Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Adversarial attack detection in resource-constrained environments: A stable and sequential federated learning architecture with TinyLlama-1.1B

Sevim Ceylan Böcekçi

sevimbocekci@marun.edu.tr

Computer Engineering

Kazım Yıldız

Faculty of Technology

Research Article

Keywords: Federated learning(FL), Large language models(LLM), Full fine-tuning, Mutex mechanism, Adversarial attacks

Posted Date: February 19th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8613070/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Adversarial attack detection in resource-constrained environments: A stable and sequential federated learning architecture with TinyLlama-1.1B

Sevim Ceylan Böcekçi[1*] and Kazım Yıldız[2]

[1*]Institute of Pure and Applied Sciences, Computer Engineering, Marmara University, İstanbul, 34722, Turkey.
[2]Computer Engineering, Faculty of Technology, Marmara University, İstanbul, 34854, Turkey.

*Corresponding author(s). E-mail(s): sevimbocekci@marun.edu.tr;
Contributing authors: kazim.yildiz@marmara.edu.tr;

Abstract

Large Language Models (LLMs) face training challenges on resource-constrained devices, and the performance losses caused by compression methods necessitate a 'full fine-tuning' approach. In this study, a Mutex-based architecture is proposed for the full fine-tuning of the TinyLlama-1.1B model in a federated learning environment. The proposed method prevents Out of Memory (OOM) errors by queuing GPU access while minimizing system load and providing an efficient training process through 'Incremental Averaging' and FP16 optimization on the server side. In experiments conducted on 5 clients using the TCAB dataset, the conventional federated learning method failed due to memory constraints, while the proposed method completed training with a 100 % success rate. As a result of the training, the model's accuracy in detecting adversarial attacks increased from 60.84% to 99.02%, and the balanced distribution of Precision and Recall values proved that the model did not develop bias. Additionally, FP16 optimization on the server side resulted in a 3.2 GB memory savings and reduced server computation costs. The findings reveal that despite increasing the total training time (latency), the proposed architecture enables large language models to be trained securely and stably with full fine-tuning, even on resource-constrained edge devices.

Keywords: Federated learning(FL), Large language models(LLM), Full fine-tuning, Mutex mechanism, Adversarial attacks

1 Introduction

Before the introduction of the Transformer architecture, which represents an architectural revolution forming the foundation of Large Language Models (LLMs), the dominant approaches in natural language processing were Recurrent Neural Networks (RNNs) and their advanced variant, Long Short-Term Memory Networks (LSTMs) [1]. These models treat language as a temporal sequence and process the input from left to right, word by word, to produce a contextual representation. Due to the long-term memory problem, they may forget the beginning of the sentence by the time they reach the end [2].

Today, Large Language Models (LLMs), which are becoming increasingly intelligent, represent the latest technological advancements in artificial intelligence (AI) and natural language processing (NLP). They are the most advanced systems capable of understanding and generating human language [3].With parameters that can be expressed in the billions, they can create new information based on patterns and structures learned from existing data [4].

High-quality and large amounts of data are important factors in LLM performance [5]. Qualified data not only increases the accuracy of the model; it also enables smaller models to perform tasks that massive models can do [6]. The "organic" or "natural" data that models are trained on is obtained from real human interactions and productions on the internet [7–9]. More specific and often restricted data is used to adapt models to a specific task (e.g., medical diagnosis, legal advice, or code generation) [10–15]. The use and sharing of this data is restricted for various legal and ethical reasons. Regulations such as GDPR (Europe) and HIPAA (US) restrict the collection and processing of personal data, preventing data from being collected on a central server [16]. Due to these restrictions, Federated Learning methods, where the data remains where it is located (on the client device or within the organization) rather than training the models directly in a central location, are preferred [17]. In this method, raw data never leaves the source device; instead, only model updates (parameters or gradients) are shared with a central server [10].

Due to the enormous size of LLMs, training the entire model on clients and sending it to the server is difficult [18]. Training LLMs with billions of parameters requires high memory and processing power. The floating-point operations (FLOPs) required during the forward and backward propagation processes in model training increase exponentially as the number of model parameters increases. This extends training time and leads to excessive energy consumption in battery-powered devices [19]. Furthermore, the constant transfer of large model parameters strains network bandwidth [20]. To overcome these negative effects, Parameter-Efficient Fine-Tuning - PEFT [21, 22], LoRA (Low-Rank Adaptation) [18], Model Compression and Transformation (Quantization, Pruning), Layer Skipping/Freezing [21] methods can be used [23, 24].

Compared to full fine-tuning, the LoRA method can learn new information more slowly in complex target domains such as coding and mathematics, and may lack sufficient reasoning ability when dealing with data it has never seen before [25]. Particularly in Continued Pretraining (CPT) scenarios, i.e., training with billions of unstructured tokens, applying full fine-tuning on the language model yields more accurate results than LoRA and is more data-efficient [26].

Full fine-tuning methods provide higher reward scores and accuracy rates compared to quantization methods in mathematical comparison and reinforcement learning (RL) scenarios. For example, in the Qwen3-8B model, full-precision GRPO training achieved a reward score of 0.594, while the use of 8-bit quantization (STE) during training reduced this score to 0.496 [27]. Performance degradation occurs as the number of bits used in compression decreases, especially in challenging reasoning tests [28]. Furthermore, LLM models that employ compression methods generally exhibit reduced robustness against adversarial attacks, and smaller models are even more vulnerable to such attacks [29]. Compressed models can be up to 285% more susceptible to adversarial attacks [30].

Compared to full fine-tuning, the frozen layer method preserves the old information in the frozen layer, making it difficult for the model to integrate new information [31]. As the task in model training becomes more complex, it is observed that frozen parameters are insufficient to represent new concepts in the model [32]. As the number of model parameters decreases, a 10-20% performance loss may occur in tasks requiring complex reasoning due to insufficient expressive power in frozen layers [33].

The approaches mentioned above, which aim to reduce the size of large language models, may result in insufficient expressive power within the model. Full fine-tuning training is more successful in out-of-domain generalization, and the relationship between terms is established more deeply. If an LLM is to be trained in technical fields that require intensive logic, such as health, law, mathematics, and coding, the model must be trained using full fine-tuning [34–36].

In this study, the objective is to train the large language model using full fine-tuning across all clients in the federated learning architecture. To prevent the out-of-memory (OOM) error that occurs when clients start training simultaneously in a 5-client federated learning architecture, a GPU locking mechanism was employed. Client access to the GPU is provided sequentially, and after sending their training results to the server in their own rounds, VRAM cleanup is performed using the garbage collector method. To solve the bottleneck problem that may occur when sending the global parameters obtained on the server to the clients, the lock mechanism was also used to ensure the sequential delivery of the global model to the clients. To eliminate read and write conflicts when clients save the global model they receive to disk, a mutual exclusion technique is used. The TinyLlama model, selected as the large language model, was trained using the TCAB dataset consisting of adversarial attacks. In the federated learning conducted over 3 rounds, the model was fully fine-tuned on adversarial attacks across all clients. Adversarial attack prediction was also performed on the global model obtained from the parameters received from the clients. The main objective of the study is to propose a software solution to the memory shortage

problem that arises when training a large language model with full fine-tuning on all clients in a federated learning environment.

