Position: Exploring the Robustness of Pipeline-Parallelism-Based Decentralized Training

Lin Lu [*1] **Chenxi Dai** [*1] **Wangcheng Tao** [2] **Binhang Yuan** [2] **Yanan Sun** [3] **Pan Zhou** [1]

Abstract

Modern machine learning applications increasingly demand greater computational resources for training large models. Decentralized training has emerged as an effective means to democratize this technology. However, the potential threats associated with this approach remain inadequately discussed, posing a hurdle to the development of decentralized training infrastructures. This paper aims to initiate discussion towards this end by exploring the robustness of decentralized training from three primary perspectives. Firstly, we articulate our position on establishing robust decentralized training by outlining potential threats and the corresponding countermeasures. Secondly, we illustrate a nascent poisoning attack targeting decentralized training frameworks, easily executable by malicious stages. To mitigate this security threat and ensure efficient training, we propose a robust training framework, integrating a 100% detection strategy and efficient training mechanisms. Finally, we demonstrate the severity of the proposed attack and the effectiveness of our robust training framework. This position paper emphasizes the urgency of exploring the robustness of decentralized training and proposes a feasible solution. The code is available at `https://github.com/dcx001016/pipeline_attack`

1. Introduction

Deep neural networks (DNNs), particularly large language models (LLMs) (Brown et al., 2020; Touvron et al., 2023; Zhang et al., 2022; Workshop et al., 2022), have demonstrated exceptional accuracy in various domains, thus gaining widespread acceptance and usage (Cui et al., 2023; Singhal et al., 2023; Shen et al., 2023). However, for enhanced accuracy across different domains, DNNs have expanded aggressively in terms of model scale and pre-train data volumes, resulting in time- and cost-intensive training processes (Bommasani et al., 2021; Kaddour et al., 2023; Zhao et al., 2023). For example, the Falcon-180B (Institute, 2023) model has 180 billion parameters trained on 3.5 trillion tokens. Naturally, due to the intensive computational load, scaling DNNs training has garnered significant attention over the past years (Shoeybi et al., 2019; Narayanan et al., 2021; Lian et al., 2022; Yuan et al., 2022). Among these strategies, parallel training frameworks have emerged as a primary approach to addressing this computational intensity (Huang et al., 2019; Li et al., 2020; Narayanan et al., 2019; 2021; Rajbhandari et al., 2020).

A promising direction to democratize the training of large DNNs is through decentralized training (Diskin et al., 2021; Ryabinin et al., 2023b; Yuan et al., 2022), 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* (Huang et al., 2019; Narayanan et al., 2019)). These parallel paradigms require communication of *activations* during forward propagation and *corresponding gradients* during backward propagation, which is fundamentally different from federated learning (FL) that only requires synchronization of *model gradients* in a data parallel paradigm. As a result, the potential threats associated with such decentralized training have not been formally discussed, which may hinder the democratization of LLMs.

The most relevant technique discussed on resilient parallel training strategies in the machine learning communities is secure aggregation in FL, which limits its scope under the data parallel paradigm (Fang et al., 2022; Karimireddy et al., 2021b; Farhadkhani et al., 2022; Tao et al., 2023). In such scenarios, when malicious gradient values arise, the parameter server employs resilient gradient aggregation methods, which mainly employ outlier detection algorithms to mitigate the impact of these malicious gradient values on the global model. Different from the security concerns and re-

[*]Equal contribution [1]Huazhong University of Science and Technology [2]Hong Kong University of Science and Technology [3]Sichuan University. Correspondence to: Pan Zhou <panzhou@hust.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

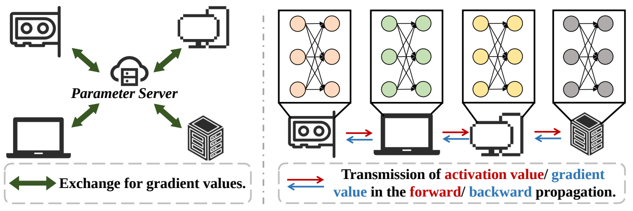
silient methods in FL, the communication of activations and the corresponding gradients in decentralized training could lead to distinct safety issues, demanding distinct approaches for malicious detection and defense.

Therefore, we highlight the pressing issue of addressing the robustness of decentralized training and advocate for heightened attention to this matter within the machine learning communities. In general, this paper aims to explore the following critical research questions: **(RQ1)** What are the potential threats inherent in decentralized training? **(RQ2)** How are these vulnerabilities exploited, and what are the resulting adverse consequences? **(RQ3)** How can we develop resilient strategies to counteract these threats?

(Contribution 1) To explore the vulnerability of decentralized training, we initially introduce a threat model specific to this scenario. Guided by this threat model, we have examined two classic potential threats: privacy inference attacks and poisoning attacks. Then we analyze the methods through which a malicious stage can execute these attacks easily. Subsequently, in order to elucidate the distinct nature of security issues in decentralized training, we delineate two key disparities between decentralized training and FL, highlighting the structural differences between pipeline parallelism and data parallelism. Therefore, defense algorithms designed specifically for FL cannot be directly employed. Finally, we concentrate on the robustness of the algorithm and the efficiency of the system training, analyzing the necessary components of a robust decentralized training framework from both an algorithm view and a system view.

