器组成，每个实例同时运行 8 个训练器。每个训练器处理从 FineWeb-EDU（Penedo 等人，2024年）经由 HuggingFace（Lhoest 等人，2021年）流式传输而来的互不重叠的数据分片，确保训练期间数据没有重叠。此配置使我们能够在真实的部署条件下测试基于 DHT 的违规报告系统的可扩展性。

攻击场景。 我们在两种不同的场景下评估 SENTINEL 的检测性能s:

3. **仅梯度攻击**：由常量攻击（全零和一）、偏置添加、随机值注入和缩放攻击组成的异构混合攻击。4. **梯度与激活组合攻击**：扩展的攻击套件，包含梯度攻击与激活攻击的混合，例如常量攻击（全零和一）、偏置添加、随机值、缩放、延迟和随机符号攻击。这种更复杂的威胁模型评估了 SENTINEL 在整个前向与反向传播过程中的综合检测能力。

对于这两种场景，我们模拟了一个具有挑战性的环境，其中恶意工作者与诚实工作者比例为 3:5

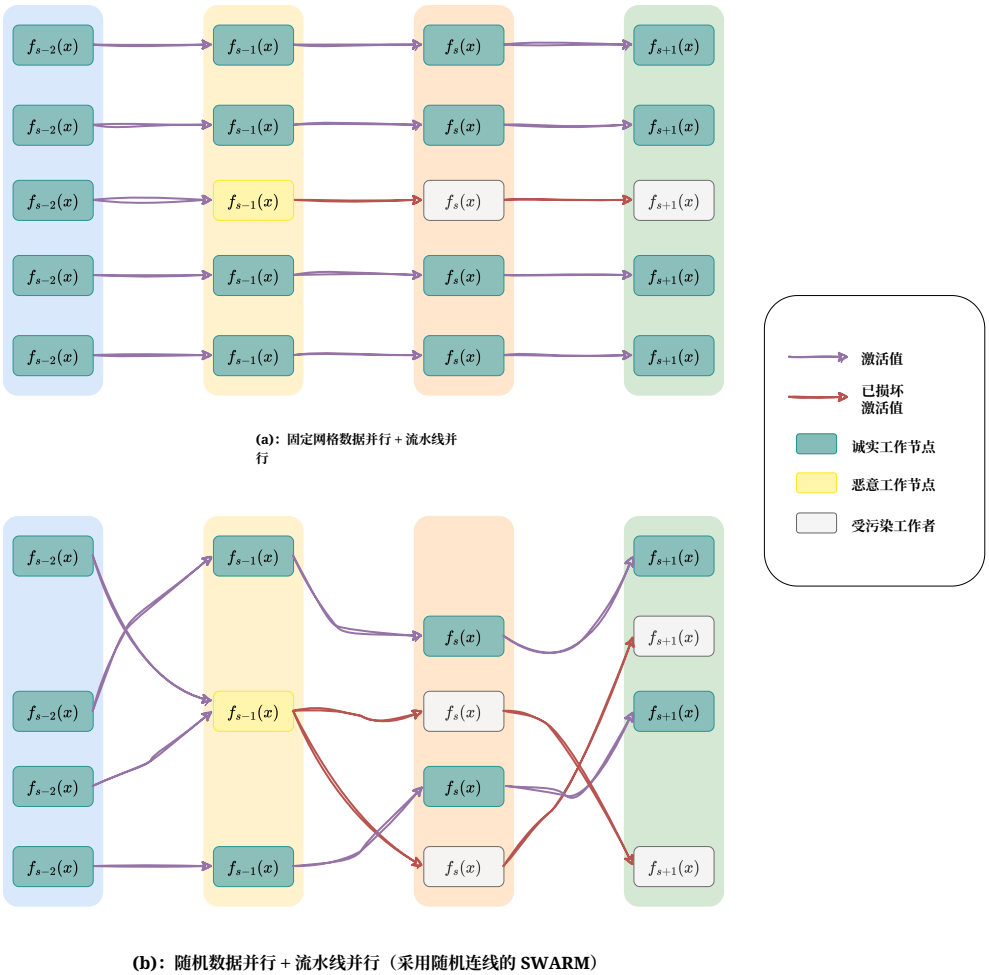


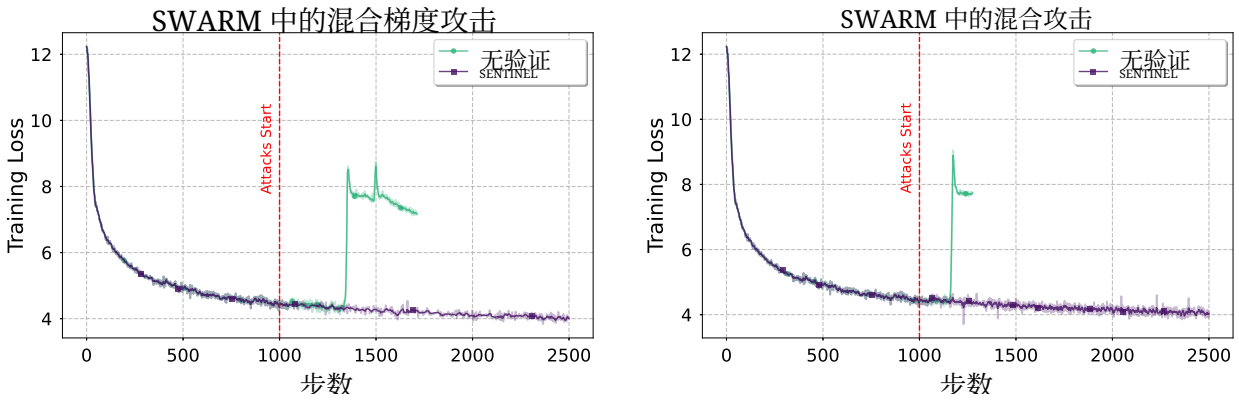
图 19：(a) 固定网格数据并行 + 流水线并行与 (b) SWARM 的视觉对比（Ryabinin 等人，2023年）。与固定网格数据并行 + 流水线并行中每个工作节点都会将数据发送给下一阶段的特定工作节点不同，SWARM 使用随机连线在各阶段之间路由数据批次。这样可以确保工作节点达到最大利用率，并将空闲时间降至最低。