The contributions of this study to the literature are as follows:

- Using the comprehensive TCAB dataset created with the TextAttack and OpenAttack libraries, a novel training scenario is introduced to the federated learning literature. A full fine-tuning approach is adopted, where not only specific layers but all parameters of the model are trained; thus, the effectiveness of distributed learning and the model's generalization success on complex datasets containing attack variations are demonstrated.
- To counter the instantaneous resource contention caused by clients starting concurrent training in conventional federated learning architectures, a time-ordered locking mechanism controlling GPU access has been developed. This approach minimizes peak memory usage by spreading the training load over time and prevents OOM (Out of Memory) errors even in hardware-constrained environments, enabling large-scale models to be trained with full fine-tuning.
- In the federated learning process, a memory cleanup routine that activates the Garbage Collector mechanism at the end of each iteration has been implemented. This increases VRAM usage efficiency and provides a resilient architecture against OOM (Out of Memory) errors, which are frequently encountered in long-term training.
- Network bottlenecks frequently encountered in client-server communication during federated learning processes and I/O (input/output) conflicts arising from concurrent disk access have been eliminated through the proposed sequential model distribution and Mutual Exclusion protocols. This approach ensures data integrity during the distribution of global parameters, enabling the training process to continue seamlessly and reliably.
- In contrast to Parameter Efficient Fine-Tuning (PEFT/LoRA) approaches commonly found in federated learning literature, this study focuses on enabling full fine-tuning of the TinyLlama model. This approach aims to enable the model to learn more deeply from the heterogeneous data and complex patterns in the federated network by updating not only specific layers but the entire weight space.
- The models used and the results obtained in this study will form a basis that will guide future research on the end-to-end training and optimization of large-scale models on hardware-constrained edge devices.

Chapter 2 provides a detailed review of similar studies in the literature and the methods used. Chapter 3 is the Materials and Methods section. This chapter describes in detail the TCAB dataset, the TinyLlama large language model, the federated learning architecture, the AdamW optimization technique, and resource management. In Chapter 4, titled Experimental Environment and Experimental Results, detailed information about the methods, parameters, and results used to train the large language model with the TCAB dataset using full fine-tuning in the experimental environment created with the federated learning architecture is presented, supported by tables and

graphs. The final part of our study, Conclusions and Future Work, discusses the findings obtained in Chapter 4 and provides recommendations for future research based on the summarized results of the study.

2 Related work

In the study seeking to answer the question "How can full training be performed on large language models despite hardware limitations?", model parameters, gradients, and optimizer states were distributed across GPUs. The proposed method has formed the basis for today's PyTorch Fully Sharded Data Parallel technology [37].

They conducted a performance comparison using LORA and P-Tuning-v2 optimization techniques on the ChatGLM-6B language model in their FATE-LLM model, developed for the industrial application of LLM. They concluded that LORA optimization is more cost-effective, with 0.058% for LORA and 0.475% for P-Tuning-v2 in the language model they trained with the AdvertiseGen dataset [38].

They used the LLaMA-2-7B LLM in their fine-tuning model, which they named FedBiOT. They conducted work on federated architectures with 3 clients using the GSM-8K dataset for Math Problem Solving, 9 clients using the Rosetta dataset for Code Generation, and 8 clients using the Dolly dataset for Question Answering. The proposed FedBiot method has a lower computational load compared to offsite-tuning and FedOT methods due to having the fewest trainable parameters [39].

In their work on natural language understanding and text generation, they propose a framework called FL-GLM. In this work, where the input and output blocks are placed on the client side, key encryption is performed between the client and server. Similar results to the chatGLM model were obtained using optimization methods such as client-batching and server-hierarchical [40].

The proposed edge-assisted split federated learning framework aims to reduce the memory load on mobile devices and servers. During training, the processing order of edge devices on the server side is optimized. When comparing the proposed framework with split federated learning, a 79% reduction in memory and a 6% reduction in training time were observed [41].

To optimize communication and computational efficiency, clients use different LORA levels during the fine-tuning process within the proposed HLORA framework. In a study conducted with 100 clients on the RoBERTa-large language model, the Microsoft Research Paraphrase Corpus (MRPC), the Quora Question Pairs (QQP), and the Recognizing Textual Entailment (RTE) datasets were used in the model's training [42].

Binary text classification was performed on a dataset created using real-world news reports in English, Portuguese, Spanish, and Hindi. The performance of the FedAvg, FedAvgM, FedYogi, FedAdam, and FedAdagrad FL algorithms in the federated learning section was compared. The Distilbert model achieved an F1 Score of 0.7789 in the FedYogi algorithm, while the BERT language model achieved an F1 Score of 0.8015 in the FedAvg algorithm [43].

The study, which stems from the idea that heterogeneous table data from clients hinders the aggregation process in federated learning, proposes the FedLLM-Align

framework as a solution to this problem. Using the Framingham dataset, which has been converted into text, coronary heart disease predictions are made in the AvgFed architecture. In the study conducted with 3, 5, and 10 clients on the Framingham Heart Study dataset, an improvement of up to +0.25 in the F1 score and a 65% reduction in communication costs were achieved [44].

Due to the large amounts of medical image and report data distributed across endpoints, the FEDMRG framework was proposed in the study using the FEDERATED learning architecture. All parameters of the low-parameter Conventional Language Model used in the study were trained, while some parameters of the Llama2-7B-Chat-HF model, which contains more parameters, were frozen to reduce communication costs. With the proposed framework, clinically accurate reports were obtained from visual health data with the help of LLM [45].

The study, which utilizes the OPT-1.3B, Bloom-3B, and Llama 2-7B language models, fine-tunes the models using the wikitext-2-raw-v1 dataset and proposes the TiTANIC LM framework. The server distributes the LLM model to selected clients. The proposed model has demonstrated superiority over conventional learning in terms of bandwidth and local computational costs [46].

The study proposing the FeDeRa framework developed the LORA method using the RoBERTa-base and DeBERTa language models. The 20Newsgroups and SemEval-2010Task8 datasets were selected for text classification, the WNUT2017 and PLONER datasets for entity recognition, and the SQuADv1 and MRQA datasets for question-answering systems. In the study, LORA does not randomly assign low rank matrix values; singular value decomposition is applied to pre-trained weights. With a trainable 1% parameter, over 90% training time savings are achieved [47].

In the federated architecture established with 100 clients, the prompts of 10 randomly selected clients are combined on the server. In the study, where each layer of the GPT2LARGE and LLaMA 3B models is optimized with P-tuning v2, 10 different data sets are used to train the models. Compared to studies evaluated within the proposed Fed-PepTAO framework, accuracy is up to 60.8% higher, and efficiency is up to 97.59% higher [48].

As a result, our method has been shaped by taking advantage of the strengths of the studies in the literature, while also considering the existing shortcomings. Therefore, the chosen method is based not only on technical aspects, but also on the motivation to overcome the limitations observed in the literature.

Table 1 summarizes the language models used in previous studies conducted with LLM in the federated learning architecture, the methods used in model training, the data sets, and the metrics related to success.

3 Methodology

The primary objective of this study is to enable the use of a fully fine-tuned model to detect adversarial attacks targeting large language models within a federated learning architecture that protects user data privacy. It aims to develop a system that addresses the potential memory shortage issue during the training of the fully fine-tuned model by training clients in series and using a mutual exclusion method for

Table 1 Previous works on LLMs in federated learning architectures

Ref	LLM Name	Year	Method	Dataset	Metrics
[46]	Llama 2.7B OPT-1.3B Bloom-3B	2024	Full Fine Tunning	wikitext-2-raw-v1	Perplexity= 2.54
[37]	Turing-NLG	2020	ZeRO Optimization	Common Crawl Wikipedia Books RealNews & OpenWebText	Petaflops= 15
[38]	ChatGLM-6B	2023	Lora P-Tuning-v2	AdvertiseGen	Lora Efficiency=0.058 % P-Tuning-v2 Efficiency =0.475%
[42]	RoBERTa	2025	HLORA	MRPC, QQP, RTE	Accuracy=86.1 %
[44]	DistilBERT ALBERT RoBERTa ClinicalBERT	2025	FedLLM-A-lign	Framingham Heart Study	F1=+0.25 Communication Cost= - %65
[45]	Llama2- 7B Chat-HF	2025	Layer Freezing	MIMIC-CXR IU X-Ray	BLEU Skor= 12.68 ROUGE-L Skor= 27.55
[49]	DataJuicer-1.3B	2024	FlexLoRA	Natural Instructions Dolly-15K	Rouge-L Skor= + %3.1

resource sharing. The TCAB (Text Classification Attack Benchmark) dataset was used in the development of the system. The training of the TinyLlama model was performed in a federated learning architecture consisting of 5 clients and 3 rounds.