(Contribution 2) To substantiate the validity and severity of the presented threat model, we implement a poisoning attack in decentralized training. Targeting the forward and backward propagation processes between stages in decentralized training, we design *forward attack* and *backward attack* using activation value poisoning and gradient poisoning, respectively. Through symbol flipping and noise injection, these two attack methods tamper with the original transmission values. Our experiments demonstrate that our attack method can impede the LLM from converging, even after a significant number of training iterations.

(Contribution 3) Based on the above threat model, we have formulated a robust training framework under decentralized training, which mainly includes the detection strategy and efficient training. The detection strategy can identify the presence of a malicious poisoning attacker with a 100% probability within the current training iteration, notwithstanding the assumption that the attacker possesses sufficient capability. The efficient training methodology replaces the traditional restart training technique. By skipping the malicious stage during training, a reduction in the number of training iterations is achieved, concurrently ensuring model convergence. Furthermore, we demonstrate the efficiency


Parameter Server
Exchange for gradient values.
Transmission of activation value/ gradient value in the forward/ backward propagation.

(a) Data Parallelism (b) Pipeline Parallelism

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

of our robust training framework through empirical study.

This position paper aims to scrutinize the potential threats in decentralized training and propose possible robust methods. We hope our position on exploring the robustness of decentralized training will attract wide attention from the machine learning communities.

2. Background

Parallel training for LLMs. To distribute the training computation of LLMs 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 (Li et al., 2020). FL (McMahan et al., 2017; Konečný et al., 2016; Bonawitz et al., 2019) 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 (Zhang et al., 2015; Reddi et al., 2016). *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 (Huang et al., 2019; Narayanan et al., 2019; Yang et al., 2021). In contrast to data parallelism, pipeline parallelism optimizes the utilization of computational resources (Shoeybi et al., 2019; Rasley et al., 2020; Baines et al., 2021), and has become the main technique for decentralized training to train larger DNNs (e.g, LLMs).

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. (Yuan et al., 2022) initially investigates the decentralized training for large foundation models using model parallelism. Subsequently, some studies (Ryabinin et al., 2023a; Wang et al., 2023) accomplish billion-scale training on heterogeneous devices with slow interconnect. Similarly, (Tang et al., 2023) 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 (Tang et al., 2023; Borzunov et al., 2022), no systematic research has studied this issue extensively. Existing research focuses mainly on ensuring seamless pipeline operations (Athlur et al., 2022; Thorpe et al., 2023; Jang et al., 2023). However, they only discuss machine failures, neglecting the vulnerability of decentralized training to various imperceptible security risks.

3. Position: Decentralized Training Robustness

In this section, we address **RQ1** by presenting a comprehensive perspective on the urgency of robust decentralized training. Our primary focus is on the potential threats, the limitations of existing defense methods migrating to this scenario, and concerns regarding the balance between robustness and efficiency.

3.1. Potential Threat Model and Attacks

This subsection primarily discusses the potential threats in decentralized training. Initially, We introduce a threat model tailored to this context. Based on this threat model, we examine two potential threats: privacy inference attacks and poisoning attacks, and their consequences. We observe that these two classic attack forms indicate different capabilities of two types of attackers. A less sophisticated attacker can stealthily pilfer the training data through transmission value without being perceived. A more sophisticated attacker can manipulate the transmission value, resulting in a significant reduction in global model training accuracy.

3.1.1. Threat Model

Following the initial problem setting of the Byzantine problem in FL (Blanchard et al., 2017), we propose a decentralized training framework with K computation stages, where M_i denotes the sub-layer of the i-th stage. In contrast to the previous works that aim to study the fault tolerance problem, we consider a decentralized training framework whose stages are not trustworthy. Given the openness of the training process and the uncertainty of the attending stages, we assume each stage in this pipeline could be a malicious stage, denoted as $\mathcal{A}$. Specifically, as the owner of the training data and the corresponding labels, we assume that the initiate and the final stage are absolutely trustworthy. To enhance realism without sacrificing generality, we presume that it is like a white box for $\mathcal{A}$, who has the knowledge of the entire framework and most training details.

3.1.2. Potential Attacks

Based on the above threat models, many potential attacks are derived. We mainly describe the realizability of privacy inference attacks and poisoning attacks under the framework of decentralized training and their new attack paradigms.

Privacy inference attacks. DNNs, especially LLMs, have found widespread application in various domains, such as healthcare (Xiong et al., 2023; Singhal et al., 2022) and law (Cui et al., 2023; Huang et al., 2023). 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 (Zhu et al., 2019; Aono et al., 2017a). Certain study (Zhao et al., 2020) even introduces an approach that achieves 100% accuracy in extracting ground-truth labels from the gradients.

Furthermore, in addition to directly accessing the original data, some studies (Ateniese et al., 2015; Hitaj et al., 2017) focus on properties unrelated to the characteristic features of the class. In such a scenario, 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.