然而，这种行为也有双刃剑效应：通过在工作节点之间随机路由数据批次，我们实际上扩大了恶意工作节点的影响范围；如果未能及时发现，它们就可能污染后续阶段的“任意”工作节点。这凸显了在现实环境中采用高度精准的验证机制的重要性。请注意，在 SWARM 中，工作节点之间传输的信号都会经过训练器节点，因此工作节点之间不存在点对点通信。为了更直观地与固定网格进行对比，我们直接连接了这些工作节点。

（37.5% 的恶意工作节点）。在仅梯度设置中，我们假设攻击者各自独立行动，互不协同，并随机发起攻击。在组合攻击场景中，我们模拟了部分协同：15% 的恶意工作节点会随机发起同步攻击，以模拟协同的对抗行为，同时对攻击者的能力作出符合现实的假设。

训练配置。 我们为每个工作节点使用大小为 4 的微批次，每个优化器步的总目标批量大小为 512。对于子空间压缩，我们使用的压缩因子为 25。所有其他超参数和训练设置均与表 13 中给出的相同。鉴于运行 128 个 SWARM 工作节点的成本，我们选择仅训练 2500 步。尽管如此，我们确保将所有攻击开始时间都纳入这些步数之内。

表 20：SENTINEL 在包含 128 个工作节点的分布式 SWARM 中的检测性能。37.5% 的工作节点是恶意工作节点，它们提交随机选取的混合梯度与激活攻击。请注意，召回率较低是因为一些节点采用了不会干扰训练的弱攻击，因此不会被标记为恶意节点，训练也会继续且不受干扰。这与我们在图 1 中的观察以及定理 3.1的直觉一致：弱攻击者可能逃过检测，但不会对训练造成危害。

