Exploring the Robustness of Decentralized Training for Large Language Models

Lin Lu*

Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science of Technology Wuhan, China loserlulin@hust.edu.cn

Binhang Yuan Hong Kong University of Science of Technology Hong Kong, China biyuan@ust.hk

Chenxi Dai*

Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science of Technology Wuhan, China dcx001@hust.edu.cn

Yanan Sun

College of Computer Science, Sichuan University Sichuan, China ysun@scu.edu.cn Wangcheng Tao
School of Cyber Science and
Engineering, Huazhong University of
Science of Technology
Wuhan, China
dangmai@hust.edu.cn

Pan Zhou

Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science of Technology Wuhan, China panzhou@hust.edu.cn

ABSTRACT

Decentralized training of large language models has emerged as an effective way to democratize this technology. However, the potential threats associated with this approach have not been carefully discussed, which would hinder the development of decentralized training infrastructures. This paper aims to initiate discussion towards this end by exploring the robustness of decentralized training from three main perspectives. First, we demonstrate the vulnerabilities inherent in decentralized training frameworks in terms of hardware, data, and models. Second, we highlight the fundamental difference between decentralized foundation model training and vanilla federated learning, where the security techniques employed in federated learning cannot be applied directly. Third, we discuss the essential components required for a robust and efficient decentralized training framework and present a case study by modeling a concrete threat model. Our objective in this vision paper is to emphasize the importance of addressing security concerns in the context of decentralized training for large language models.

1 INTRODUCTION

Large language models (LLMs) [12, 63, 66, 72] have shown exceptional accuracy in numerous natural language processing tasks, thus gaining widespread acceptance and usage [15, 55, 58]. However, to improve accuracy in various domains, LLMs have expanded aggressively in terms of model scale and pre-train data volumes, resulting in time- and cost-intensive training processes [8, 31, 74]. For example, the state-of-the-art Falcon-180B [29] model has 180 billion parameters trained on 3.5 trillion tokens. Given the intensive computational load, sophisticated parallel strategies must be leveraged to speed up and scale out the training procedure [28, 38, 44, 45, 49].

A promising direction to democratize the training of large language models is through decentralized training [17, 52, 69], which presents a substantial solution to alleviate this resource-intensive challenge. On the other hand, these decentralized training frameworks are primarily based on *model parallelism* (e.g., *pipeline parallelism* [28, 44]), supplemented by data parallelism. These parallel paradigms require communication of *activations* during forward propagation and *corresponding gradients* during backward propagation, which is fundamentally different from vanilla federated learning (FL) that only requires synchronization of *model gradients* in a data parallel paradigm. As a result, the potential risks and vulnerabilities associated with such decentralized training have not been formally discussed, to the best of our knowledge.

The most relevant technique discussed in the data management and machine learning communities is secure aggregation in FL, which limits its scope under the data parallel communication paradigm [18, 19, 34, 60]. In such scenarios, when malicious gradient values arise, the parameter server employs resilient gradient aggregation methods. These methods mainly employ outlier detection algorithms, such as the voting mechanism and the bucketing mechanism, to mitigate the impact of these malicious gradient values on the global model. On the other hand, safety issues under the scope of model parallelism are mostly unexplored, where the communication of activations and the corresponding gradients demands different approaches for malicious detection and defense.

Therefore, in this paper, we initiate the discussion of three fundamental questions about the robustness of decentralized training, particularly in the context of pipeline parallelism. For each question, we give our answer and make a detailed explanation:

Q1: What types of threat may occur in decentralized training? How will they influence the statistical efficiency of the training? To assess the vulnerability and sensitivity of decentralized training, we have analyzed three potential threats: hardware failures, privacy inference attacks, and poisoning attacks. These threats represent three kinds of malicious attackers with escalating attack capabilities. We have investigated the feasibility of these attack forms under the scenario of decentralized training and emphasized their potential consequences.

- Q2: Can the existing defense methods in traditional distributed learning (i.e., FL) be applied to decentralized training based on pipeline parallelism directly? The short answer is No We enumerate two primary distinctions between decentralized training and FL, which underscore the structural differences between pipeline parallelism and data parallelism techniques. These distinctions lead to notable discrepancies in the threats encountered with the decentralized training framework compared to previous security challenges of distributed systems. Consequently, defense algorithms tailored for traditional data parallelism techniques cannot be directly employed.
- Q3: From what perspectives can we enhance the robustness
 of the decentralized training framework? Based on the potential threats mentioned above, our study focuses on determining
 the fundamental components necessary for a robust and efficient
 decentralized training framework. We also explain the organizational architecture of these components to effectively mitigate
 the aforementioned threats.

We also present a case study illustrating a straightforward and potent poisoning attack method targeting the forward and backward data propagation processes within a decentralized training framework. This case study emphasizes the urgency of addressing this issue as such poisoning attacks can profoundly impede model convergence and compromise model performance. To encounter this attack, we propose a relatively robust and efficient training framework. Additionally, we validate the effectiveness of both our attack and defense strategies through experimental verification.

The primary objective of this paper is to examine the potential threats inherent in decentralized training frameworks and propose possible defense methods to tackle these challenges. We hope that the discussion presented in this study will garner substantial attention from researchers specializing in related fields.