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. Untargeted attacks hinder the model's convergence by freely manipulating transmitting values, whereas targeted attacks aim to inject backdoors into the global model.

Previous studies (Cao et al., 2019; Tolpegin et al., 2020) thoroughly investigate the detrimental impact of untargeted attacks on the convergence of the global model in distributed systems. They were either limited to FL scenarios or only involved poisoning datasets by tampering with the corresponding labels. However, 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 also show the possible consequences of this new untargeted poisoning attack form in Section 4.

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 (Li et al., 2021; Hong et al., 2022) 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 training as well. We evaluate the vulnerability of decentralized training to poisoning attacks in Section 3.2, providing an explanation for why decentralized training is more susceptible to such attacks compared to FL.

3.2. Decentralized Training vs. Federated Learning

In this subsection, we posit that the direct application of current security methods in FL to decentralized training encounters significant challenges for the following reasons by illustrating the structural differences between them.

3.2.1. 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 (Bonawitz et al., 2017) or secret-sharing-based methods (Bonawitz et al., 2017), are used to prevent privacy inference attacks. To mitigate poisoning attacks, outlier detection algorithms, like the voting mechanism (Melnyk et al., 2018; Datar et al., 2022; Wang & Guo, 2019) and bucketing mechanism (Karimireddy et al., 2021a; Zhu et al., 2023; Allouah et al., 2023) 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 (Li & Hoefler, 2021; Narayanan et al., 2021; Jang et al., 2023), 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 3.1.2, 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.

3.2.2. Change of Transmitting Mechanism

Compared to the data transmitting mechanism in FL, the exchange of objects and the 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 training accuracy. 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.

3.3. Robustness or Efficiency

In this subsection, we outline essential components for a robust and efficient decentralized training framework from two perspectives. From an algorithm view, we analyze the necessary modules required to secure the decentralized training process against the aforementioned two attack forms. On the other hand, from a system view, we elaborate on the key technologies that sustain the decentralized system throughput. Upon examining the associated challenges, we conclude that achieving robustness, efficiency, and accuracy is akin to an impossible trinity. Therefore, ensuring global model convergence through robust measures invariably results in decreased system throughput.

3.3.1. An algorithm view

Privacy preservation. Privacy-preserving methods, particularly in FL, can be categorized into two main approaches: encryption-based and perturbation-based methods. Encryption-based methods encompass homomorphic encryption (Aono et al., 2017b; Zhang et al., 2020), secret sharing (Shamir, 1979), and secure multiparty computation methods (Mohassel & Zhang, 2017). 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. A promising direction is homomorphic encryption which allows computation on ciphertext and retrieval of the computed plaintext with a single decryption operation. However, it imposes stringent requirements on the calculation method and the time it occupies cannot be overlooked.

Perturbation-based methods, such as differential privacy (Geyer et al., 2017; Hao et al., 2019; McMahan et al., 2018) and additive perturbation (Chamikara et al., 2021; Hu et al., 2020; Liu et al., 2020) 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 (Kargupta et al., 2003), 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.

Stage-level detection for malicious behaviors. 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 detecting 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 5, we present a comprehensive case study to illustrate the effectiveness of this detection methodology. Despite the additional training time, our approach's defense capability convincingly validates its potential for future research.

3.3.2. A SYSTEM VIEW

In the event of a hardware failure or a detected attack, it is crucial for the decentralized system to keep the pipeline seamless and recover promptly. This matter has garnered significant attention as a problem of fault tolerance and frequent interruptions in decentralized training. Relevant research focuses on maintaining model throughput while enabling automatic recovery from this malfunction (Athlur et al., 2022; Thorpe et al., 2023; Jang et al., 2023). However, several challenges remain unsolved. For example, the aforementioned approaches experience additional overhead of loading the checkpoint when restarting the training process. On the other hand, some strategies even require extra computing resources as a backup, which contradicts the original objective of decentralized training. Furthermore, these strategies tend to emphasize the reliability of each stage on the pipeline, while ignoring their potential to create malicious behaviors.

Another straightforward approach is to use a combination of malicious behavior detection and restarting the training iteration from updating the wrong gradients. However, leaving aside the time consumption caused by the detection algorithm, the restart method wastes the results obtained in the current training iteration and leads to prolonged idle time. Consequently, the subsequent computing resources have to remain underutilized for an extended period. Therefore, ensuring robustness in a decentralized training framework while maintaining high system throughput is a rather tricky problem. In Section 5, we present alternative solutions to minimize computing resource consumption while achieving swift recovery from failures or attacks.

4. *Forward Attacks* and *Backward Attacks*

This section addresses **RQ2** by introducing our attack methods: *forward attack* and *backward attack*. In contrast to Section 3, which provides a general overview of the decentralized training robustness, this section focuses on a specific security scenario: We consider a scenario where an attacker can only compromise one stage in either forward activation value propagation or backward gradient value propagation. To maintain generality, we assume the attacker's index can vary while keeping other assumptions constant.