In our study, a six-step methodology was followed to detect adversarial attacks on the TinyLlama large language model, which was trained using full fine-tuning in a federated learning (FL) architecture.

1. Application of pre-processing steps to the dataset
2. Establishment of the federated learning framework
3. Processor locking for sequential training of clients, creation of mutual exclusion mechanisms for sequential resource access, memory cleanup after performing model training on clients and sending parameters to the server.
4. Creating a sequential transmission mechanism to prevent bottlenecks during the transmission of the global model created on the server to clients
5. Testing the model as a full fine-tuning and
6. Evaluation of clients' training results

The general workflow of the proposed federated learning framework is shown in Figure 1.

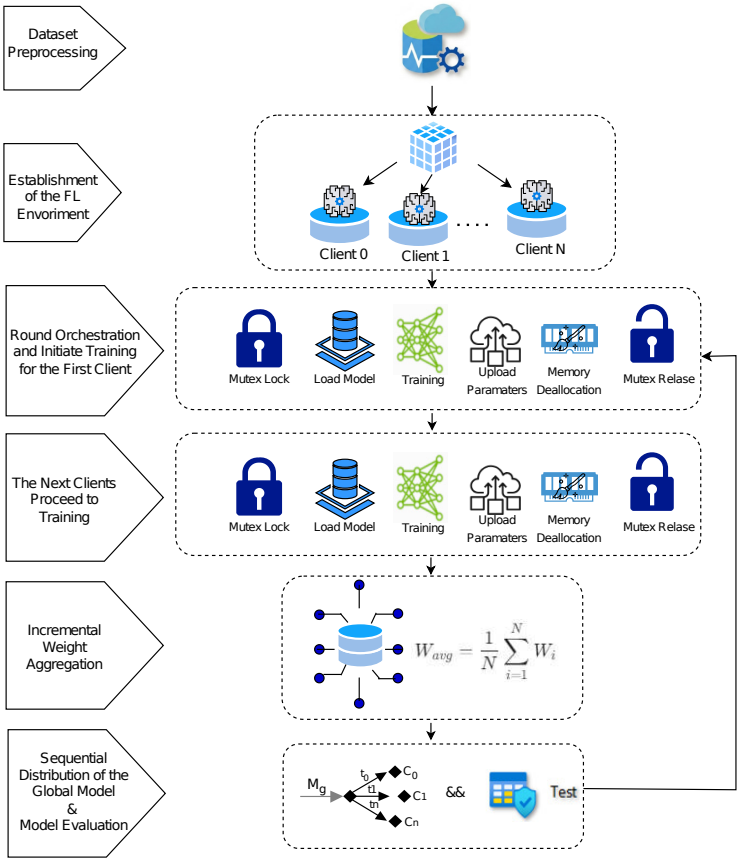

Fig. 1 Flow diagram of the proposed algorithm

3.1 TinyLlama large language model

The TinyLlama large language model was obtained by processing approximately 950 billion tokens from the combined SlimPajama and StarCoder datasets using the Llama tokenizer over three periods. The slimPajama dataset used in the model was obtained by subjecting the RedPajama corpus, containing over 1.2 billion tokens, to comprehensive filtering after additional cleaning and deduplication processes. This dataset is a high-quality text collection specifically created for training large language models [50]. Another dataset used in the model, the StarCoder dataset, contains code data in 86 programming languages, GitHub issues, and text-code pairs composed of natural languages [51]. The datasets used in training the TinyLlama large language model, the data sources used in the dataset creation phase, and the preprocessing steps the data

underwent are shown in Figure 2. TinyLlama, consisting of 22 layers, is a language model with 1.1 billion parameters and a hidden size of 2048 [50].

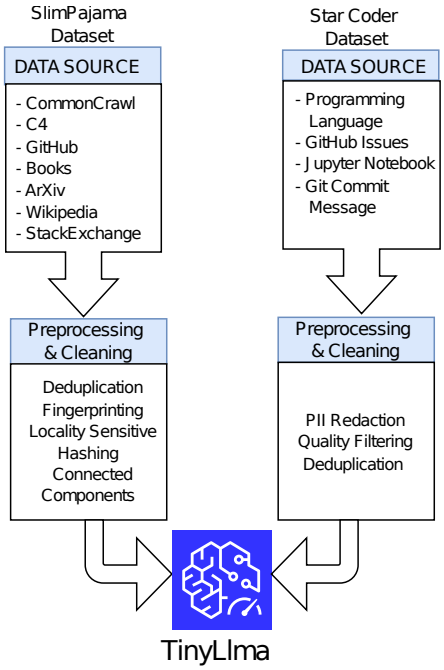

Fig. 2 Datasets used for TinyLlama model development

3.2 Federated learning

In recent years, deep neural networks have become widely used and rapidly developed due to their high performance in many areas [46–48, 52]. However, high performance in deep neural networks depends on large amounts and diversity of input data. The amount of space required to collect and store large amounts of data brings with it certain challenges alongside its high performance. Furthermore, transferring data to another location raises data security issues. General data protection policies emphasize the importance of maintaining confidentiality in data sharing between different organizations [53]. The federated learning model, a collaboration-based and effective paradigm that preserves confidentiality between different parties, can be seen as a solution to the problems described above. While raw data is collected from edge devices in traditional centralized learning, federated learning involves sharing the weights of trained models among clients. Figure 3 provides information about how federated learning works. As shown in the figure, no data sharing takes place in the federated learning architecture, and privacy can be preserved to a certain extent [54]. Locally trained model parameters on clients are shared with the central server. Depending on

the selected algorithm, all models are evaluated on the central server, and the resulting global model is then distributed to each client. This increases both data privacy and model accuracy [55]. In recent years, the federated learning architecture has been widely used in areas where data privacy is important, such as banking and healthcare [56].

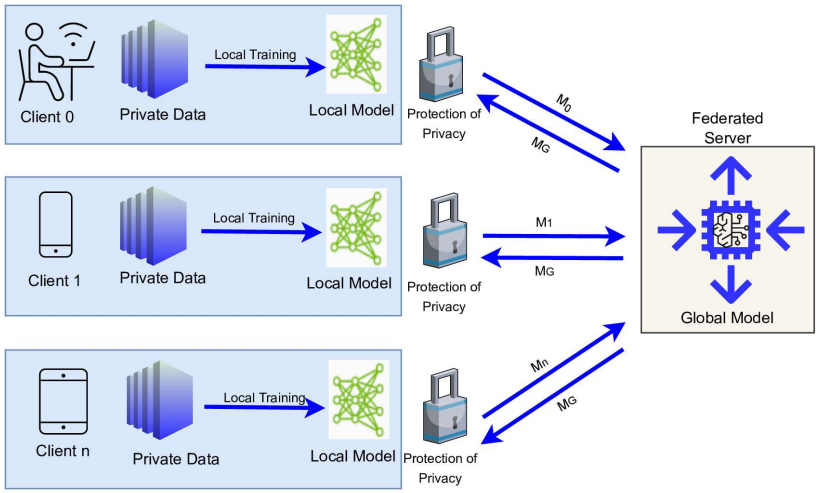

Fig. 3 Operation of the federated learning architecture

3.3 Dataset

The TCAB (Text Classification Attack Benchmark) dataset was developed for the analysis, understanding, detection, and labeling of adversarial attacks targeting text classifiers. The dataset uses a total of 6 different datasets, including climate change tweets for sentiment analysis, IMDB and SST-2 movie reviews; and Wikipedia toxic comments, hate speech, and Civil Comments for misuse detection [60]. The original and attack sentences in the TCAB dataset and the attack types are shown in Table 2. The dataset used for the attacks, which utilizes the TextAttack and OpenAttack libraries, contains 12 different types of attacks. The distribution of attacks in the TCAB dataset train file is shown in Figure 4(a) , while the distribution of attacks in the validation file is shown in Figure 4(b). The dataset contains over 1.5 million successful attack examples against text classification models [59].