2 BACKGROUND

Parallel training for LLMs. To distribute the training computation of large language models over thousands of compute devices (usually GPUs), different categories of parallel strategies have been proposed. Data parallelism partitions the mini-batch by training samples to distribute the computation load, where each GPU holds a local model replica for forward and backward propagations and communicates the gradients for synchronization, usually by a parameter server or an AllReduce operation [38]. FL [9, 35, 40] is mainly based on data parallelism. Figure 1(a) illustrates an example of data parallelism with 4 workers. They send gradients computed from their local datasets to the parameter server and receive aggregated gradients to update their local models [51, 71]. Pipeline parallelism partitions the training computation into multiple stages as a pipeline, where each GPU handles one stage. Figure 1(b) provides an illustration of pipeline parallelism, in which the model is partitioned into distinct sub-models, and each computational device handles a specific subset of model layers [28, 44, 68]. In contrast to data parallelism, pipeline parallelism requires fewer communication exchanges and optimizes the utilization of computational resources [6, 50, 56]. Due to these advantages, pipeline parallelism has become the main technique for decentralized training.

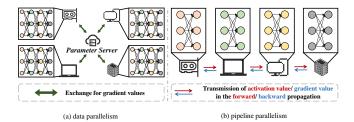


Figure 1: A comparison of model layer segmentation in data parallelism and pipeline parallelism.

Decentralized training. Decentralized training strategies have emerged as practical means to facilitate collaborative training of LLMs among multiple contributors, thereby enhancing the democratization of the training process. [69] initially investigates the decentralized training for large foundation models using model parallelism. Subsequently, [53] and [65] accomplish billion-scale training on heterogeneous devices with slow interconnect. Similarly, [59] aims to leverage vast untapped consumer-level GPUs. Robustness of the decentralized training. While the security problems in decentralized training have been mentioned in previous works [11, 59], no systematic research has studied this issue extensively. Existing research focuses mainly on ensuring seamless pipeline operations [5, 30, 61]. However, these discussions face a prevalent limitation. They only discuss machine failures, neglecting the vulnerability of decentralized training to various imperceptible security risks.

3 POTENTIAL THREATS

This section aims to address Q1 by discussing the potential threats in decentralized training, including hardware failures, privacy inference attacks, and poisoning attacks. We observe that these three threat forms represent attackers with increasing capabilities. The weakest attacker can cause hardware failures without access to the training datasets or the transmitting values. A stronger attacker can steal the data during training, while the strongest attacker can manipulate the transmitting activation values or gradient values.

3.1 Hardware Failures

Hardware failures are common in distributed learning systems. For instance, Meta AI experienced numerous hardware failures while training OPT [72], resulting in over one hundred restarts in their compute cluster. In the event of a hardware failure, decentralized training keeps all training resources inactive until repairs are made, leading to significant resource wastage.

This matter has garnered significant attention as a problem of fault tolerance and frequent interruptions. Relevant research focuses on maintaining model throughput while enabling automatic recovery from hardware failures. For instance, Varuna [5] introduces job morphing, allowing for the reconfiguration of the training job. Similarly, Bamboo [61] and Oobleck [30] facilitate the use of backup computing resources in case of hardware failures, ensuring a seamless training process.

However, several challenges remain unsolved. For example, the aforementioned approaches experience additional overhead when restarting the training process or require additional computing resources, which leads to the inefficient utilization of computing resources and contradicts the original objective of decentralized training strategies. A more serious problem is how to ensure the normal operation of the pipeline in the face of large-scale and high-frequency hardware failures.

3.2 Privacy Inference Attacks

LLMs have found widespread application in various domains, such as healthcare [57, 67] and law [15, 27]. Data privacy concerns in these domains make the fine-tuning process of LLMs vulnerable to privacy inference attacks. Decentralized training frameworks are particularly susceptible to privacy inference attacks due to the frequent exchange of data and the inherent openness of distributed environments. For instance, gradient values enable an adversary to obtain training inputs with only a few iterations, as highlighted in [2, 76]. [73] introduces an approach that achieves 100% accuracy in extracting ground-truth labels from the gradients.

Furthermore, in addition to directly accessing the original data, there are studies [4, 23] that focus on properties unrelated to the characteristic features of the class. These studies show that an attacker, armed with auxiliary training data labeled with the desired property, can deduce valuable information that was previously unknown. Whether through direct or indirect means, privacy inference attacks pose a risk in decentralized training frameworks, potentially exposing sensitive content in the training datasets.

3.3 Poisoning Attacks

In contrast to the act of stealing information in privacy inference attacks, poisoning attacks enable attackers to manipulate data transmission between stages. Depending on the attacker's objectives, poisoning attacks can be categorized as targeted attacks or untargeted attacks. Targeted attacks hinder the model's convergence by freely manipulating transmitting values, whereas untargeted attacks aim to inject backdoors into the global model.

Previous studies [13, 62] thoroughly investigate the detrimental impact of untargeted attacks on the convergence of the global model in distributed systems. However, these studies were either limited to FL scenarios or only involved poisoning datasets by tampering with the corresponding labels. We evaluate the vulnerability of decentralized training to untargeted attacks in Section 4, providing an explanation for why decentralized training frameworks are more susceptible to such attacks compared to FL.