If $\mathcal{A}$ gains control of a stage or a certain stage intends to exhibit malicious behaviors, it transmits the tampered value $\mathbf{a}'_{\text{out}}$ to the subsequent stage, instead of the intended output $\mathbf{a}_{\text{out}}$, upon receiving a value $\mathbf{a}_{\text{in}}$. The malicious behavior during the model training process can be categorized as either a *forward attack* or a *backward attack* depending on the context, as demonstrated in Figure 2 by orange and blue arrows, respectively. Notably, according to the threat model, we assume that the initial and the final stages, functioning as data providers, remain immune to attacks.

We employ two straightforward untargeted poisoning attack methods to simulate the actions of $\mathcal{A}$. In *forward attack*,

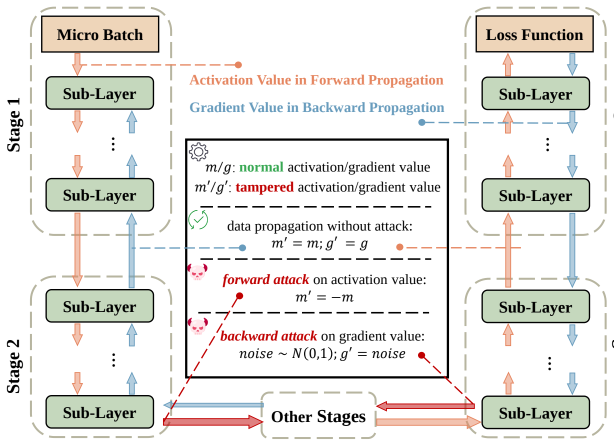

Micro Batch
Loss Function
Activation Value in Forward Propagation
Gradient Value in Backward Propagation
Sub-Layer
Stage 1
Stage 2
Stage K
Stage K − 1
m/g: normal activation/gradient value
m'/g': tampered activation/gradient value
data propagation without attack: $m' = m; g' = g$
forward attack on activation value: $m' = -m$
backward attack on gradient value: $noise \sim N(0,1); g' = noise$
Other Stages

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

the malicious stage simply flips the sign of $\mathbf{a}_{\text{out}}$ 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 $\mathbf{a}_{\text{out}}$ and sets $\mathbf{a}'_{\text{out}} = \phi$ to modify the malicious behavior of random noise injection. Then the malicious stage sends $\mathbf{a}'_{\text{out}}$ to the next stage.

5. Our Robust Training Framework

This section addresses **RQ3** by introducing our robust training framework consisting of two main components: attack detection and efficient training, as 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 and ensure training efficiency. Since the backward propagation mirrors the forward propagation but with the reversed data transmission direction, we take the forward propagation as an example and analyze our robust training framework in detail in this section.

5.1. Detection Strategy

Aiming to equip each stage with the capability to verify the accuracy of received values and naturally drawing the inspiration of redundant computation (Patterson et al., 1988; Bogatyrev & Bogatyrev, 2015) and Bamboo (Thorpe et al., 2023), we propose the *duplicated block* which takes the place of the original sub-layers in each stage and serves as a crucial component of our detection strategy. Taking the i-th stage as an example, it consists of the redundant layers M'_{i-1}, duplicated from M_{i-1} of the $(i-1)$-th stage, and the original layers M_i of the i-th stage. To ensure the detection accuracy, the parameters of M_{i-1} and M'_{i-1} remain identical throughout the entire training process.

During the detection part, 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.

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 experienced attacker who leverages an arbitrary input $\mathbf{a}'_{\text{in}}$ and sends the corresponding $\mathbf{a}'_{\text{out}}$ to the next stage.

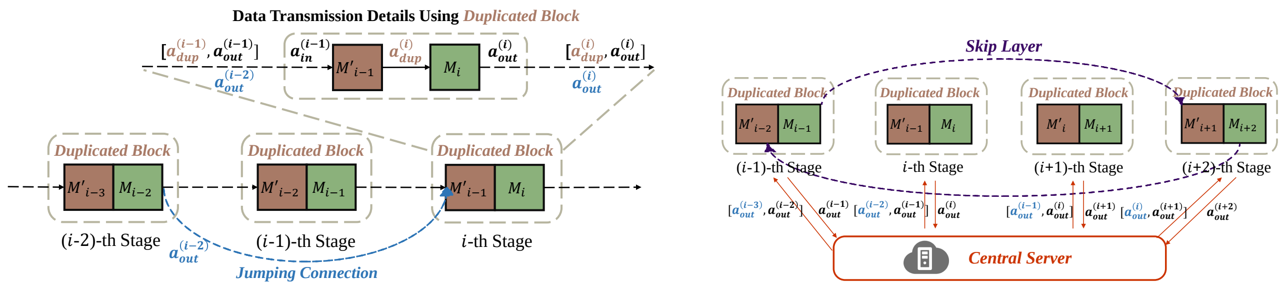
To sum up, the i-th stage needs to verify two conditions: **i)** check if $\mathbf{a}_{\text{dup}}^{(i-1)}$ is equal to $\mathbf{a}_{\text{out}}^{(i-2)}$, and **ii)** confirm if the pair $[\mathbf{a}_{\text{dup}}^{(i-1)}, \mathbf{a}_{\text{out}}^{(i-1)}]$ matches on M'_{i-1}. If either of these two conditions is not met, 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.