3.4 Resource management

Mutual exclusion is a fundamental class of algorithms that synchronizes access to shared resources by concurrently running processes. Based on the theoretical framework proposed by Dijkstra, GPU memory resources and the global model file, which multiple clients attempt to access concurrently, are treated as "Critical Sections" [61].

Table 2 Examples of original and attack sentences from the TCAB dataset

Attack Type	Orjinal Text / Perturbed Text
Hotflip	@KelsieBelsi: You're not a man if you refer to every girl as a bitch
	@ kelsiebelsi : you ' re not a boy without you refer to any maiden as a blackguard
Viper	@KelsieBelsi: You're not a man if you refer to every girl as a bitch
	@KelsieBelŝi: Yôŭ're not a mân if yôu rêf̀er to eṽery girl as a biẗcĥ
Genetic	@KelsieBelsi: You're not a man if you refer to every girl as a bitch
	@KelsieBelŝi: Yôŭ're not a mân if yôu rêf̀er to eṽery girl as a biẗcĥ
Textbugger	With my Bestf Shyanna Pen and ma main niggah Ronnie A. Jackson III 󾮟󾓤 Movie bound after we pick up the homies
	With my Bestf Syhanna P en and ma maij niggah Rnnoie A. Jacksn II I &# 104359 ;# 104166 ; Moie bouhd agter we pic k up te homies
Textfooler	RT @guccii1017: wtf was drake asking us to pull over so he can get my autograph bitch I’m scheelp fuck you drake http://t.co/LzL0JlhdvK
	RT @guccii1017: wtf was drake asking us to pull over so he can have my autograph floozy I’meter scheelp sodding you drake http://t.co/LzL0JlhdvK
Deepwordbug	RT @guccii1017: wtf was drake asking us to pull over so he can get my autograph bitch I’m scheelp fuck you drake http://t.co/LzL0JlhdvK
	RT @guccii1017: wtf was drake asking us to pull over so he can egt my autograph bicth I’m csheelp fcuk you drak http://t.cCo/LzL0JlhdvK

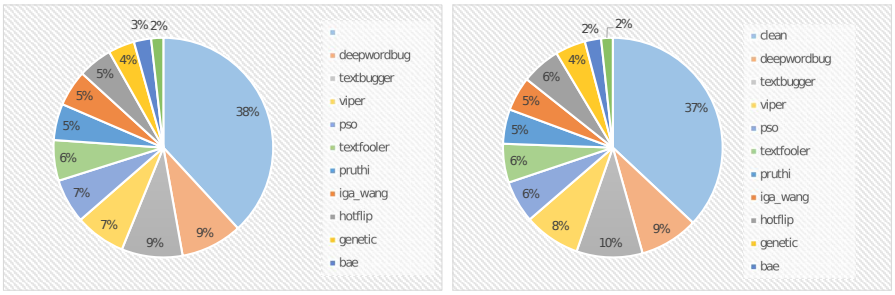

Fig. 4 Distribution of attack types (a) TCAB training dataset (b) TCAB validation dataset

To prevent data inconsistencies and conflicts (deadlock/race condition), a locking mechanism (file-based locking) with atomic operation capability is implemented on the file system. This mechanism ensures that access to critical sections is performed by only one process at any given time, thereby maintaining system stability [65, 66]. The access states of processes A and B to the critical section at time T, where one process is blocked when the other enters the critical section, are shown in Figure 5.

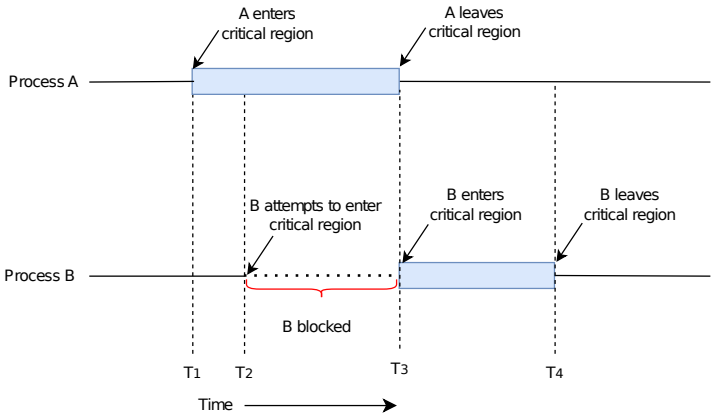

Fig. 5 Mutual exclusion mechanism in resource access [62].

3.5 AdamW optimization

ADAM optimization is a popular optimization method used in deep neural network training. AdamW, a specialized variation of Adam, separates the weight decay process from gradient-based updates. This results in improved generalization performance [67]. AdamW optimization stores two fundamental statistics in memory for each model parameter: Momentum (m) and Variance (v). During the training of a model where each parameter is represented by 2 bytes, the memory cost of the AdamW optimization algorithm, which is only related to the optimizer state, creates an additional load per parameter. In this context, a total of 4 bytes of additional memory per parameter is required: 2 bytes for the first moment (momentum) estimate and 2 bytes for the second moment (variance) estimate [68].

4 Evaluation and experimental results

In this study, a PC with an 11th Gen Intel® Core™ i9-11900F @ 2.50 GHz processor, 32.0 GB RAM, and NVIDIA GeForce RTX 3070 (8 GB) graphics card was used to enable large language models to be trained and tested with full fine-tuning against adversarial attacks in a federated learning architecture. To prevent version conflicts and provide an isolated working environment, the Anaconda Navigator platform was chosen for the project, and Python 3.10.19 was used as the programming language. The numpy, pandas, and sklearn libraries were included in the experimental environment.

4.1 Preprocessing of the dataset

In the federated learning architecture, the first 5000 samples were selected from the TCAB dataset to train the Large Language Model (LLM) as full fine-tuning against

adversarial attacks. The distribution of classes labeled as '1' (successful attack) and '0' (clear) in the train and validation sets of the dataset is shown in Figure 6.

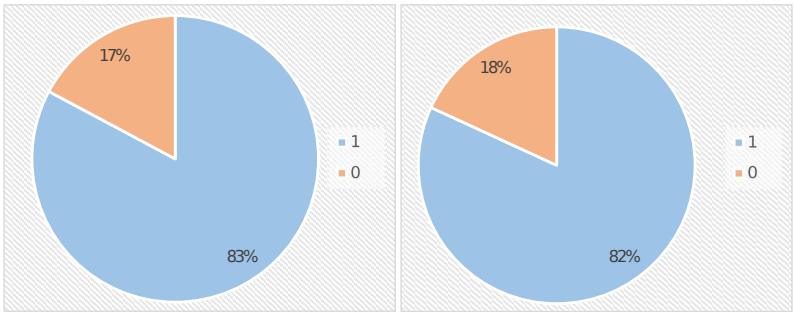

Fig. 6 Distribution of class labels (a) Training sets. (b) Validation sets.

To address the class imbalance problem observed in the dataset and prevent biased learning of the model, a balanced subsample was created by randomly selecting 2500 samples from each class labeled as '0' and '1'.