In the case of targeted attacks, a significant distinction arises from the inherent assumption of absolute security regarding the data providers in decentralized training. Nevertheless, several studies [24, 36] demonstrate the feasibility of implanting backdoors without access to the original data. Since the decentralized training framework involves frequent transfer and update of gradients, these attacks can be applied to decentralized learning as well. The frequent data exchange of decentralized training provides a new form of poisoning attacks, that is tampering with the activation values or gradient values. We show the possible consequences of this new untargeted poisoning attack form in our case study.

4 LIMITATION OF SECURE AGGREGATE IN FL

In this section, we aim to address **Q2**. We posit that the direct application of current security methods in FL to decentralized training encounters significant challenges for the following reasons.

4.1 Inherent Serial Characteristic

Decentralized training frameworks primarily rely on pipeline parallelism as the main training technique. However, due to limited computational resources, a majority of training initiators only deploy one pipeline. This constraint results in an inherent serial characteristic within decentralized training frameworks, impeding the direct application of existing methods in two critical aspects.

Lack of comparable values. In traditional FL, each worker possesses a complete copy of the global model. Privacy-preserving techniques, such as secure multiparty computation [10] or secret-sharing-based methods [10], are used to prevent privacy inference attacks. To mitigate poisoning attacks, outlier detection algorithms, like the voting mechanism [16, 42, 64] and bucketing mechanism [1, 33, 75] can be employed to filter the Byzantine workers.

However, during decentralized training, each stage in the pipeline can solely receive activation values or gradient values from the preceding stage. Due to the lack of comparable values, directly applying outlier detection algorithms or other privacy-preserving methods is not feasible. Although some training initiators try to solve this problem by adding more pipelines [30, 37, 45], striking a balance between computing resource utilization and obtaining an adequate number of comparable values is challenging.

Heavy dependence on the predecessor stage. Each stage in decentralized training relies exclusively on the preceding stage due to the absence of a central server. In the context of poisoning attacks, if a stage becomes malicious, the remaining stages will remain unaware and mistakenly treat the malicious stage as honest. Furthermore, once a malicious stage manipulates the transmitting values, the subsequent stage cannot detect this malicious behavior and can only propagate the tampered data.

To illustrate, we consider the scenario where a malicious stage transmits an all-zero vector to the next stage. The honest stage is unable to determine if the value has been maliciously tampered with by the available algorithm and must rely on the preceding stage. In Subsection 3.3, we extensively discuss the dangers associated with poisoning attacks. However, in real training scenarios, such malicious alterations to the transmitting values will be considerably less apparent, but the resulting harm can still be substantial.

4.2 Change of Exchange Object and Frequency

Compared to data exchange between the parameter server and workers, the exchange objects and frequency have changed a lot in decentralized training. In terms of the exchange object, stages should additionally transmit activation values in the forward propagation. Compared to gradient values, activation values vary more with the training data. As a result, the average-value-based resilient aggregation method cannot ensure the accuracy of training.

On the other hand, the parameter server only exchanges with the workers once during each iteration. However, the number of data exchanges in the decentralized training relies on the number of stages. The unknown target of the attacker requires a robust algorithm in every data exchange, which undoubtedly extends the training time and greatly reduces the training efficiency.

5 ROBUST DECENTRALIZED TRAINING

In this section, our attention is centered on Q3. Here, we delineate vital components necessary for a robust and efficient decentralized training framework. In addition, we analyze the associated challenges. From a theoretical perspective, we discuss the viability of existing defense algorithms in mitigating privacy inference attacks and poisoning attacks. Furthermore, we underscore the imperative of fast recovery from a systematic viewpoint.

5.1 Privacy Preservation

Privacy-preserving methods, particularly in FL, have gained significant traction in diverse areas of machine learning. Existing research on privacy preservation can be categorized into two main approaches: encryption-based and perturbation-based methods.

Encryption-based methods encompass homomorphic encryption [3, 70], secret sharing [54], and secure multiparty computation methods [43]. These approaches focus on safeguarding data privacy during transmission and preventing unauthorized access to the original data by employing encryption and decryption in each data exchange process. However, the frequent encryption and decryption operations reduce the decentralized training efficiency greatly. Although homomorphic encryption allows computation on ciphertext and retrieval of the computed plaintext with a single decryption operation, it imposes stringent requirements on the calculation method and the time it occupies cannot be overlooked.

Perturbation-based methods, such as differential privacy [20, 21, 41] and additive perturbation [14, 25, 39] are utilized in studies to prevent attackers from inferring data privacy. These methods involve adding noise directly to gradient values or training datasets. Although these methods are straightforward and require minimal additional training time, weak noise can be easily mitigated by noise reduction algorithms [32], while strong noise significantly reduces the training efficiency of the global model.

In summary, both types of privacy-preserving algorithms face a specific challenge when implemented in decentralized training frameworks: how to control the decline of training accuracy within an acceptable range while ensuring the efficiency of encryption. Further investigation is needed in future studies to determine the appropriate perspective to adopt in specialized training scenarios.