5.2. Efficient Training

Although our detection strategy successfully identifies all malicious activities carried out by $\mathcal{A}$, accurately pinpointing the exact stage of malicious behavior poses a challenge. Now we present a detailed logical analysis specifically addressing the errors encountered with the $(i+1)$-th stage:

- The $(i+1)$-th stage itself is malicious and intentionally reports an error whatever it receives.
- The i-th stage is malicious and transmits tampered value pair $[\mathbf{a}_{\text{dup}}^{(i)}, \mathbf{a}_{\text{out}}^{(i)}]$ to the $(i+1)$-th stage.
- The $(i-1)$-th stage is malicious. It sends normal values to the i-th stage, but transmits a manipulated $\mathbf{a}_{\text{out}}^{(i-1)}$ to the $(i+1)$-th stage.

To narrow down the scope of suspicion, we propose *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



(a) Detection strategy (b) Efficient training framework

Figure 3. Details about data transmission and structure of our proposed robust 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.

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 (Huang et al., 2016; Orhan & Pitkow, 2018; Hayou & Ayed, 2021), 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. Experiments

Our experiments aim to validate our attack methods and robust training framework. Specifically, we demonstrate: **i)** Decentralized pipeline parallelism training is vulnerable to both *forward attack* and *backward attack*, which negatively impact the model's convergence and final metric; **ii)** When employing our robust training strategy, the training effectiveness of LLMs is comparable to the results under normal conditions and may even perform better; and **iii)** Our robust training framework enhances the training process of the pipeline parallel strategy compared to the restart framework. It is important to note that our experiments do not solely assess the detection strategy, as we have already logically proven its ability to detect malicious behaviors with one hundred percent accuracy in Subsection 5.1. However, its significance as the foundation of the robust training framework should not be overlooked.

6.1. Experimental Setup

Datasets and benchmarks. Decentralized training frameworks are commonly utilized in the fine-tuning of LLMs. Various LLMs, such as GPT-2 (Radford et al., 2019), Bloom (Workshop et al., 2022), and Opt (Zhang et al., 2022), are selected with parameter sizes ranging from 345M to 7B. Checkpoints for all models above can be obtained from HuggingFace. We employ text-generation tasks on wikitext2, arxiv abstracts, and openwebtext datasets to conduct our evaluations. All datasets are publicly available and do not contain sensitive or offensive content. Perplexity serves as our primary metric for evaluating model performance, and our experiments are founded on the GPipe (Huang et al., 2019) framework.

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

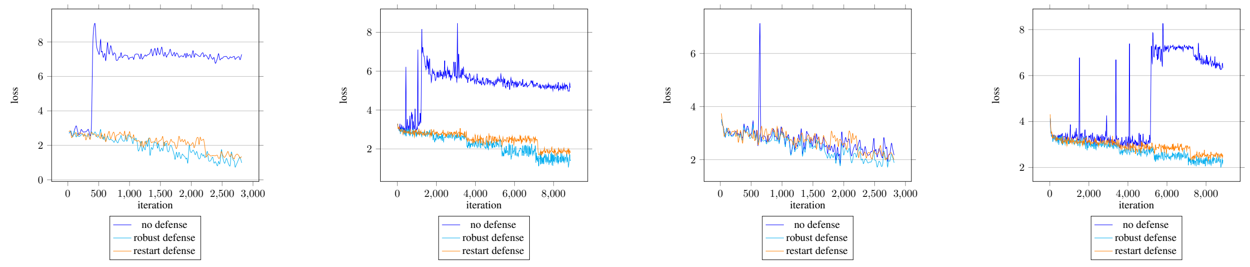
Decentralized computing environment. To simulate heterogeneous computing resources in real scenarios and to train LLMs of varying sizes, we utilize several types of GPU devices, including A40, V100, RTX 3090, and Quadro RTX 5000. For a single pipeline, we partition a model onto 6 machines.

Baselines. 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.

Hyperparameter tuning. We set the learning rate to $5e{-}6$ during training, and the batch size and micro-batch size to 4 and 1, respectively.

6.2. Main Results

Vulnerability of decentralized training. We first evaluate the vulnerability of different LLMs under various malicious behaviors. We evaluate the impact of *forward attack* and



(a) WikiText2,gpt2-1.5b (b) arXiv,gpt2-1.5b (c) WikiText2,opt-350m (d) arXiv,opt-350m

Figure 4. Training loss on different clean iterations under method without defense, robust training framework, and restart method.