Data preprocessing steps are a critical stage for improving the model's efficiency [69]. Adversarial attacks typically involve syntactic disruptions and special character manipulations. Special symbols within the prompt are part of the attack. Unlike traditional Natural Language Processing (NLP) processes, which would eliminate attack attributes, data preprocessing steps such as 'stop-words removal', cleaning words that do not convey meaning on their own, removing invisible characters, and punctuation cleaning were not applied in this study [64]. To preserve the attack patterns, the dataset was processed in its raw form to fit the tokenizer structure of the TinyLlama model.

The preprocessing steps performed in the application are as follows.

- Deleting rows that do not contain data
- Extracting the features to be used in the study
- Converting the success and clear values in the status feature to binary
- Adding idx for each data row

The flow diagram showing the order in which the preprocessing steps are applied is presented in Figure 7.

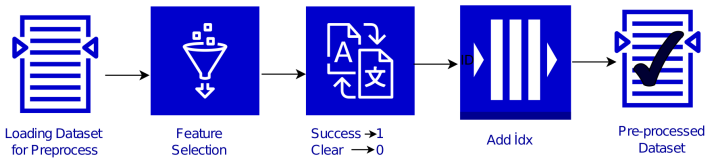

Fig. 7 Dataset preprocessing steps

4.2 Evaluation and analysis

Mathematically, the memory requirement of the Large Language Model (LLM) used in our study can be defined as follows:

$$M_{model} \approx \frac{|\Theta| * \mathcal{Q}}{8 * 10^9} GB \tag{1}$$

For the TinyLlama-1.1B model, static memory usage is determined to be approximately 2.2 GB in the $|\Theta| = 1.1 \times 10^9$ and $\mathcal{Q} = 16$ (FP16) configuration. where:

M_{model}: Model weights memory footprint (GB).
$|\Theta|$: Total number of trainable parameters in the model.
$\mathcal{Q}$: Bit precision of weights (Bit-width / Quantization level).

Considering the hardware constraints of Federated Learning clients, the 'Pure FP16' training strategy was adopted instead of 'Mixed Precision' during the training process. In this configuration, AdamW optimization disabled FP32 master weights, and memory management was supported by garbage collection mechanisms. The peak VRAM usage during training of the 1.1 billion parameter TinyLlama model has been reduced from 17.6 GB to 6.6 GB. The low resource consumption on the client side can be represented by the following mathematical equation:

$$M_{\text{PureFP16}} \approx \frac{|\Theta| \times (P_{\text{weight}} + P_{\text{grad}} + P_{\text{opt}})}{10^9} + M_{\text{act}} \tag{2}$$

where:

$|\Theta|$: Number of parameters (1.1×10^9).
P_{weight}: Model of weights – FP16 (2 Byte).
P_{grad}: Gradients – FP16 (2 Byte).
P_{opt}: Optimizer states ($\sim$ 2 Byte).

$$M \approx \frac{1.1 \times (2 + 2 + 2)}{1} = 1.1 \times 6 = 6.6\,\text{GB}$$

Each client in the federated learning architecture requires 6.6 GB of VRAM capacity. In the classic federated learning architecture, each client loads the model it will train into VRAM and begins training in parallel. Figure 8 shows the system behavior when 5 Federated Learning clients attempt to load the TinyLlama-1.1B model into GPU memory simultaneously. In the first 900-second time interval marked as 'Memory Thrashing Zone' in the figure, the total memory demand of the 5 concurrently launched clients exceeded the physical RAM capacity. The operating system intensively engaged the disk-based virtual memory (paging/swap) mechanism to keep the processes alive. With the virtual memory limits (Commit Limit) also being pushed, processes were randomly terminated with OOM failures to maintain system stability. Each crash that occurred gradually freed up occupied VRAM and RAM resources (Resource Reclaiming), alleviating the pressure on the system and allowing the necessary memory space to be freed up for the last surviving process to run. In a classic

federated learning architecture, if clients load the model they are training into memory simultaneously, memory limits are exceeded. This leads to system deadlock or inefficient resource usage (thrashing).

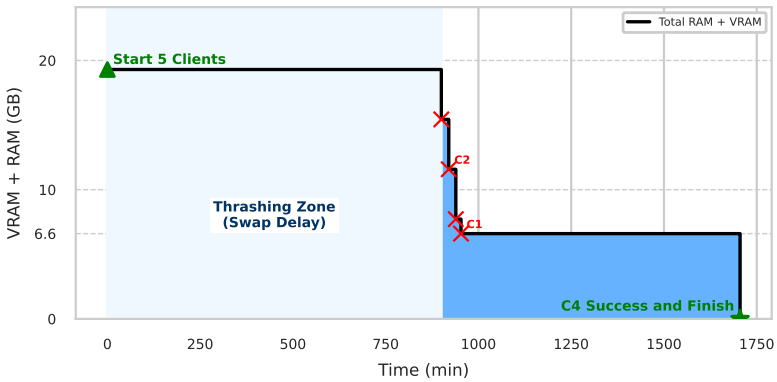

Fig. 8 System behavior during the concurrent loading of the TinyLlama-1.1B model across 5 clients.

In the proposed mutex-based solution, only one client GPU is allocated time per period, and it trains its model by loading it onto VRAM. Upon completing training, the client sends the parameters of the trained model to the server. Once the parameter upload process to the server is complete, an aggressive garbage collector process is performed on the VRAM, the mutex lock is released, and the training sequence passes to the next client. Figure 9 shows the sequence of operations for the proposed mutex-locked solution.

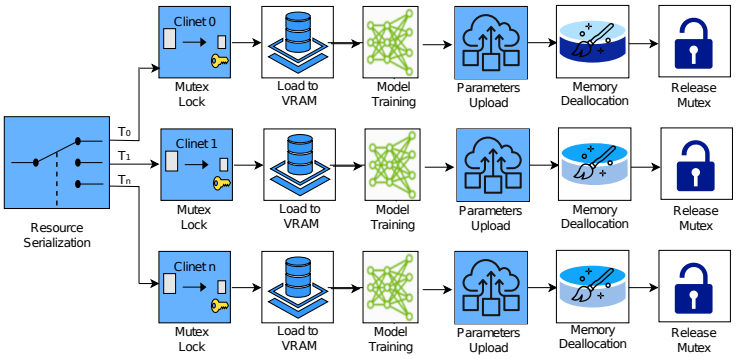

Fig. 9 Execution flow of the proposed mutex-based approach

The impact of the developed software-based GPU lock mechanism on system resources is shown in Figure 10. Unlike the uncontrolled classical federated learning scenario, in this approach, clients accessed GPU resources sequentially. During

the training of each client, VRAM usage was at the 6.6 GB level required for the TinyLlama-1.1B model, and hardware limits were not exceeded. The client that took the GPU time could not complete its training and release the lock, and the subsequent client could not obtain GPU time and did not enter the memory. This solution eliminates Race Condition and resource contention risks by 100%. Thanks to the 'Garbage Collector' mechanism that comes into play during the transition processes between clients, system resources are completely released with each client change; thus, 'Out of Memory' (OOM) errors and Thrashing situations that reduce system performance are prevented. Although the total processing time (Latency) increased, the system's throughput was guaranteed, and the full fine-tuning training process was completed with a 100% success rate.