5.2 Stage-Level Malicious Behaviors Detection

As stated in Section 3, attackers engaging in poisoning attacks and privacy inference attacks demonstrate distinct motivations, capabilities, and malicious behaviors, thereby resulting in substantial divergences in the security algorithms applied to these scenarios. Prior studies have elucidated the practicality of defense mechanisms against targeted attacks, such as eliminating backdoors from trained models. However, this strategy proves inadequately effective against untargeted attacks. Nonetheless, it is evident that both poisoning attacks pursue a shared goal: tampering with activation values or gradient values. Consequently, conventional iteration-level defense methods, for instance, resilient aggregation techniques tackling Byzantine problems in FL, cannot be directly utilized in

decentralized training frameworks. Therefore, a direct and efficient defense approach involves the implementation of a detection algorithm to identify malicious behaviors at the stage level.

Regrettably, this issue has not received adequate attention in the existing literature. To address this problem, we propose employing redundant computation to detect any malicious tampering between stages. In Section 6, we present a comprehensive case study to illustrate the effectiveness of this detection methodology. Despite the additional GPU storage space requirements and the resulting decrease in training efficiency, our approach's robust defense capability convincingly validates its potential for future research.

5.3 Fast Recovery from Failures

In the event of a hardware failure or a detected poisoning attack, it is crucial for the pipeline to recover promptly. A straightforward method is to restart this training iteration every time encountering malicious behaviors. Despite its feasibility, this restart method wastes the results obtained in the current training iteration and leads to prolonged idle time, as subsequent computing resources remain underutilized for an extended period. Ensuring the continuous operation of the pipeline and quickly recovering the original data are essential considerations for a robust decentralized training framework. In Section 6, we present alternative solutions to minimize computing resource consumption while achieving swift recovery from failures or attacks.

6 A CASE STUDY

We present a case study to examine the vulnerability of decentralized training and introduce our robust training framework. We substantiate our findings with experimental evidence that demonstrates the potential of this threat to disrupt model convergence. Furthermore, we demonstrate the effectiveness of our robust training framework in mitigating this risk. For convenience, we suggest a decentralized training framework consisting of K stages. Additionally, M_i represents the sub-layer of the i-th stage.

6.1 Threat Model and Attack Methods

We assume an attacker, denoted as \mathcal{A} , who can randomly manipulate a stage, including both forward and backward propagation, during each iteration with a predetermined attack rate. If \mathcal{A} successfully gains control of a stage, this particular stage transmits the malicious value \mathbf{a}'_{out} to the subsequent stage, instead of the intended output \mathbf{a}_{out} , upon receiving a value \mathbf{a}_{in} . The malicious behavior during the model training process can be categorized as either a forward attack or a backward attack, as demonstrated in Figure 2 by orange and blue arrows, respectively. Notably, the initial and the final stages, which provide data and the corresponding labels, remain immune to attacks.

We employ two straightforward untargeted poisoning attack methods to simulate the actions of \mathcal{A} . In *forward attack*, the malicious stage simply flips the sign of \mathbf{a}_{out} resulting in $\mathbf{a}'_{\text{out}} = -\mathbf{a}_{\text{out}}$. In *backward attack*, the malicious stage generates a Gaussian random variable $\phi \sim N(0,1)$ with the shape as \mathbf{a}_{out} and sets $\mathbf{a}'_{\text{out}} = \phi$. Then the malicious stage sends \mathbf{a}'_{out} to the next stage.

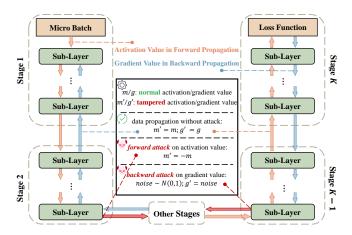


Figure 2: Illustration of the threat model in decentralized training with *K* stages, depicting *forward attack* and *backward attack*. The orange arrows represent the transmission process of activation values during the forward propagation while the blue arrows represent the transmission process of gradient values during the backward propagation. The red arrows indicate the location malicious behaviors occur.

6.2 Our Robust Training Framework

Our robust training framework consists of two main components: attack detection and efficient training, depicted in Figure 3. If our robust training framework does not detect malicious attacks, the training process continues as usual. However, in cases where malicious attacks are detected, the pipeline adopts the efficient training component to eliminate the bad consequences.

Detection strategy. Naturally drawing the inspiration of redundant computation [7, 47] and Bamboo [61], we propose the *duplicated block*. Taking the *i*-th stage as an example, it consists of redundant layers M'_{i-1} , duplicated from the (i-1)-th stage, and the original layers M_i of the *i*-th stage.

During the forward propagation, the i-th stage sends its input $\mathbf{a}_{\text{out}}^{(i-1)}$ as $\mathbf{a}_{\text{dup}}^{(i)}$ and its output $\mathbf{a}_{\text{out}}^{(i)}$ to the next stage. Upon receiving data from the previous stage, the i-th stage first uses M'_{i-1} to verify the compatibility between $\mathbf{a}_{\text{dup}}^{(i-1)}$ and $\mathbf{a}_{\text{out}}^{(i-1)}$. Only after this verification, the subsequent training process is performed. Once mismatched, the i-th stage triggers an alert and notifies the training initiator. Then the training initiator could take measures such as restarting this iteration and reusing the data sample.

