FedSEA-LLaMA: A Secure, Efficient and Adaptive Federated Splitting Framework for Large Language Models

Zishuai Zhang[1,2], Hainan Zhang[1,2*], Weihua Li[2], Qinnan Zhang[1,2], Jin Dong[1,3*], Yongxin Tong[4*], Zhiming Zheng[1,2]

[1]Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing, Beihang University, China
[2]School of Artificial Intelligence, Beihang University, China
[3]Beijing Academy of Blockchain and Edge Computing, China
[4]School of Computer Science and Engineering, Beihang University, China
zhangzishuai@buaa.edu.cn, zhanghainan@buaa.edu.cn

Abstract

Private data holds promise for improving LLMs due to its high quality, but its scattered distribution across data silos and the high computational demands of LLMs limit their deployment in federated environments. To address this, the transformer-based federated split models are proposed, which offload most model parameters to the server (or distributed clients) while retaining only a small portion on the client to ensure data privacy. Despite this design, they still face three challenges: 1) Peer-to-peer key encryption struggles to secure transmitted vectors effectively; 2) The auto-regressive nature of LLMs means that federated split learning can only train and infer sequentially, causing high communication overhead; 3) Fixed partition points lack adaptability to downstream tasks. In this paper, we introduce FedSEA-LLaMA, a Secure, Efficient, and Adaptive Federated splitting framework based on LLaMA2. First, we inject Gaussian noise into forward-pass hidden states to enable secure end-to-end vector transmission. Second, we employ attention-mask compression and KV cache collaboration to reduce communication costs, accelerating training and inference. Third, we allow users to dynamically adjust the partition points for input/output blocks based on specific task requirements. Experiments on natural language understanding, summarization, and conversational QA tasks show that FedSEA-LLaMA maintains performance comparable to centralized LLaMA2 and achieves up to 8× speedups in training and inference. Further analysis of privacy attacks and different partition points also demonstrates the effectiveness of FedSEA-LLaMA in security and adaptability.

Code — https://github.com/TAP-LLM/SplitFedLLM/tree/main/FedSEA-LLaMA
Extended version — https://arxiv.org/abs/2505.15683

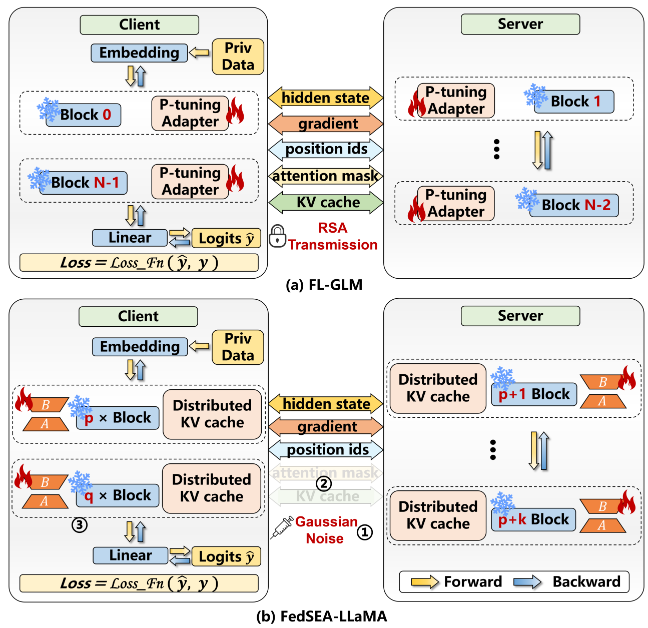


Figure 1: Model architecture of FL-GLM and FedSEA-LLaMA. ①FedSEA-LLaMA employs Gaussian noise addition to protect intermediate vectors. ②FedSEA-LLaMA minimizes the transmission of intermediate data and further accelerates long-context inference. ③FedSEA-LLaMA adaptively adjusts model partition points based on task requirements and system load.

Introduction

While the success of Large Language Models (LLMs) has largely stemmed from their ability to process vast amounts of public data (Kaplan et al. 2020), the true potential of these models lies in their application to high-quality private data. Private datasets often contain more granular and specific insights about individuals or organizations, typically collected through more controlled and precise methodologies (Yang et al. 2024). Additionally, private data tend to be more timely, targeted, and unique, thus providing models with semantic contexts and environmental settings that more closely mirror real-world scenarios (Ye et al. 2024). Therefore, effectively incorporating private data will further advance the performance of LLMs.