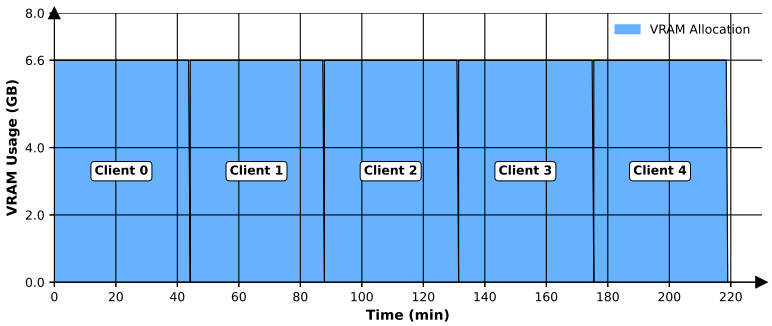

Fig. 10 System behavior under the proposed mutex-based method

Figure 11 Shows the analysis of client-based training loss variation and model convergence in rounds in the federated learning architecture. The training process, which began with high variance in round 0 (blue) of federated learning, stabilized rapidly in Round 1 (orange) and Round 2 (green) thanks to the global weights received from the server. In particular, the fact that the error rate (loss) of all clients approached zero at the end of Round 2 proves the learning success of the proposed architecture. The sudden spikes observed in Client 2 and Client 4 during Round 0 indicate that the data within the batch is challenging (adversarial) for the model or that the learning rate remained high at that moment. However, since the model learned the attacks by Round 2, these spikes have almost completely disappeared. By the end of Round 2, all Clients' Loss values have dropped to values like 0.03 and 0.002. This means the model has now learned that dataset very well.

The accuracy, loss value, F1 score, precision, and recall metrics for each round in the proposed federated learning architecture are provided in Table 3. The accuracy value, which was 60.84% in Round 0, increased to 98.88% in Round 1. This demonstrates that the federated learning structure (Global Aggregation) works extremely effectively and that the weights received from the clients are successfully combined. The loss value dropped from 0.6197 to 0.0463, proving that the model's error decreased by 92%. Transitioning from Round 1 to Round 2, the model's learning speed slowed down but continued (%98.88 -> %99.02). The loss value dropped further to 0.0251.

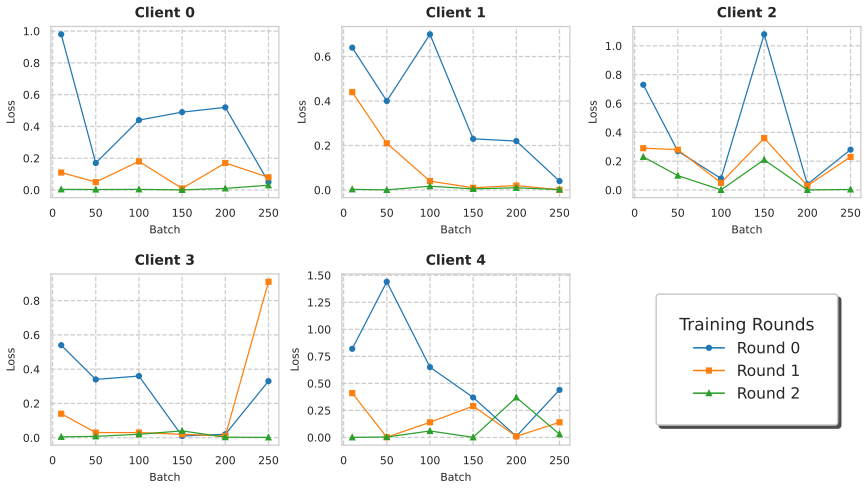

Fig. 11 Analysis of client-based training loss and model convergence across rounds in the proposed federated learning architecture

This indicates that the model had already "converged" at the end of Round 1 and increased its stability through fine-tuning in Round 2. The fact that the Precision (99.03%) and Recall (99.02%) values are very close to each other in Round 2, the final round of training, indicates that the model achieved balanced learning and did not develop any bias towards any class.

Table 3 Performance metrics of the proposed switching-based federated learning architecture

Round	Accuracy	Loss	F1-Score	Precision	Recall	Test Duration
Before Training	0.5220	1.2951	0.4464	0.5485	0.5220	01:48
Round 0	0.6084	0.6197	0.5383	0.7759	0.6084	01:49
Round 1	0.9888	0.0463	0.9888	0.9888	0.9888	01:48
Round 2	0.9902	0.0251	0.9902	0.9903	0.9902	01:48

In the proposed method, the 'Incremental Averaging' method has been chosen to minimize memory access cost and processing time during the calculation of the global model on the server side using parameters received from clients. The most distinctive feature of this architecture is that it completely eliminates the need for upcasting intermediate operations to 32-bit (FP32) precision during the aggregation process on the server side. Parameters transferred from clients are processed directly in their native 16-bit (FP16) precision without undergoing any format conversion. Thanks to this end-to-end 16-bit (end-to-end FP16) stream, where parameters are dynamically

divided by the number of clients (N=5) and included in the cumulative summation; minor precision losses that could occur given the distribution of model weights and the low number of N remain negligible compared to the gains in processor efficiency and RAM bandwidth.

The impact of the proposed FP16-based sequential summation architecture on system resources compared to the traditional FP32-based approach is presented in Table 4. Maintaining the cumulative sum at 16-bit precision instead of 32-bit precision resulted in a net memory savings of 3.2 GB across the system. This provided a 15.6% reduction in total system memory usage and prevented the server from entering an out-of-memory (OOM) state.

Additionally, halving the size of data transferred from memory to the processor has reduced the load on memory bandwidth. This has resulted in a performance improvement reflected in approximately a 10% decrease in CPU usage (39.3% $\rightarrow$ 35.4%) and a 7.4% increase in processing speed. The results confirm that the proposed "end-to-end 16-bit" architecture significantly increases computational and memory efficiency without compromising accuracy.

Table 4 Performance comparison of the conventional (FP32) and proposed (FP16) methods in terms of accuracy and resource utilization.

Performance Metric	Conventional Method (FP 32)	Proposed Method (FP 16)	Absolute Change	Improvement Rate
Average RAM Usage	$\sim$ 20.5 GB	$\sim$ 17.3 GB	$\sim$ 3.2	15.6%
Average CPU Load	39.3%	35.4%	-3.9 Points	9.9%
Average Processing Time	37.8 sn	35.0 sn	-2.8 s	7.4%
Accuracy	99.02	99.14	+0.12	0.12%

5 Conclusion

This study focuses on the use of a fully fine-tuned language model to detect adversarial attacks on large language models. Our goal is to develop a mutex-based asynchronous system where clients are trained sequentially to address the memory shortage problem that may arise during training of the fully fine-tuned model, where mutual exclusion is used for resource sharing, and where the 'Incremental Averaging' method is preferred during the calculation of the global model on the server side.

The study first determined that training the TinyLlama-1.1B model on each client as a full fine-tuning process requires 6.6 GB of VRAM capacity. The success rate of full fine-tuning training was examined when five clients simultaneously initiated the training process in a traditional federated learning architecture. When examining the graph data in Figure 7, the total memory demand exceeded the physical RAM capacity during the first 900 seconds marked as the 'Memory Thrashing Zone'. OOM errors occurred as the operating system intensively activated the disk-based virtual

memory (paging/swap) mechanism in order to continue client model training. This situation demonstrates that the traditional federated learning architecture leads to system deadlock and inefficient resource usage (thrashing) in resource-constrained environments.

However, in the Mutex-switched solution method shown in Figure 8, clients accessed GPU resources sequentially. As shown in Figure 9, VRAM usage during each client's training was kept constant at the hardware limit of 6.6 GB. Thanks to the developed software-based GPU locking mechanism, resource contention between clients was eliminated by 100%. The aggressive 'Garbage Collector' mechanism, which kicks in after a client completes its training, created a clean memory space for the next client and prevented 'Out of Memory' errors. Although the total processing time (Latency) increased, the system's throughput was guaranteed, and the training process was completed with a 100% success rate.

The learning success of the proposed architecture against adversarial attacks has been analyzed in Figure 10 based on the change in 'Training Loss' exhibited by clients throughout the training rounds. Loss values, which started with high variance in Round 0 of the training process, stabilized rapidly in Round 1 and Round 2 thanks to the global weights (Global Aggregation) received from the server at the end of each round.