To defend a more knowledgeable attacker and ensure the consistency of $\mathbf{a}_{\text{dup}}^{(i-1)}$ and $\mathbf{a}_{\text{out}}^{(i-2)}$, we introduce the *jumping connection*. During each iteration, in addition to the aforementioned operations, the *i*-th stage transmits its output $\mathbf{a}_{\text{out}}^{(i)}$ to the (i+2)-th stage and receives $\mathbf{a}_{\text{out}}^{(i-2)}$ from the (i-2)-th stage. This verification invalidates a more stealthy attack that leverages an arbitrary input $\mathbf{a}_{\text{in}}^{\prime}$ and sends the corresponding $\mathbf{a}_{\text{out}}^{\prime}$ to the next stage. The verification and transmission process in the backward propagation mirrors the forward propagation but with the reversed data transmission direction. Throughout the entire training process, the parameters of M_{i-1} and M_{i-1}^{\prime} remain identical.

Efficient training. If the *i*-th stage raises an alert, there will be a malicious stage among the (i-2)-th, (i-1)-th, and *i*-th stage. To narrow down the scope of suspicion, we introduce the *central server*, which is immune to attacks, like the initial and final stages. Direct data transmission is no longer used across the stages. Instead, as demonstrated in Figure 3(b), the *i*-th stage forwards output $\mathbf{a}_{\text{out}}^{(i)}$ to the central server. The central server subsequently sends the data pair $[\mathbf{a}_{\text{out}}^{(i-1)}, \mathbf{a}_{\text{out}}^{(i)}]$ to the (i+1)-th stage. All the verification and transmission behaviors inside the *duplicated block* remain the same. Consequently, if the (i+1)-th stage raises an alert, the malicious stage is either the *i*-th or the (i+1)-th stage.

Inspired by stochastic depth [22, 26, 46], we propose the *skip layer* method to avoid restarting the training iteration when encountering attacks. Specifically, if the (i+1)-th stage raises an alert in the forward propagation, the central server will bypass the i-th and (i+1)-th stage, transmitting data directly between the (i-1)-th and (i+2)-th stage. To ensure consistency of parameters between the original and redundant layers, we keep M_{i-1} and M'_{i+1} unchanged while updating model parameters.

6.3 Experiments

Experimental setup. We fine-tune multiple LLMs, including GPT-2 [48], Bloom [66], and Opt [72], with different parameter sizes ranging from 345M to 7B. All the model checkpoints can be downloaded from HuggingFace. We employ text-generation tasks on wikitext2, arxiv abstracts, and openwebtext datasets to conduct our evaluations. Our primary metric for assessing model performance is perplexity and GPipe [28] is used for our experiments as the base framework. To simulate the heterogeneous computing resources in real scenarios, the model is partitioned into six different computing resources, including A40, V100, RTX 3090, and Quadro RTX 5000. We utilize the clean model as the baseline to evaluate the vulnerability of the decentralized pipeline parallel training, and the attacked model to evaluate our robust training framework. We set the learning rate to 5e-6 during training, and the batch size and micro-batch size to 4 and 1, respectively. In order to maintain consistency between the duplicated model and the main model, dropout is not employed in any of our experiments.

Vulnerability of decentralized pipeline parallel training. We first assess the vulnerability of different LLMs to forward attack and backward attack. Their influences on training accuracy are presented in Table 1. We denote the ratio of the attacked training iterations to the total number of training iterations as the attack rate. It is observed that the two attack methods yield excellent results when the attacking rate is set to 0.7. After sufficient training iterations without applying any defense measures, the perplexity of the model under backward attacks increases by at least sevenfold in comparison to that of the clean model, often extending to several tens of times. However, when the attack rate is set to 0.3, the attack's effectiveness is not consistently as good. We analyze that this discrepancy may arise from the lower probability of malicious behaviors, offset by the higher probability of normal values, which allows the model to converge with adequate training iterations. The absence of the dropout, leading to a severe overfitting phenomenon, could be a significant factor contributing to the subpar performance of the attacked model at an attack rate of 0.3.

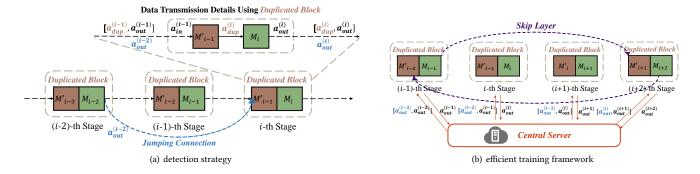


Figure 3: Details about data transmission and structure of our proposed detection strategy and efficient training framework. For the detection strategy, the brown squares and green squares represent the duplicated layers and the original layers, respectively. And they together denote the duplicated block. Blue arrows represent the jumping connection which is designed to detect a more knowledgeable attacker. For the efficient training framework, the red arrows represent the updated data transmission while the purple arrow represents the data flow using skip layer.