The distributed private data silos and the significant computational demands of LLMs pose substantial challenges for their deployment in federated environments. Private data is

*Corresponding author

often isolated in local silos, such as mobile devices (Ghimire and Rawat 2022), enterprise servers (Huang et al. 2024), or medical institutions (Yuan et al. 2024), as centralized storage and processing of such data may raise privacy concerns and regulatory risks (Chen et al. 2024). Federated Learning (FL) (Stremmel and Singh 2021) offers a promising solution by enabling local training on user devices and only sending model parameters for aggregation. They often leverage Adapters (Houlsby et al. 2019) or LoRA (Hu et al. 2022) as transmitted parameters, or employ collaborative training between large and small models (Hinton, Vinyals, and Dean 2015) to fine-tune LLMs. However, in extremely resource-constrained scenarios (e.g., less than 3.5GB of GPU memory), fine-tuning the LLMs remains infeasible. Therefore, how to effectively deploy LLMs in FL frameworks while ensuring privacy and scalability in resource-constrained scenarios remains an open research problem.

To address this, transformer-based federated splitting models (Zheng et al. 2024; Su et al. 2024) have been proposed to adapt LLM fine-tuning to resource-constrained scenarios. Titanic (Su et al. 2024) handles limited server resources by partitioning LLM across clients and enabling collaborative training. FL-GLM (Zheng et al. 2024) offloads most model parameters to a resource-ample server while retaining the input and output layer on the client. As shown in Figure 1a, FL-GLM divides Transformers into three parts, i.e., the first input block and the last output block on the client, while the middle blocks on the server. The forward propagation (FP) path is client-to-server-to-client with peer-to-peer key encryption. After the client calculates loss, gradients backpropagate (BP) from client-to-server-to-client with the same encryption. Although FL-GLM provides a potentially viable solution for clients in resource-constrained scenarios, it still faces several limitations: **1) Privacy Leakage:** during forward and backward propagation, activation values and gradients from the input blocks are transmitted, which can be exploited to reconstruct original input data through reverse engineering (Asnani et al. 2023). **2) Communication Cost:** LLMs are auto-regressive, requiring sequential token generation based on prior tokens. In federated split learning, this necessitates a full forward and backward pass with client-server communication for each token, greatly increasing communication rounds and bandwidth usage. **3) Lack of Adaptability:** Fixed partition points of FL-GLM lack flexibility to downstream tasks and hardware constraints, as different devices vary in their capacity to handle workloads and parameter sizes.

In this paper, we introduce FedSEA-LLaMA, a Secure, Efficient, and Adaptive Federated split framework based on LLaMA2, as shown in Figure 1b. To **Enable Secure Propagation**, we add Gaussian noise for the hidden states of input blocks and send these noised states to the server for further processing. Through both gradient analysis and experimental verification, this noise ensures privacy and security during subsequent training and inference stages. To **Reduce Communication Cost**, we introduce the attention-mask compression and collaborative KV cache mechanisms, minimizing overhead while preserving the integrity of auto-regressive inference. By synchronously maintaining cache states on both client and server, the originally cross-device transmission is transformed into dynamic attention-mask generation, reducing communication from megabytes (MB) to bytes (B). To **Achieve Adaptive Partition**, we design a dynamic partition strategy allowing users to flexibly adjust the number of input and output blocks according to specific downstream tasks and available computational resources. This adaptability enables an optimal balance between performance and efficiency.

Experiments on natural language understanding (NLU), summarization, and conversational QA show that FedSEA-LLaMA maintains performance comparable to centralized LLaMA2 and achieves up to 8× inference speedups. Through inference acceleration, the average token generation time of FedSEA-LLaMA has been reduced by 87.6%, and the speed gain will further increase with the growing length of context. Additional analysis of privacy attacks and various partition points further highlights FedSEA-LLaMA's effectiveness in terms of security and adaptability.

The innovations in this paper are as follows:

- We design a federated split framework tailored for resource-limited scenarios, featuring adaptive partitioning and client-side Gaussian noise injection to enable privacy-preserving training and inference of LLMs.
- We leverage collaborative KV caching and dynamic attention-mask generation to drastically reduce communication overhead from megabytes to bytes during auto-regressive inference, achieving up to 8x speedups.
- Experimental results on three different tasks show that our FedSEA-LLaMA achieves performance comparable to centralized LLaMA2, and validate the security, efficiency, and adaptability of our framework.