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

backward attack on the training outcome. The obtained results are presented in Table 1. We notice that the two attack methods perform exceptionally well when the attacking rate is set to 0.7. Without applying any defense measures and after sufficient training iterations, the perplexity of the attacked model usually degrades to at least 10 times or more of the original model and exceeds 200 times at an attack rate of 0.7. However, the attack's effectiveness is not always as good when the attack rate is set to 0.3. We analyze that this may be due to the small probability of malicious values being flipped or added noise, which is offset by the high probability of normal values, allowing the model to converge with sufficient training time. The overfitting phenomenon resulting from the long training iterations may constitute a prominent factor contributing to the performance of the normal model to be inferior to that of the attacked model with the attack rate set to 0.3.

Effectiveness of robust training framework. We show the effectiveness of our robust training framework in Table 2. We denote the model's perplexity on the clean model, the attacked model without any defense, and the attacked model under our robust training framework by clean, attack, and ours respectively. In the attack configuration of this experiment, we select *forward attack* as the attack criterion for its greater impact. In all comparisons, we demonstrate the robust training framework's ability to resist attacks.

First, when using the robust training framework, the perplexity of the model can be at most 74.6 times better than that of the perplexity of the attacked model. What's more, models using this framework even exhibit lower perplexity than the original clean models without attacks. Even when testing the perplexity of Bloom-560M on dataset arxiv21, we find that the models using this robust framework have a perplexity of only one-third that of the original model. We speculate that this parallels the anomalous results in Table 1. Compared with the regular training process, the skip-layer mechanism serves as an effective regularization technique and successfully alleviates model overfitting.

6.3. Robust Training Framework vs. Traditional Restart

In order to evaluate the training efficiency of our robust training framework, we conduct a comparison between the traditional restart method which simply restarts the current training iteration once an attack is detected based on the detection strategy described in section 5.1, and our proposed robust training framework. We recorded the training loss at each iteration for the two defense methods across all models and datasets. To enhance visual clarity, we exclude the loss corresponding to the iteration under attack and plot a point every 10 clean iterations, as shown in Figure 4. The x-axis represents the real step, while the y-axis represents the selected clean iteration's training loss. We observe that our robust training framework consistently exhibits lower loss than the traditional restart method at most iterations, indicating that our approach facilitates faster model convergence. Simultaneously, when training without any defensive measures, the loss remains persistently high, further substantiating the presence of vulnerabilities in the decentralized pipeline parallelism training process.

7. Conclusion

Through this paper, we focus on investigating the robustness of decentralized training frameworks that implement pipeline parallelism. Initially, we introduce the threat model specific to this context, based on which we delineate two potential attack forms and their consequences. We then compare the structural differences between decentralized training and FL. Additionally, we scrutinize the fundamental reasons why existing security methods cannot be readily applied to the decentralized training frameworks. In light of this analysis, we present a vision for a decentralized training framework with both robustness and training efficiency. To underscore the pressing nature of security concerns in decentralized training, we exemplify two untargeted poisoning attacks, named *forward attack* and *backward attack*, using straightforward methodologies. Building on these poisoning attacks, we provide our robust training framework, consisting of the detection strategy and efficient training mechanism to identify the aforementioned malicious behaviors while ensuring training efficiency. Through experiments, we confirm that conventional decentralized training frameworks are vulnerable to attacks, while our approach effectively improves their robustness and efficiency. We anticipate that this position paper can raise awareness of security concerns and contribute to enhancing the safety and reliability of decentralized training.

Acknowledgements

We sincerely thank all the reviewers for their valuable reviews of our position paper.

Impact Statement

This paper investigates the robustness of decentralized training frameworks. By identifying vulnerabilities and proposing a more robust training framework, our research contributes to mitigating the risks of malicious attacks and enhancing the security of decentralized training. We acknowledge the significance of ethical considerations in machine learning, and our work aligns with the current endeavors to ensure the responsible development and deployment of AI technologies. While there may be additional ethical implications that arise from the broader adoption of decentralized training, we believe they are well-established in the field and do not require specific highlighting in this paper.

References

Allouah, Y., Farhadkhani, S., Guerraoui, R., Gupta, N., Pinot, R., and Stephan, J. Fixing by mixing: A recipe for optimal byzantine ml under heterogeneity. In *International Conference on Artificial Intelligence and Statistics*, pp. 1232–1300. PMLR, 2023.

Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE transactions on information forensics and security*, 13(5):1333–1345, 2017a.

Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE transactions on information forensics and security*, 13(5):1333–1345, 2017b.

Ateniese, G., Mancini, L. V., Spognardi, A., Villani, A., Vitali, D., and Felici, G. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3):137–150, 2015.

Athlur, S., Saran, N., Sivathanu, M., Ramjee, R., and Kwatra, N. Varuna: scalable, low-cost training of massive deep learning models. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pp. 472–487, 2022.

Baines, M., Bhosale, S., Caggiano, V., Goyal, N., Goyal, S., Ott, M., Lefaudeux, B., Liptchinsky, V., Rabbat, M., Sheiffer, S., et al. Fairscale: A general purpose modular pytorch library for high performance and large scale training, 2021.

Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017.