Table 1: Vulnerability of pipeline parallelism in decentralized training of LLMs on three datasets and two attack rates.

attack methods & attack rates→		clean	forward attack		backward attack	
LLM & datasets↓			0.3	0.7	0.3	0.7
Opt-350M	wikitext	29.77	24.82	52.37	27.73	2128.31
	arxiv	22.61	20.90	1383.81	56.14	1384.22
	openwebtext	41.38	38.30	3578.41	355.31	3584.42
GPT2-1.5B	wikitext	40.05	56.43	2503.65	25.454	788.4
	arxiv	35.34	28.89	843.38	23.42	275.4
	openwebtext	53.41	988.80	3226.01	104.87	2064.69

Table 2: Effectiveness of our robust training framework compared to clean and attacked model without any defense.

datasets & modes→	arxiv			openwebtext		
models↓	clean	attack	ours	clean	attack	ours
Opt-350M	22.61	601.92	19.31	41.38	3563.56	34.85
Bloom-560M	67.15	1682.94	22.54	122.25	3984.84	61.65
GPT2-1.5B	35.34	185.11	19.12	53.41	2435.17	36.56
Bloom-7B	59.06	818.43	27.91	102.94	3077.24	52.62

Effectiveness of robust training framework. We demonstrate the effectiveness of our robust training framework when employing the *forward attack* with the attack rate set to 0.5 in Table 2. We denote the perplexity on the clean model, the attacked model without any defense, and the attacked model under our robust training framework as clean, attack, and ours, respectively.

We observe that the perplexity of our model can improve up to 102.2 times compared to the perplexity of the attacked model when using the robust training framework. What's more, models using this framework even exhibit lower perplexity than the original models. Even when assessing the perplexity of Bloom-560M on arxiv, we note that the models employing our robust framework have only one-third perplexity of the original model. We speculate that this finding is consistent with the anomalous results presented in Table 1. Remarkably, in the absence of employing the dropout parameter, the skip layer acts as a highly effective regularization technique, mitigating the overfitting phenomenon of models.

7 CONCLUSION

This paper primarily explores the robustness of decentralized training frameworks utilizing pipeline parallelism for training LLMs. Initially, we identify and classify the potential threats, including hardware failures, privacy inference attacks, and poisoning attacks, based on the attackers' objectives and capabilities. We then compare the structural differences between decentralized pipeline parallel training and FL. Additionally, we analyze the inherent reasons why existing security methods cannot be directly applied to the decentralized training frameworks. Following this, we propose a vision for a secure and robust framework for decentralized training. Lastly, we illustrate the vulnerability of the decentralized pipeline parallel training framework through a concrete case study and introduce an attack detection method, as well as an efficient training framework. Through experiments, we confirm that conventional decentralized training frameworks are vulnerable to attacks, and our approach effectively enhances its security. We anticipate that this paper can raise awareness of security concerns and contribute to enhancing the safety and reliability of decentralized training for LLMs.

REFERENCES

- Youssef Allouah, Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Rafaël Pinot, and John Stephan. 2023. Fixing by mixing: A recipe for optimal byzantine ml under heterogeneity. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1232–1300.
- [2] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. 2017. Privacy-preserving deep learning via additively homomorphic encryption. IEEE transactions on information forensics and security 13, 5 (2017), 1333–1345.
- [3] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. 2017. Privacy-preserving deep learning via additively homomorphic encryption. IEEE transactions on information forensics and security 13, 5 (2017), 1333–1345.
- [4] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. 2015. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. International Journal of Security and Networks 10, 3 (2015), 137–150.
- [5] Sanjith Athlur, Nitika Saran, Muthian Sivathanu, Ramachandran Ramjee, and Nipun Kwatra. 2022. Varuna: scalable, low-cost training of massive deep learning models. In Proceedings of the Seventeenth European Conference on Computer Systems. 472–487.
- [6] Mandeep Baines, Shruti Bhosale, Vittorio Caggiano, Naman Goyal, Siddharth Goyal, Myle Ott, Benjamin Lefaudeux, Vitaliy Liptchinsky, Mike Rabbat, Sam Sheiffer, et al. 2021. Fairscale: A general purpose modular pytorch library for high performance and large scale training.