Related Works

Federated Learning for LLMs

Traditional Federated Learning (FL) frameworks face scalability challenges with large language models (LLMs) due to high communication and computational costs (McMahan et al. 2017; Stremmel and Singh 2021; Ji et al. 2019). Recent federated LLM frameworks (Fan et al. 2023; Kuang et al. 2024; Ye et al. 2024) address this by using instruction tuning, Adapters, and LoRA to reduce parameter updates, lowering communication overhead. However, transmission demands often still exceed system limits. To mitigate this, gradient compression (Wu et al. 2024b; Shu et al. 2024) and quantization (JianHao et al. 2024) techniques have been applied. Structural pruning based on Fisher Information (Liu et al. 2024) further narrows updates to key layers, reducing model size without performance loss. Knowledge distillation (Wu et al. 2024a; Fan et al. 2024, 2025) offers another solution, using smaller proxy models to transfer knowledge with minimal communication. Client heterogeneity remains a major challenge (Zhang et al. 2024), prompting methods like LoRA-based stacking (Bai et al. 2024; Wang et al. 2024) to align models from diverse clients and improve fairness.

Despite advances in efficient fine-tuning and parameter reduction for federated LLMs, resource-constrained clients

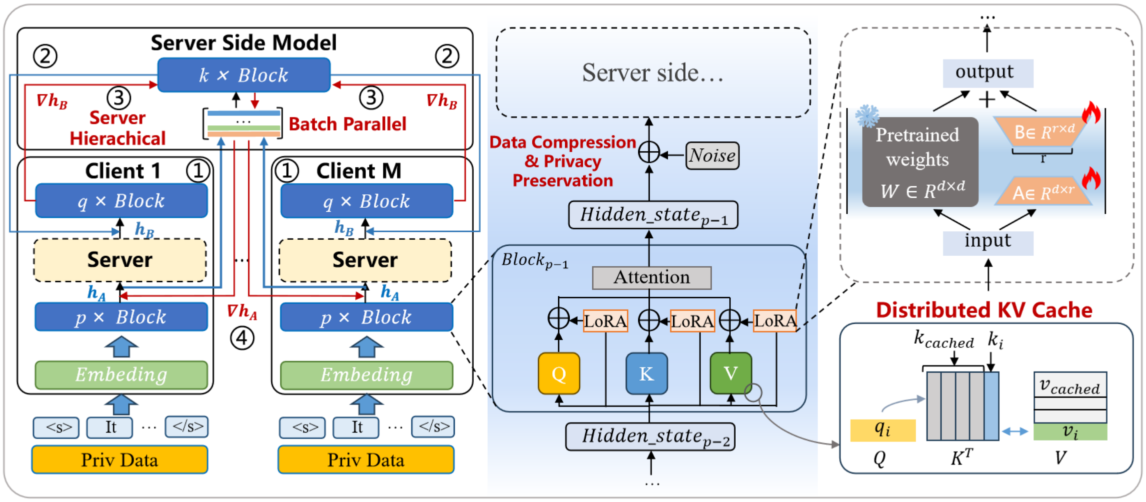


Figure 2: Overall Framework of FedSEA-LLaMA. Model Split with adaptive partition points and Gaussian noise on forward-passing vectors to preserve privacy for LoRA fine-tuning strategy, and achieving attention-mask compression and KV Cache collaborations to accelerate inference process. The local client stores the first p blocks ($Blocks_A$) and the last q blocks ($Blocks_C$), while the server stores the remaining k blocks ($Blocks_B$). During training, ① h_A is transmitted to the server-side model $Blocks_B$; ② h_B is transmitted to the client-side model $Blocks_C$; ③ the gradient with respect to h_B is backpropagated to the server; ④ the gradient with respect to h_A is backpropagated to the client-side model.

still face challenges with independent fine-tuning, particularly under compression overhead. Split learning (Thapa et al. 2022) offers a promising alternative in such settings.