攻击模式	度量指标		
	PR. (%) ↑	RE. (%) ↑	F1 分数 (%) ↑
混合梯度	100.0	75.0	85.7
混合激活/梯度	100.0	66.7	80.0



(a) 混合梯度攻击　　(b) 混合梯度/激活攻击

图 20：在由 128 个工作节点组成的分布式 SWARM（Ryabinin 等人，2023年）中，使用子空间压缩（Ramasinghe 等人，2025年）训练 Llama-3-0.6B 模型时的损失。工作节点采用各种激活/梯度操纵攻击来破坏训练。没有验证训练时，训练会受到干扰；而 SENTINEL 能够通过检测并封禁恶意工作节点，成功保护训练免于发散。

G.4.2 结果

在本节中，我们展示我们的结果。从图20可以看出，在不可信分布式环境中进行训练时，若缺乏可行的验证机制，训练很容易被破坏。在表 20中，我们还展示了 SENTINEL 的检测性能，其中我们取得了高于 80% 的 F1 分数。需要注意的是，混合激活/梯度攻击中较低的召回率是由于某些节点采用了不会干扰训练的弱攻击，因此它们不会被标记为恶意，但训练仍能不受干扰地继续。这与我们在图 1 中的观察以及定理3.1 所给出的直觉一致，即弱攻击者可能逃过检测，但不会损害训练。这可以从图 20 中看出，两种情况下训练都在没有发散的情况下继续。这些结果证明了 SENTINEL 在涉及分布式训练的现实应用中的通用性。

训练器之间的EMA方差。 在采用 SENTINEL的 SWARM 中，我们讨论过每个训练器如何累积各自的指数移动平均值。因此，这些信号在不同训练器之间可能有所差异。如果方差较大，可能会导致验证中断：某个工作节点在一个训练器看来是诚实的，却可能仅仅因为训练器之间的 EMA 方差而被标记为恶意。为了证明训练器之间的指数移动平均值发散程度很小，我们绘制了 SWARM 训练所用的全部 32 个训练器中，激活和激活梯度 EMA 的标准差。如图 21所示，激活和激活梯度在不同训练器之间的方差都处于可控范围，这表明不同训练器中的指数移动平均值以相似的方式演变。

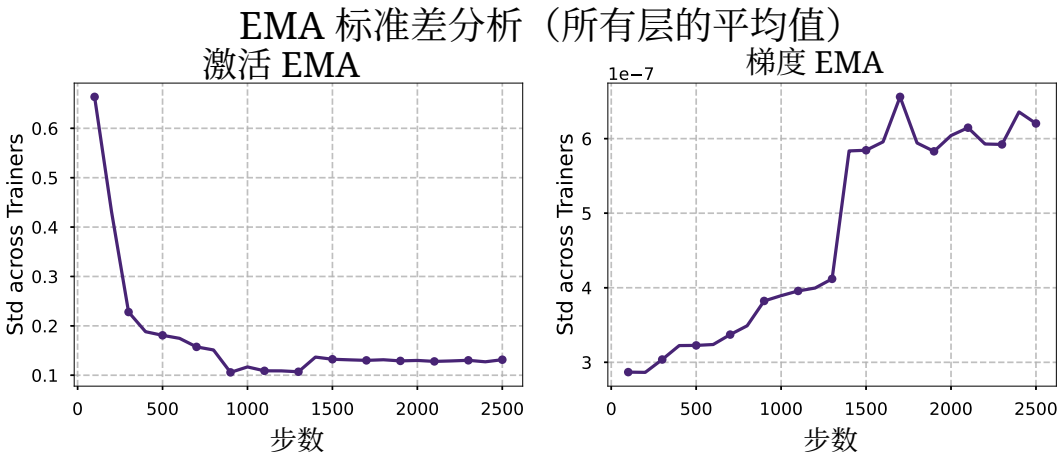


图 21：用于训练我们的 Llama-3-0.6B 模型的 32 个训练器之间 EMA 信号（激活/梯度）的标准差。由于每个训练器都保留了所有层的指数移动平均值，因此此处报告的是所有层的平均值。如图所示，随着训练的进行，激活 EMA 方差逐渐减小，而训练器之间的梯度 EMA 方差逐渐增大。尽管方差有所增加，但请注意，梯度 EMA 方差的量级非常接近于零（10^{-7}）。