When examining Table 3, which provides the performance metrics of the global model tested with the validation set at the end of each round, it can be seen that the accuracy value, which was 60.84% in Round 0, increased to 98.88% at the end of Round 1. The decrease in the loss value from 0.6197 to 0.0463 proves that the model error has decreased by 92%. In Round 2, the final round of training, the model reached an accuracy value of 99.02%, completing the convergence process. Furthermore, the fact that the Precision (99.03%) and Recall (99.02%) values are extremely close to each other indicates that the model achieved balanced learning and did not develop any bias towards any class.

During the calculation of the global model on the server side, an end-to-end 16-bit (End-to-end FP16) architecture was implemented using the 'Incremental Averaging' method, unlike addition operations performed with traditional FP32 (32-bit) precision. Parameters received sequentially as a result of client-side local training were processed directly in FP16 format. According to the comparative analysis results of the FP32 and FP16 formats presented in Table 4, maintaining the cumulative total on the server side at 16-bit precision has resulted in a net memory savings of 3.2 GB across the system. This has created a 15.6% reduction in total memory usage, preventing the server from entering a memory bottleneck. In terms of CPU efficiency, halving the data size reduced the pressure on memory bandwidth, resulting in an approximately 10% decrease in CPU usage and a 7.4% increase in processing speed.

In conclusion, all experimental findings demonstrate that the proposed Mutex-based federated learning architecture and end-to-end FP16 optimization maximize system stability and resource efficiency without compromising model accuracy. This study provides concrete evidence that large language models can be trained seamlessly, securely, and with high performance using a federated learning approach, even in resource-constrained environments.

The Mutex-based full fine-tuning federated learning architecture presented in this study offers an effective solution for overcoming memory bottlenecks and ensuring system stability in limited hardware resources. Although the current sequential access method guarantees system stability, it linearly increases the total processing time in scenarios where the number of clients increases. In this context, researchers aiming to advance the topic are advised to focus on optimization approaches that increase scalability and minimize training time while maintaining the stability advantages provided by the system.

Author contributions: S.C.B. conceptualized and designed the study, implemented the proposed methods, and conducted the experiments. K.Y. supervised the research, contributed to the theoretical modeling, and provided guidance on the structure and content of the article.

Funding: The authors have not disclosed any funding.

Data availability: No datasets were generated or analysed during the current study.

Declarations

Competing interests: The authors declare no competing interests.

Ethics approval and consent to participate: Since an open access dataset was used, ethics committee approval was not required.

References

[1] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[3] Nazir, A., Wang, Z.: A comprehensive survey of chatgpt: advancements, applications, prospects, and challenges. Meta-radiology **1**(2), 100022 (2023)

[4] Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., *et al.*: A survey on evaluation of large language models. ACM transactions on intelligent systems and technology **15**(3), 1–45 (2024)

[5] Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D.: Scaling laws for neural language models. arXiv preprint arXiv:2001.08361 (2020)

[6] Gunasekar, S., Zhang, Y., Aneja, J., Mendes, C.C.T., Del Giorno, A., Gopi, S., Javaheripi, M., Kauffmann, P., Rosa, G., Saarikivi, O., et al.: Textbooks are all

you need. arXiv preprint arXiv:2306.11644 (2023)

[7] Qin, Z., Dong, Q., Zhang, X., Dong, L., Huang, X., Yang, Z., Khademi, M., Zhang, D., Awadalla, H.H., Fung, Y.R., et al.: Scaling laws of synthetic data for language models. arXiv preprint arXiv:2503.19551 (2025)

[8] Held, W., Hall, D., Liang, P., Yang, D.: Relative scaling laws for llms. arXiv preprint arXiv:2510.24626 (2025)

[9] Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Las Casas, D., Hendricks, L.A., Welbl, J., Clark, A., et al.: Training compute-optimal large language models (2022). arXiv preprint arXiv:2203.15556 (2022)

[10] Xu, M., Cai, D., Wu, Y., Li, X., Wang, S.: Fwdllm: Efficient fedllm using forward gradient. arXiv preprint arXiv:2308.13894 (2023)

[11] Kuang, W., Qian, B., Li, Z., Chen, D., Gao, D., Pan, X., Xie, Y., Li, Y., Ding, B., Zhou, J.: Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 5260–5271 (2024)

[12] Wu, Y., Tian, C., Li, J., Sun, H., Tam, K., Zhou, Z., Liao, H., Guo, Z., Li, L., Xu, C.: A survey on federated fine-tuning of large language models. arXiv preprint arXiv:2503.12016 (2025)

[13] Shabani, N.: Harnessing Federated Learning for LLM Fine-Tuning: A Distributed Approach (2024)

[14] Jueal Mia, M., Hadi Amini, M.: Fedshield-llm: A secure and scalable federated fine-tuned large language model. arXiv e-prints, 2506 (2025)

[15] Chen, J., Cai, Z., Chen, W., Wang, W., Zheng, Z., Yu, P.S.: A federated adaptive large language model fine-tuning framework for software development. IEEE Transactions on Services Computing (2025)

[16] Kokash, N., Wang, L., Gillespie, T.H., Belloum, A., Grosso, P., Quinney, S., Li, L., Bono, B.: Ontology-and llm-based data harmonization for federated learning in healthcare. arXiv preprint arXiv:2505.20020 (2025)

[17] Che, H., Jin, H., Gu, Z., Lin, Y., Jin, C., Chen, H.: Llm-driven medical report generation via communication-efficient heterogeneous federated learning. IEEE Transactions on Medical Imaging (2025)

[18] Shabani, N.: Harnessing Federated Learning for LLM Fine-Tuning: A Distributed Approach (2024)

[19] Wu, Y., Tian, C., Li, J., Sun, H., Tam, K., Zhou, Z., Liao, H., Guo, Z., Li, L., Xu, C.: A survey on federated fine-tuning of large language models. arXiv preprint

arXiv:2503.12016 (2025)

[20] Xu, M., Cai, D., Wu, Y., Li, X., Wang, S.: Fwdllm: Efficient fedllm using forward gradient. arXiv preprint arXiv:2308.13894 (2023)

[21] Chen, J., Cai, Z., Chen, W., Wang, W., Zheng, Z., Yu, P.S.: A federated adaptive large language model fine-tuning framework for software development. IEEE Transactions on Services Computing (2025)

[22] Mia, M.J., Amini, M.H.: Fedshield-llm: A secure and scalable federated fine-tuned large language model. arXiv preprint arXiv:2506.05640 (2025)

[23] Fan, T., Kang, Y., Ma, G., Chen, W., Wei, W., Fan, L., Yang, Q.: Fate-llm: A industrial grade federated learning framework for large language models. arXiv preprint arXiv:2310.10049 (2023)

[24] Luo, W., Keung, J., Yang, B., Ye, H., Le Goues, C., Bissyande, T.F., Tian, H., Le, X.B.D.: When fine-tuning llms meets data privacy: An empirical study of federated learning in llm-based program repair. ACM Transactions on Software Engineering and Methodology (2024)

[25] Bohnet, B., Dangovski, R., Swersky, K., Moore, S., Chaudhry, A., Kenealy, K., Fiedel, N.: A comparative analysis of llm adaptation: Sft, lora, and icl in data-scarce scenarios. arXiv preprint arXiv:2511.00130 (2025)

[26] Biderman, D., Portes, J., Ortiz, J.J.G., Paul, M., Greengard, P., Jennings, C., King, D., Havens, S., Chiley, V., Frankle, J., et al.: Lora learns less and forgets less. arXiv preprint arXiv:2405.09673 (2024)

[27] Kumar, M., Xu, Z., Wang, X., Webb, T.: The impact of quantization on large reasoning model reinforcement learning. arXiv preprint arXiv:2511.15694 (2025)