Split Learning for LLMs

Split learning is an emerging decentralized training approach designed for resource-constrained settings (Abedi and Khan 2024; Rahman et al. 2020; Matsubara and Levorato 2021), especially mobile or GPU-limited clients. It works by dividing a neural network into sub-networks, each processed on a different device for collaborative learning. To address the training of sequential data in LLMs, FedBERT (Tian et al. 2022) introduces an innovative federated learning framework designed to pre-train language models in a split architecture, addressing the challenge of limited computational capacity on client devices. Although this design enables more efficient distribution of training workloads, it comes with drawbacks, such as increased communication overhead, fixed partition points, and susceptibility to privacy threats like embedding gradient attacks.

Recently, Su et al. (2024) extends the scope of split learning by exploring scenarios with heterogeneous resources, particularly considering a case where the server lacks sufficient computational capacity. In their approach, the LLM is automatically divided into k partitions, each assigned to one of k clients, while the server merely performs aggregation. Although this strategy alleviates server-side computation, it considerably increases the overall training and inference time. Furthermore, repeated transmission of hidden states increases the risk of single points of failure and introduces potential security vulnerabilities. Zheng et al. (2024) propose FL-GLM to place the input and output blocks locally on client devices, while the remaining primary model parameters are centralized on a server with ample computational resources. And then, they employ key encryption during client-server communication to prevent reverse engineering attacks from peer clients. But the peer-to-peer key encryption is not enough because the peer clients and servers can collaborate to infer the clients' private data.

Our work diverges from these studies by enhancing both training/inference efficiency, adaptive partition points, and data privacy for distributed LLMs through optimized algorithmic designs and communication strategies.

Model

In this section, we present a detailed description of the FedSEA-LLaMA framework, as shown in Figure 2, which comprises three components: privacy preservation, inference acceleration, and adaptive partitioning. Following FL-GLM, we adopt split learning to divide the LLaMA into three parts: the local client stores the first p blocks($Blocks_A$) and the last q blocks($Blocks_C$), while the remaining k blocks($Blocks_B$), comprising the majority of the model parameters, are hosted on the server.

Privacy Preservation

In distributed training of LLMs, the split learning paradigm partitions the entire model into multiple sub-models deployed across different physical nodes. As a result, the hidden states and gradients must be transferred across nodes via the network. Unlike traditional federated learning, which only shares gradients, this approach exposes mid-layer activations and their gradients, greatly increasing the risk of adversaries reverse-engineering user data and challenging the data protection framework of federated split LLMs.

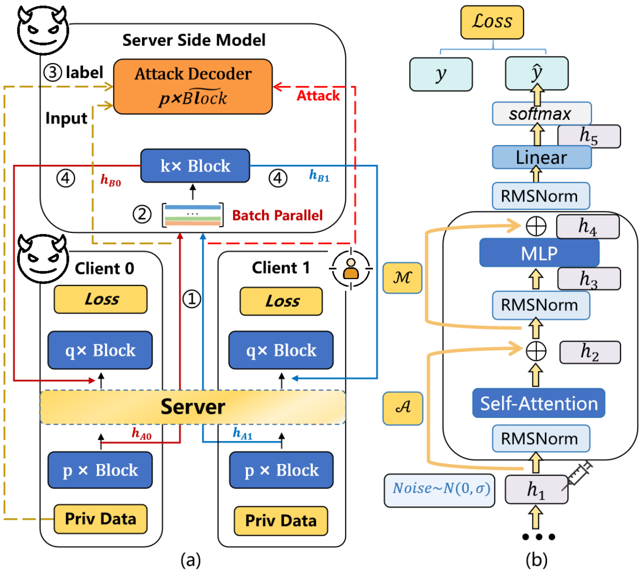


Figure 3: (a)Threat model attack in Multi-client Training. ①Clients upload hidden states; ②Server processes via $Blocks_B$; ③Attack decoder learns from malicious client's data; ④Normal outputs are sent back to all clients. (b) Gaussion noise injection. Adding noise to h_1 provides protection for both the forward and backward passes.

Threat Model Definition We design an extreme attack scenario to evaluate the robustness of FedSEA-LLaMA. Inspired by Pasquini, Ateniese, and Bernaschi (2021), we construct a model inversion attack in a multi-client federated setting, as illustrated in Figure 3a, where a malicious client colludes with an honest-but-curious server. The malicious client shares its private data in plaintext with the server, allowing the server to train an attack model locally. The goal of the attack model is to reconstruct the original input text from the hidden states transmitted by the client. During this process, the server does not tamper with the hidden states sent to other honest clients, thereby performing a stealthy inversion attack against their private data without awareness.