检验统计量的演变。 最后，我们考察 S ENTINEL 如何被 SWARM 中相互独立的训练器节点使用。为此，我们跟踪某个工作节点（第12层的工作节点 7）处理来自不同训练器的微批次时的检验统计量。我们还跟踪使用算法 5 中的自适应 IQR 机制确定的下限和上限阈值。图22 展示了这些检验统计量随时间的变化。从图中可以看出，尽管训练器节点之间没有直接通信 EMA 或阈值，全部 32 个训练器仍呈现出相似的演变模式。这种一致性至关重要，因为每个训练器都会将其微批次分配给不同的工作节点，因此必须保持检测标准一致。若缺乏这种一致性，系统可能出现假阳性或假阴性，进而破坏训练稳定性。在 S EN TINEL 中使用 EMA 的主要优势在于，它能够高效追踪历史模式，从而使分布式训练器之间实现这种协同行为。

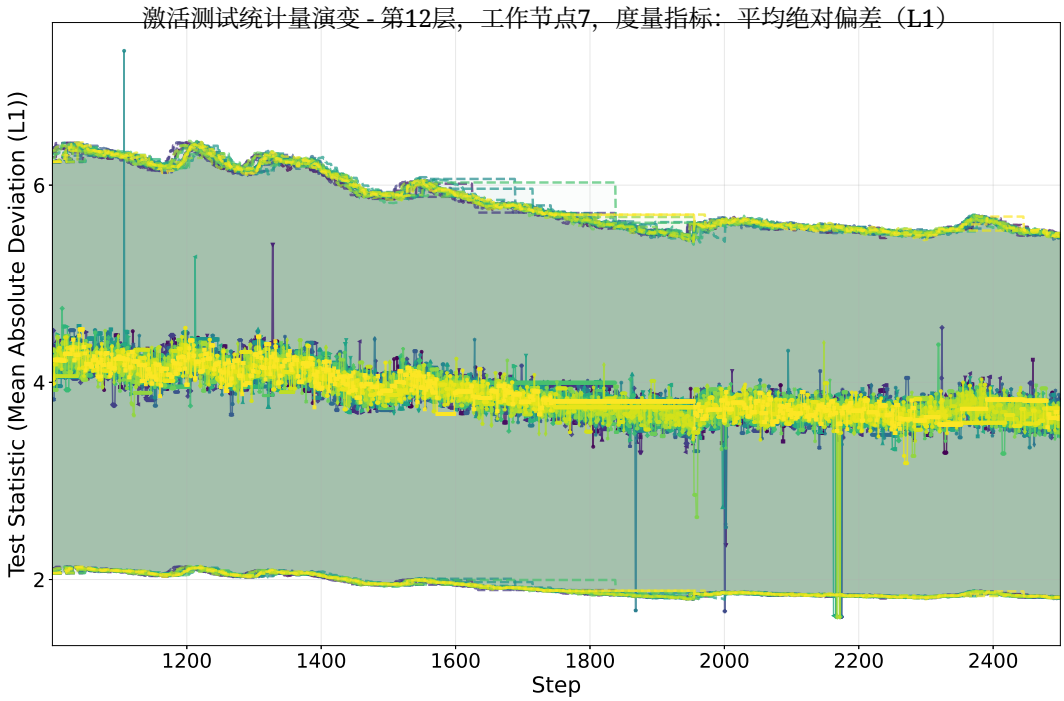


图22：在我们的 16层 Llama-3-0.6B 中，处理第12层的工作节点的自适应偏差界和工作节点统计信息的演变。每种颜色代表不同训练器记录的检验统计量。每个训练器的上、下阈值由自适应 IQR 机制确定，并分别以顶部和底部的线条表示。尽管训练器之间没有直接通信指数移动平均值或阈值，但它们的阈值通常相近。该工作节点在约第 2150 步开始攻击，之后训练器会标记并封禁该工作节点。