Bogatyrev, V. A. and Bogatyrev, A. Functional reliability of a real-time redundant computational process in cluster architecture systems. *Automatic Control and Computer Sciences*, 49:46–56, 2015.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, B., et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

Borzunov, A., Ryabinin, M., Dettmers, T., Lhoest, Q., Saulnier, L., Diskin, M., Jernite, Y., and Wolf, T. Training transformers together. In *NeurIPS 2021 Competitions and Demonstrations Track*, pp. 335–342. PMLR, 2022.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Cao, D., Chang, S., Lin, Z., Liu, G., and Sun, D. Understanding distributed poisoning attack in federated learning. In *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 233–239. IEEE, 2019.

Chamikara, M. A. P., Bertok, P., Khalil, I., Liu, D., and Camtepe, S. Privacy preserving distributed machine learning with federated learning. *Computer Communications*, 171:112–125, 2021.

Cui, J., Li, Z., Yan, Y., Chen, B., and Yuan, L. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *arXiv preprint arXiv:2306.16092*, 2023.

Datar, A., Rajkumar, A., and Augustine, J. Byzantine spectral ranking. *Advances in Neural Information Processing Systems*, 35:27745–27756, 2022.

Diskin, M., Bukhtiyarov, A., Ryabinin, M., Saulnier, L., Sinitsin, A., Popov, D., Pyrkin, D. V., Kashirin, M., Borzunov, A., Villanova del Moral, A., et al. Distributed deep learning in open collaborations. *Advances in Neural Information Processing Systems*, 34:7879–7897, 2021.

Fang, C., Yang, Z., and Bajwa, W. U. Bridge: Byzantine-resilient decentralized gradient descent. *IEEE Transactions on Signal and Information Processing over Networks*, 8:610–626, 2022.

Farhadkhani, S., Guerraoui, R., Gupta, N., Pinot, R., and Stephan, J. Byzantine machine learning made easy by resilient averaging of momentums. In *International Conference on Machine Learning*, pp. 6246–6283. PMLR, 2022.

Geyer, R. C., Klein, T., and Nabi, M. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

Hao, M., Li, H., Xu, G., Liu, S., and Yang, H. Towards efficient and privacy-preserving federated deep learning. In *ICC 2019-2019 IEEE international conference on communications (ICC)*, pp. 1–6. IEEE, 2019.

Hayou, S. and Ayed, F. Regularization in resnet with stochastic depth. *Advances in Neural Information Processing Systems*, 34:15464–15474, 2021.

Hitaj, B., Ateniese, G., and Perez-Cruz, F. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 603–618, 2017.

Hong, S., Carlini, N., and Kurakin, A. Handcrafted backdoors in deep neural networks. *Advances in Neural Information Processing Systems*, 35:8068–8080, 2022.

Hu, R., Guo, Y., Li, H., Pei, Q., and Gong, Y. Personalized federated learning with differential privacy. *IEEE Internet of Things Journal*, 7(10):9530–9539, 2020.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 646–661. Springer, 2016.

Huang, Q., Tao, M., An, Z., Zhang, C., Jiang, C., Chen, Z., Wu, Z., and Feng, Y. Lawyer llama technical report. *arXiv preprint arXiv:2305.15062*, 2023.

Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

Institute, T. I. Falcon 180b, 2023. URL https://falconllm.tii.ae/falcon-180b.html.

Jang, I., Yang, Z., Zhang, Z., Jin, X., and Chowdhury, M. Oobleck: Resilient distributed training of large models using pipeline templates. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 382–395, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613152. URL https://doi.org/10.1145/3600006.3613152.

Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., and McHardy, R. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*, 2023.

Kargupta, H., Datta, S., Wang, Q., and Sivakumar, K. On the privacy preserving properties of random data perturbation techniques. In *Third IEEE international conference on data mining*, pp. 99–106. IEEE, 2003.

Karimireddy, S. P., He, L., and Jaggi, M. Byzantine-robust learning on heterogeneous datasets via bucketing. In *International Conference on Learning Representations*, 2021a.

Karimireddy, S. P., He, L., and Jaggi, M. Learning from history for byzantine robust optimization. In *International Conference on Machine Learning*, pp. 5311–5319. PMLR, 2021b.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Li, L., Song, D., Li, X., Zeng, J., Ma, R., and Qiu, X. Backdoor attacks on pre-trained models by layerwise weight poisoning. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3023–3032, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.241. URL https://aclanthology.org/2021.emnlp-main.241.

Li, S. and Hoefler, T. Chimera: efficiently training large-scale neural networks with bidirectional pipelines. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–14, 2021.

Li, S., Zhao, Y., Varma, R., Salpekar, O., Noordhuis, P., Li, T., Paszke, A., Smith, J., Vaughan, B., Damania, P., and Chintala†, S. Pytorch distributed: experiences on accelerating data parallel training. *Proceedings of the VLDB Endowment*, 13(12):3005–3018, 2020.

Lian, X., Yuan, B., Zhu, X., Wang, Y., He, Y., Wu, H., Sun, L., Lyu, H., Liu, C., Dong, X., et al. Persia: An open, hybrid system scaling deep learning-based recommenders up to 100 trillion parameters. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3288–3298, 2022.