Concretely, we consider an adversary that has full knowledge of the client-side model architecture and the number of layers deployed on the client, but does not have access to the actual model parameters. Thus, the honest-but-curious server locally initializes a decoder with the same architecture and depth as the client's model and uses it as the attack model. Without loss of generality, we consider a two-client FedSEA-LLaMA setting, where the server receives hidden states from both $client_0$ and $client_1$. During each training step, the server first performs a normal forward pass through $Blocks_B$ to obtain the intermediate hidden states h_B. It then feeds the hidden states from the malicious $client_0$ into the attack model and uses the corresponding private data D_{priv} as supervision. Afterward, h_B is split and sent back to the respective clients, allowing training to proceed as usual without disrupting the federated process.

Defense with Gaussian Noise Injection We explore two defense strategies to protect the transmission in FedSEA-LLaMA: adding noise to gradients and adding noise to forward-pass hidden states. The first strategy introduces random Gaussian noise into the initial gradient during backpropagation (i.e., the gradient ∇h_B), while the second applies Gaussian noise to the client's first forward-pass output (i.e., the hidden state h_A). Empirical results reveal that federated split LLMs are highly sensitive to gradient perturbations, so injecting noise into gradients often leads to instability in federated training, with the loss frequently diverging to NaN after several training steps. In contrast, perturbing the hidden state h_A during the forward pass yields more stable training. Therefore, FedSEA-LLaMA adopts the latter to enhance robustness while preserving privacy.

Next, we derive how the noise introduced in the forward pass can also propagate to the gradients during backpropagation. Without loss of generality, we apply noise perturbation to h_1 in the theoretical proof, although in practice we apply noise perturbation to h_A. The proof will be similar in both cases. Figure 3b illustrates the schematic of noise injection during the forward propagation of LLMs. Let W_n denote the weight matrix of the down-projection fully-connected layer within the final MLP layer, and W_{n+1} denote the weight of the output linear layer. In the noise-free scenario, the hidden states h_1 undergo forward propagation:

$$
\begin{aligned}
h_2 &= \mathcal{A}(h_1) + h_1, && (1)\\
h_3 &= \text{RMSNorm}(h_2), && (2)\\
h_4 &= \mathcal{M}(h_3) + h_2, && (3)\\
&= W_n[Silu(gate_{proj}(h_3)) \cdot up_{proj}(h_3)] + h_2, && (4)\\
h_5 &= W_{n+1} \cdot \text{RMSNorm}(h_4), && (5)\\
\hat{y} &= \text{softmax}(h_5), && (6)\\
\mathcal{L} &= CrossEntropy(y, \hat{y}), && (7)
\end{aligned}
$$

where $\text{RMSNorm}(*)$ is the layernorm function, $\mathcal{A}$ is the attention layer, $\mathcal{M}$ is the MLP layer.

During backpropagation, according to (Clark 2017), the gradient of W_n is:

$$
\begin{aligned}
\frac{\partial L}{\partial W_n} &= \frac{\partial L}{\partial h_5} \cdot \frac{\partial h_5}{\partial h_4} \cdot \frac{\partial h_4}{\partial W_n}, && (8)\\
\frac{\partial L}{\partial h_5} &= (\hat{y} - y), && (9)\\
\frac{\partial h_5}{\partial h_4} &= W_{n+1} \cdot J_{\text{RMSNorm}}(h_4), && (10)\\
\frac{\partial h_4}{\partial W_n} &= [Silu(gate_{proj}(h_3)) \cdot up_{proj}(h_3)]^T. && (11)
\end{aligned}
$$

According to Eq. 1-11, the gradient of W_n is closely related to the value of h_1. Thus, when Gaussian noise $\mathcal{N}(0, \delta)$ is injected into the hidden states h_1, perturbations are consequently introduced into the gradients of parameters, thereby influencing subsequent gradient computations. In other words, injecting noise into the forward propagation process can protect the hidden states and their gradients transmitted during the training of the splitting LLMs.

Inference Acceleration

The inference time of LLMs is crucial for the efficient and timely execution of complex tasks. To enhance the efficiency and usability of distributed LLMs, our proposed FedSEA-LLaMA framework incorporates attention-mask compression and collaborative KV cache mechanisms to minimize redundant data transmission.