- [7] Vladimir A Bogatyrev and AV Bogatyrev. 2015. Functional reliability of a realtime redundant computational process in cluster architecture systems. Automatic Control and Computer Sciences 49 (2015), 46–56.
- [8] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258 (2021).
- [9] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. Proceedings of machine learning and systems 1 (2019), 374–388.
- [10] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 1175–1191
- [11] Alexander Borzunov, Max Ryabinin, Tim Dettmers, Quentin Lhoest, Lucile Saulnier, Michael Diskin, Yacine Jernite, and Thomas Wolf. 2022. Training transformers together. In NeurIPS 2021 Competitions and Demonstrations Track. PMLR. 335–342.
- [12] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. Advances in neural information processing systems 33 (2020), 1877–1901.
- [13] Di Cao, Shan Chang, Zhijian Lin, Guohua Liu, and Donghong Sun. 2019. Understanding distributed poisoning attack in federated learning. In 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 233–239
- [14] Mahawaga Arachchige Pathum Chamikara, Peter Bertok, Ibrahim Khalil, Dongxi Liu, and Seyit Camtepe. 2021. Privacy preserving distributed machine learning with federated learning. Computer Communications 171 (2021), 112–125.
- [15] Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. Chatlaw: Open-source legal large language model with integrated external knowledge bases. arXiv preprint arXiv:2306.16092 (2023).
- [16] Arnhav Datar, Arun Rajkumar, and John Augustine. 2022. Byzantine spectral ranking. Advances in Neural Information Processing Systems 35 (2022), 27745– 27756
- [17] Michael Diskin, Alexey Bukhtiyarov, Max Ryabinin, Lucile Saulnier, Anton Sinitsin, Dmitry Popov, Dmitry V Pyrkin, Maxim Kashirin, Alexander Borzunov, Albert Villanova del Moral, et al. 2021. Distributed deep learning in open collaborations. Advances in Neural Information Processing Systems 34 (2021), 7879–7897.
- [18] Cheng Fang, Zhixiong Yang, and Waheed U Bajwa. 2022. BRIDGE: Byzantineresilient decentralized gradient descent. IEEE Transactions on Signal and Information Processing over Networks 8 (2022), 610–626.
- [19] Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Rafael Pinot, and John Stephan. 2022. Byzantine machine learning made easy by resilient averaging of momentums. In *International Conference on Machine Learning*. PMLR, 6246–6283.
- [20] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. arXiv preprint arXiv:1712.07557 (2017).
- [21] Meng Hao, Hongwei Li, Guowen Xu, Sen Liu, and Haomiao Yang. 2019. Towards efficient and privacy-preserving federated deep learning. In ICC 2019-2019 IEEE international conference on communications (ICC). IEEE, 1–6.
- [22] Soufiane Hayou and Fadhel Ayed. 2021. Regularization in resnet with stochastic depth. Advances in Neural Information Processing Systems 34 (2021), 15464–15474.
- [23] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep models under the GAN: information leakage from collaborative deep learning. In Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. 603–618.
- [24] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. 2022. Handcrafted backdoors in deep neural networks. Advances in Neural Information Processing Systems 35 (2022), 8068–8080.
- [25] Rui Hu, Yuanxiong Guo, Hongning Li, Qingqi Pei, and Yanmin Gong. 2020. Personalized federated learning with differential privacy. *IEEE Internet of Things Journal* 7, 10 (2020), 9530–9539.
- [26] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. 2016. Deep networks with stochastic depth. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. Springer, 646–661.
- [27] Quzhe Huang, Mingxu Tao, Zhenwei An, Chen Zhang, Cong Jiang, Zhibin Chen, Zirui Wu, and Yansong Feng. 2023. Lawyer LLaMA Technical Report. arXiv preprint arXiv:2305.15062 (2023).
- [28] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. 2019. Gpipe: Efficient training of giant neural networks using pipeline parallelism. Advances in neural information processing systems 32 (2019).
- [29] Technology Innovation Institute. 2023. Falcon 180B. https://falconllm.tii.ae/falcon-180b.html

- [30] Insu Jang, Zhenning Yang, Zhen Zhang, Xin Jin, and Mosharaf Chowdhury. 2023. Oobleck: Resilient Distributed Training of Large Models Using Pipeline Templates. In Proceedings of the 29th Symposium on Operating Systems Principles (Koblenz, Germany) (SOSP '23). Association for Computing Machinery, New York, NY, USA, 382–395. https://doi.org/10.1145/3600006.3613152
- [31] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. arXiv preprint arXiv:2307.10169 (2023).
- [32] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. 2003. On the privacy preserving properties of random data perturbation techniques. In Third IEEE international conference on data mining. IEEE, 99–106.
- [33] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. 2021. Byzantine-Robust Learning on Heterogeneous Datasets via Bucketing. In *International Conference* on Learning Representations.
- [34] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. 2021. Learning from history for byzantine robust optimization. In *International Conference on Machine Learning*. PMLR, 5311–5319.
- [35] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016).
- [36] Linyang Li, Demin Song, Xiaonan Li, Jiehang Zeng, Ruotian Ma, and Xipeng Qiu. 2021. Backdoor Attacks on Pre-trained Models by Layerwise Weight Poisoning. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 3023–3032. https://doi.org/10.18653/v1/2021.emnlpmain.241
- [37] Shigang Li and Torsten Hoefler. 2021. Chimera: efficiently training large-scale neural networks with bidirectional pipelines. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 1–14.
- [38] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala†. 2020. PyTorch distributed: experiences on accelerating data parallel training. Proceedings of the VLDB Endowment 13, 12 (2020), 3005–3018.
- [39] Xiaoyuan Liu, Hongwei Li, Guowen Xu, Rongxing Lu, and Miao He. 2020. Adaptive privacy-preserving federated learning. Peer-to-peer networking and applications 13 (2020), 2356–2366.
- [40] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics. PMLR, 1273-1282.
- [41] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BJ0hF1Z0b
- [42] Darya Melnyk, Yuyi Wang, and Roger Wattenhofer. 2018. Byzantine preferential voting. In International Conference on Web and Internet Economics. Springer, 327–340.
- [43] Payman Mohassel and Yupeng Zhang. 2017. Secureml: A system for scalable privacy-preserving machine learning. In 2017 IEEE symposium on security and privacy (SP). IEEE, 19–38.
- [44] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R Devanur, Gregory R Ganger, Phillip B Gibbons, and Matei Zaharia. 2019. PipeDream: Generalized pipeline parallelism for DNN training. In Proceedings of the 27th ACM Symposium on Operating Systems Principles. 1–15.
- [45] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.
- [46] Emin Orhan and Xaq Pitkow. 2018. Skip Connections Eliminate Singularities. In International Conference on Learning Representations. https://openreview.net/ forum?id=HkwBEMWCZ
- [47] David A Patterson, Garth Gibson, and Randy H Katz. 1988. A case for redundant arrays of inexpensive disks (RAID). In Proceedings of the 1988 ACM SIGMOD international conference on Management of data. 109–116.
- [48] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. OpenAI blog 1, 8 (2019), 9.
- [49] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 1–16.
- [50] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deep-speed: System optimizations enable training deep learning models with over 100 billion parameters. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 3505–3506.