[28] Zhang, N., Zhang, Y., Mitra, P., Zhang, R.: When reasoning meets compression: Benchmarking compressed large reasoning models on complex reasoning tasks. arXiv preprint arXiv:2504.02010 (2025)

[29] Awal, M.A., Rochan, M., Roy, C.K.: Model compression vs. adversarial robustness: An empirical study on language models for code. arXiv preprint arXiv:2508.03949 (2025)

[30] Awal, M.A., Rochan, M., Roy, C.K.: Model compression vs. adversarial robustness: An empirical study on language models for code. arXiv preprint arXiv:2508.03949 (2025)

[31] Mukhoti, J., Gal, Y., Torr, P.H., Dokania, P.K.: Fine-tuning can cripple your foundation model; preserving features may be the solution. arXiv preprint arXiv:2308.13320 (2023)

[32] Aghajanyan, A., Gupta, S., Zettlemoyer, L.: Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (volume 1: Long Papers), pp. 7319–7328 (2021)

[33] Lester, B., Al-Rfou, R., Constant, N.: The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691 (2021)

[34] Christophe, C., Kanithi, P.K., Munjal, P., Raha, T., Hayat, N., Rajan, R., Al-Mahrooqi, A., Gupta, A., Salman, M.U., Gosal, G., et al.: Med42–evaluating fine-tuning strategies for medical llms: full-parameter vs. parameter-efficient approaches. arXiv preprint arXiv:2404.14779 (2024)

[35] Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., Ma, X., Efrat, A., Yu, P., Yu, L., *et al.*: Lima: Less is more for alignment. Advances in Neural Information Processing Systems **36**, 55006–55021 (2023)

[36] Biderman, D., Portes, J., Ortiz, J.J.G., Paul, M., Greengard, P., Jennings, C., King, D., Havens, S., Chiley, V., Frankle, J., et al.: Lora learns less and forgets less. arXiv preprint arXiv:2405.09673 (2024)

[37] Rajbhandari, S., Rasley, J., Ruwase, O., 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 (2020). IEEE

[38] Kumar, A., Agarwal, C., Srinivas, S., Li, A.J., Feizi, S., Lakkaraju, H.: Certifying llm safety against adversarial prompting. arXiv preprint arXiv:2309.02705 (2023)

[39] Raina, V., Liusie, A., Gales, M.: Is llm-as-a-judge robust? investigating universal adversarial attacks on zero-shot llm assessment. arXiv preprint arXiv:2402.14016 (2024)

[40] Shayegani, E., Mamun, M.A.A., Fu, Y., Zaree, P., Dong, Y., Abu-Ghazaleh, N.: Survey of vulnerabilities in large language models revealed by adversarial attacks. arXiv preprint arXiv:2310.10844 (2023)

[41] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)

[42] Li, L., Wang, Y., Lin, K.-Y.: Preventive maintenance scheduling optimization based on opportunistic production-maintenance synchronization. Journal of Intelligent Manufacturing **32**(2), 545–558 (2021)

[43] Holcomb, S.D., Porter, W.K., Ault, S.V., Mao, G., Wang, J.: Overview on deepmind and its alphago zero ai. In: Proceedings of the 2018 International Conference on Big Data and Education, pp. 67–71 (2018)

[44] Hu, K., Gong, S., Zhang, Q., Seng, C., Xia, M., Jiang, S.: An overview of implementing security and privacy in federated learning. Artificial intelligence review **57**(8), 204 (2024)

[45] Qi, P., Chiaro, D., Guzzo, A., Ianni, M., Fortino, G., Piccialli, F.: Model aggregation techniques in federated learning: A comprehensive survey. Future Generation Computer Systems **150**, 272–293 (2024)

[46] Liu, B., Lv, N., Guo, Y., Li, Y.: Recent advances on federated learning: A systematic survey. Neurocomputing **597**, 128019 (2024)

[47] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012)

[48] Xu, J., Zhou, H., Gan, C., Zheng, Z., Li, L.: Vocabulary learning via optimal transport for neural machine translation. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 7361–7373 (2021)

[49] Bai, J., Chen, D., Qian, B., Yao, L., Li, Y.: Federated fine-tuning of large language models under heterogeneous tasks and client resources. Advances in Neural Information Processing Systems **37**, 14457–14483 (2024)

[50] Zhang, P., Zeng, G., Wang, T., Lu, W.: Tinyllama: An open-source small language model, 2024. URL https://arxiv. org/abs/2401.02385 (2024)

[51] Li, R., Allal, L.B., Zi, Y., Muennighoff, N., Kocetkov, D., Mou, C., Marone, M., Akiki, C., Li, J., Chim, J., et al.: Starcoder: may the source be with you! arXiv preprint arXiv:2305.06161 (2023)

[52] Chen, C., Sun, F., Zhang, M., Ding, B.: Recommendation unlearning. In: Proceedings of the ACM Web Conference 2022, pp. 2768–2777 (2022)

[53] Li, Y., Zhang, Z., Liu, B., Yang, Z., Liu, Y.: Modeldiff: Testing-based dnn similarity comparison for model reuse detection. In: Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 139–151 (2021)

[54] Albrecht, J.P.: How the gdpr will change the world. Eur. Data Prot. L. Rev. **2**, 287 (2016)

[55] McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282 (2017). PMLR

[56] Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(2), 1–19 (2019)

[57] Jiang, W., Luo, Y., Deng, G., Chen, S., Yang, X., Wu, S., Gao, X., Liu, L., Fu, S.: Federated large language models: Feasibility, robustness, security and future directions. arXiv preprint arXiv:2505.08830 (2025)

[58] Zhang, L., Li, Y.: Federated learning with layer skipping: Efficient training of large language models for healthcare nlp. arXiv preprint arXiv:2504.10536 (2025)

[59] Xie, Z., Brophy, J., Noack, A., You, W., Asthana, K., Perkins, C., Reis, S., Singh, S., Lowd, D.: Identifying adversarial attacks on text classifiers. arXiv preprint arXiv:2201.08555 (2022)

[60] Asthana, K., Xie, Z., You, W., Noack, A., Brophy, J., Singh, S., Lowd, D.: Tcab: a large-scale text classification attack benchmark. arXiv preprint arXiv:2210.12233 (2022)

[61] Dijkstra, E.W.: Solution of a problem in concurrent programming control. In: Pioneers and Their Contributions to Software Engineering: Sd&m Conference on Software Pioneers, Bonn, June 28/29, 2001, Original Historic Contributions, pp. 289–294 (2001). Springer

[62] Stallings, W.: Operating Systems: Internals and Design Principles. Prentice Hall Press, ??? (2011)

[63] Su, N., Hu, C., Li, B., Li, B.: Titanic: Towards production federated learning with large language models. In: IEEE INFOCOM 2024-IEEE Conference on Computer Communications, pp. 611–620 (2024). IEEE

[64] Zhang, W.E., Sheng, Q.Z., Alhazmi, A., Li, C.: Adversarial attacks on deep-learning models in natural language processing: A survey. ACM Transactions on Intelligent Systems and Technology (TIST) **11**(3), 1–41 (2020)

[65] Ricart, G., Agrawala, A.K.: An optimal algorithm for mutual exclusion in computer networks. Communications of the ACM **24**(1), 9–17 (1981)

[66] Lamport, L.: A new solution of dijkstra's concurrent programming problem. In: Concurrency: the Works of Leslie Lamport, pp. 171–178 (2019)

[67] Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2019).

https://openreview.net/forum?id=Bkg6RiCqY7

[68] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2015)

[69] Chauhan, T., Palivela, H.: Optimization and improvement of fake news detection using deep learning approaches for societal benefit. International Journal of Information Management Data Insights **1**(2), 100051 (2021)