Attention-Mask Compression FedSEA-LLaMA transmits only the minimal information necessary for effective model training and inference. In particular, it reduces redundancy in attention-mask transmissions compared to FL-GLM. Since the generation of attention-mask remains consistent across transformer blocks within each training iteration and depends solely on dataset padding, transmitting full attention-mask tensors $A_{mask} \in [0, -\infty]^{batch\_size \times seq\_len \times seq\_len}$ results in unnecessary communication overhead during training and inference. Instead, FedSEA-LLaMA transmits only essential metadata (seq_len, pad_len), indicating padding locations, thereby significantly decreasing bandwidth usage.

Collaborative KV Cache Mechanism In traditional distributed inference of LLMs, each inference step involves transmitting all previously generated tokens from the client to the server. This design leads to two major inefficiencies: (1) the computational overhead increases linearly with the growing input length, since the attention mechanism must recompute key and value (KV) representations for all past tokens; and (2) the repeated transmission of the entire context between client and server introduces substantial communication latency, especially for long sequences.

To address these issues, we propose a collaborative KV cache mechanism tailored for distributed inference. Instead of recomputing and retransmitting all past tokens at each step, our approach caches the attention KV states locally on both the client and server sides. Since these KV states remain static across the autoregressive generation process, subsequent tokens can reuse the cached representations. Built on this mechanism, FedSEA-LLaMA enables highly efficient distributed inference by significantly reducing both computational and communication overhead. Experimental results show that this strategy leads to substantial improvements in inference speed in scenarios with long user queries.

Adaptive Partition

To flexibly accommodate specific task requirements and hardware constraints, we propose an adaptive splitting strategy that enables users to adjust the number of model input and output blocks, thereby achieving an optimal balance between performance and efficiency. Based on this strategy, we explore the performance of downstream tasks when the client holds the first blocks (with $Blocks_A$=1,2,3) and the last blocks (with $Blocks_C$=1,2,3) of the model. The results indicate that the depth of the model split does not significantly impact performance on downstream tasks, while affording a more flexible allocation of computational loads between the server and the clients. Furthermore, it corroborates the efficiency of our adaptive splitting strategy and delineates a refined configuration paradigm for deploying LLMs in federated learning environments.

Experiments

We evaluate FedSEA-LLaMA on NLU, summarization, and conversational QA to assess its performance against centralized training while tackling latency and privacy via decentralized learning. More experiments, including parallel training and training prompts, can be found in the Appendix.

Experimental Settings

In this section, we introduce some empirical settings, including datasets, evaluation metrics, baselines, and parameter settings for FedSEA-LLaMA.

Dataset To evaluate model generalization across diverse NLP tasks, our experiments incorporate three distinct tasks:

- The SuperGLUE benchmark (Wang et al. 2020) evaluates eight distinct NLU tasks, from common sense reasoning to fine-grained semantic interpretation.
- The CoQA dataset (Reddy, Chen, and Manning 2019) is designed for the task of conversational question answering(QA), featuring 8k+ dialogues across seven domains. Answers are free-form text, with nearly half of the questions requiring coreference resolution and pragmatic reasoning, simulating real-world dialogue coherence.
- The XSum dataset (Narayan, Cohen, and Lapata 2018) evaluates abstractive summarization via extreme compression of BBC news articles into single-sentence summaries. It tests the model's ability to abstract and condense salient information across diverse topics.

Metrics We evaluate model performance using standard task-specific metrics. For NLU tasks such as COPA, WSC, etc, we report Accuracy, which measures the proportion of instances where the model's predicted label exactly matches the ground-truth label.

For conversational QA tasks, we use Exact Match (EM) and F1 score to evaluate model performance (Yatskar 2019; You et al. 2022). EM measures the percentage of predictions that exactly match the ground truth, accounting for punctuation and case, while F1 reflects the harmonic mean of precision and recall, rewarding partial overlaps.

For summarization tasks, we use standard ROUGE metrics (Lin 2004; Liu et al. 2021; Fang et al. 2022), specifically ROUGE-1 and ROUGE-2, which assess unigram and bigram overlap between generated and reference summaries, capturing content coverage and fluency.

Baselines We evaluate FedSEA-LLaMA based on the LLaMA2-7B model (Touvron et al. 2023), an open-source pre-trained model developed by Meta and FL-GLM. We additionally include several strong baselines: T5_{large} (Raffel et al. 2020), $\text{BART}_{\text{Large}}$ (Lewis et al. 2020), $\text{GLM}_{\text{RoBERTa}}$ (Du et al. 2022) and ChatGLM-6B (GLM et al. 2024).