Liu, X., Li, H., Xu, G., Lu, R., and He, M. Adaptive privacy-preserving federated learning. *Peer-to-peer networking and applications*, 13:2356–2366, 2020.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BJ0hF1Z0b.

Melnyk, D., Wang, Y., and Wattenhofer, R. Byzantine preferential voting. In *International Conference on Web and Internet Economics*, pp. 327–340. Springer, 2018.

Mohassel, P. and Zhang, Y. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, pp. 19–38. IEEE, 2017.

Narayanan, D., Harlap, A., Phanishayee, A., Seshadri, V., Devanur, N. R., Ganger, G. R., Gibbons, P. B., and Zaharia, M. Pipedream: Generalized pipeline parallelism for dnn training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pp. 1–15, 2019.

Narayanan, D., Shoeybi, M., Casper, J., LeGresley, P., Patwary, M., Korthikanti, V., Vainbrand, D., Kashinkunti, P., Bernauer, J., Catanzaro, B., et al. 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*, pp. 1–15, 2021.

Orhan, E. and Pitkow, X. Skip connections eliminate singularities. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HkwBEMWCZ.

Patterson, D. A., Gibson, G., and Katz, R. H. A case for redundant arrays of inexpensive disks (raid). In *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, pp. 109–116, 1988.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16. IEEE, 2020.

Rasley, J., Rajbhandari, S., Ruwase, O., and He, Y. Deepspeed: 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*, pp. 3505–3506, 2020.

Reddi, S. J., Konečný, J., Richtárik, P., Póczós, B., and Smola, A. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

Ryabinin, M., Dettmers, T., Diskin, M., and Borzunov, A. Swarm parallelism: Training large models can be surprisingly communication-efficient. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 29416–29440. PMLR, 23–29 Jul 2023a. URL https://proceedings.mlr.press/v202/ryabinin23a.html.

Ryabinin, M., Dettmers, T., Diskin, M., and Borzunov, A. Swarm parallelism: Training large models can be surprisingly communication-efficient. *arXiv preprint arXiv:2301.11913*, 2023b.

Shamir, A. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

Shen, Y., Song, K., Tan, X., Li, D., Lu, W., and Zhuang, Y. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*, 2023.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., et al. Large language models encode clinical knowledge. *arXiv preprint arXiv:2212.13138*, 2022.

Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.

Tang, Z., Wang, Y., He, X., Zhang, L., Pan, X., Wang, Q., Zeng, R., Zhao, K., Shi, S., He, B., et al. Fusionai: Decentralized training and deploying llms with massive consumer-level gpus. *arXiv preprint arXiv:2309.01172*, 2023.

Tao, Y., Cui, S., Xu, W., Yin, H., Yu, D., Liang, W., and Cheng, X. Byzantine-resilient federated learning at edge. *IEEE Transactions on Computers*, 2023.

Thorpe, J., Zhao, P., Eyolfson, J., Qiao, Y., Jia, Z., Zhang, M., Netravali, R., and Xu, G. H. Bamboo: Making preemptible instances resilient for affordable training of large {DNNs}. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pp. 497–513, 2023.

Tolpegin, V., Truex, S., Gursoy, M. E., and Liu, L. 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*, pp. 480–501. Springer, 2020.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Wang, H. and Guo, K. Byzantine fault tolerant algorithm based on vote. In *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 190–196, 2019. doi: 10.1109/CyberC.2019.00041.

Wang, J., Lu, Y., Yuan, B., Chen, B., Liang, P., De Sa, C., Re, C., and Zhang, C. Cocktailsgd: Fine-tuning foundation models over 500mbps networks. In *International Conference on Machine Learning*, pp. 36058–36076. PMLR, 2023.

Workshop, B., Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

Xiong, H., Wang, S., Zhu, Y., Zhao, Z., Liu, Y., Wang, Q., and Shen, D. Doctorglm: Fine-tuning your chinese doctor is not a herculean task. *arXiv preprint arXiv:2304.01097*, 2023.

Yang, B., Zhang, J., Li, J., Ré, C., Aberger, C., and De Sa, C. Pipemare: Asynchronous pipeline parallel dnn training. *Proceedings of Machine Learning and Systems*, 3:269–296, 2021.

Yuan, B., He, Y., Davis, J., Zhang, T., Dao, T., Chen, B., Liang, P. S., Re, C., and Zhang, C. Decentralized training of foundation models in heterogeneous environments. *Advances in Neural Information Processing Systems*, 35: 25464–25477, 2022.

Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., and Liu, Y. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20)*, pp. 493–506, 2020.

Zhang, S., Choromanska, A. E., and LeCun, Y. Deep learning with elastic averaging sgd. *Advances in neural information processing systems*, 28, 2015.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Zhao, B., Mopuri, K. R., and Bilen, H. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

Zhu, B., Wang, L., Pang, Q., Wang, S., Jiao, J., Song, D., and Jordan, M. I. Byzantine-robust federated learning with optimal statistical rates. In *International Conference on Artificial Intelligence and Statistics*, pp. 3151–3178. PMLR, 2023.

Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.