- [51] Sashank J Reddi, Jakub Konečný, Peter Richtárik, Barnabás Póczós, and Alex Smola. 2016. Aide: Fast and communication efficient distributed optimization. arXiv preprint arXiv:1608.06879 (2016).
- [52] Max Ryabinin, Tim Dettmers, Michael Diskin, and Alexander Borzunov. 2023. Swarm parallelism: Training large models can be surprisingly communication-efficient. arXiv preprint arXiv:2301.11913 (2023).
- [53] Max Ryabinin, Tim Dettmers, Michael Diskin, and Alexander Borzunov. 2023. SWARM Parallelism: Training Large Models Can Be Surprisingly Communication-Efficient. In Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research), Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.), Vol. 202. PMLR, 29416–29440. https://proceedings. mlr.press/v202/ryabinin23a.html
- [54] Adi Shamir. 1979. How to share a secret. Commun. ACM 22, 11 (1979), 612-613.
- [55] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. arXiv preprint arXiv:2303.17580 (2023).
- [56] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. arXiv preprint arXiv:1909.08053 (2019).
- [57] Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2022. Large language models encode clinical knowledge. arXiv preprint arXiv:2212.13138 (2022).
- [58] Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature* 620, 7972 (2023), 172–180.
- [59] Zhenheng Tang, Yuxin Wang, Xin He, Longteng Zhang, Xinglin Pan, Qiang Wang, Rongfei Zeng, Kaiyong Zhao, Shaohuai Shi, Bingsheng He, et al. 2023. FusionAI: Decentralized Training and Deploying LLMs with Massive Consumer-Level GPUs. arXiv preprint arXiv:2309.01172 (2023).
- [60] Youming Tao, Sijia Cui, Wenlu Xu, Haofei Yin, Dongxiao Yu, Weifa Liang, and Xiuzhen Cheng. 2023. Byzantine-resilient federated learning at edge. IEEE Trans. Comput. (2023).
- [61] John Thorpe, Pengzhan Zhao, Jonathan Eyolfson, Yifan Qiao, Zhihao Jia, Minjia Zhang, Ravi Netravali, and Guoqing Harry Xu. 2023. Bamboo: Making Preemptible Instances Resilient for Affordable Training of Large {DNNs}. In 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23). 497–513.
- [62] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data poisoning attacks against federated learning systems. In Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25. Springer, 480–501.
- [63] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023).
- [64] Haiyong Wang and Kaixuan Guo. 2019. Byzantine Fault Tolerant Algorithm Based on Vote. In 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). 190–196. https://doi.org/10.1109/ CyberC.2019.00041
- [65] Jue Wang, Yucheng Lu, Binhang Yuan, Beidi Chen, Percy Liang, Christopher De Sa, Christopher Re, and Ce Zhang. 2023. CocktailSGD: Fine-tuning Foundation Models over 500Mbps Networks. In *International Conference on Machine Learning*. PMLR, 36058–36076.
- [66] BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. arXiv preprint arXiv:2211.05100 (2022).
- [67] Honglin Xiong, Sheng Wang, Yitao Zhu, Zihao Zhao, Yuxiao Liu, Qian Wang, and Dinggang Shen. 2023. Doctorglm: Fine-tuning your chinese doctor is not a herculean task. arXiv preprint arXiv:2304.01097 (2023).
- [68] Bowen Yang, Jian Zhang, Jonathan Li, Christopher Ré, Christopher Aberger, and Christopher De Sa. 2021. Pipemare: Asynchronous pipeline parallel dnn training. Proceedings of Machine Learning and Systems 3 (2021), 269–296.
- [69] Binhang Yuan, Yongjun He, Jared Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy S Liang, Christopher Re, and Ce Zhang. 2022. Decentralized training of foundation models in heterogeneous environments. Advances in Neural Information Processing Systems 35 (2022), 25464–25477.
- [70] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. 2020. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In 2020 USENIX annual technical conference (USENIX ATC 20). 493–506.
- [71] Sixin Zhang, Anna E Choromanska, and Yann LeCun. 2015. Deep learning with elastic averaging SGD. Advances in neural information processing systems 28 (2015).
- [72] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt:

- Open pre-trained transformer language models. arXiv preprint arXiv:2205.01068 (2022).
- [73] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. idlg: Improved deep leakage from gradients. arXiv preprint arXiv:2001.02610 (2020).
- [74] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. arXiv preprint arXiv:2303.18223 (2023).
- [75] Banghua Zhu, Lun Wang, Qi Pang, Shuai Wang, Jiantao Jiao, Dawn Song, and Michael I Jordan. 2023. Byzantine-robust federated learning with optimal statistical rates. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3151–3178.
- [76] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. Advances in neural information processing systems 32 (2019).