Parameter Settings The server is equipped with a Gigabit Ethernet card and utilizes multiple GPUs, including

Model	ReCoRD F1/Acc.	COPA Acc.	WSC Acc.	RTE Acc.	BoolQ Acc.	WiC Acc.	CB F1	MultiRC F1a/EM	CoQA F1/EM	Xsum rouge-1/rouge-2
$T5_{large}$ (Raffel et al. 2020)	85.7/85.0	78.0	84.6	84.8	84.3	71.6	96.4	80.9/46.6	-	40.9/17.3
$BART_{Large}$ (Lewis et al. 2020)	88.3/87.8	60.0	65.4	84.5	84.3	69.0	90.5	81.8/48.0	-	45.1/22.3
$GLM_{RoBERTa}$ (Du et al. 2022)	89.6/89.0	82.0	83.7	87.7	84.7	71.2	98.7	82.4/50.1	-	45.5/23.5
ChatGLM-6B (Zeng et al. 2022)	80.2/78.7	85.0	71.2	81.6	83.4	71.0	85.7	78.2/45.6	-	37.6/12.5
FL-GLM (Zheng et al. 2024)	79.8/78.4	85.0	71.2	80.1	81.9	69.6	85.7	79.3/46.1	62.7/49.0	37.0/11.9
LLaMA2-7B (Touvron et al. 2023)	81.3/79.5	75.0	68.3	73.3	83.8	70.9	85.7	82.3/50.1	88.6/80.9	45.9/26.1
FedSEA-LLaMA	81.3/79.4	75.0	68.4	73.3	82.4	70.8	85.7	82.7/48.0	88.6/80.7	47.6/25.0

Table 1: Performance Comparison on SupleGLUE, CoQA and Xsum datasets between FedSEA-LLaMA and baselines.

two NVIDIA A6000 and two NVIDIA L40. The implementation is based on the Flower (Beutel et al. 2020) framework, which orchestrates the transfer of intermediate data between clients and the server. Due to limited computational resources, FP32 precision and a batch size of 1 are used only for the CoQA dataset. All other experiments are conducted with FP16 precision and a batch size of 2. The random seed is consistently set to 42. Fine-tuning is performed using distributed LoRA training, where $Blocks_A$, $Blocks_B$, and $Blocks_C$ each maintain their own LoRA adapters. For all LoRA fine-tuning experiments in this paper, the target modules are set to q_proj, v_proj, k_proj, and o_proj.

Experimental Results

FedSEA-LLaMA Performance We compared the performance of FedSEA-LLaMA with centralized LLaMA2-7B on various tasks, including NLU, summarization generation, and conversational QA, as shown in Table 1. Experimental results show that FedSEA-LLaMA achieves comparable performance to LLaMA2-7B, with metrics approaching or matching across multiple tasks, demonstrating its effectiveness. Meanwhile, compared to other baselines, FedSEA-LLaMA achieves better performance on datasets such as Record, MultiRC, and Xsum, but performs poorly on the RTE dataset. This is speculated to be related to the distribution difference between RTE and the data used for LLaMA2 pre-training, as well as suboptimal hyperparameters. The comparison of their GPU memory usage is shown in Figure 4. The result shows that FedSEA-LLaMA significantly reduces GPU memory consumption on the client side. Both models were fine-tuned on the Record dataset with a batch size of 2, using float16 precision and a LoRA rank of 8. While centralized LLaMA2-7B occupies 28.2G of GPU memory, FedSEA-LLaMA client requires only 3.4G, representing a reduction of up to 87.9% in memory requirements.

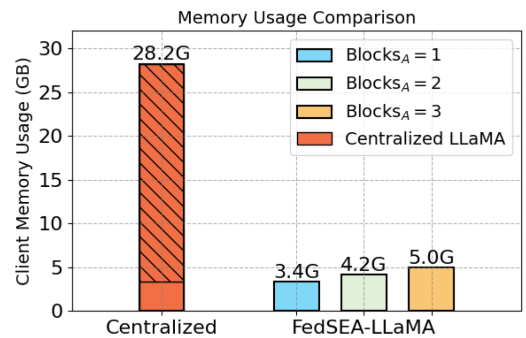


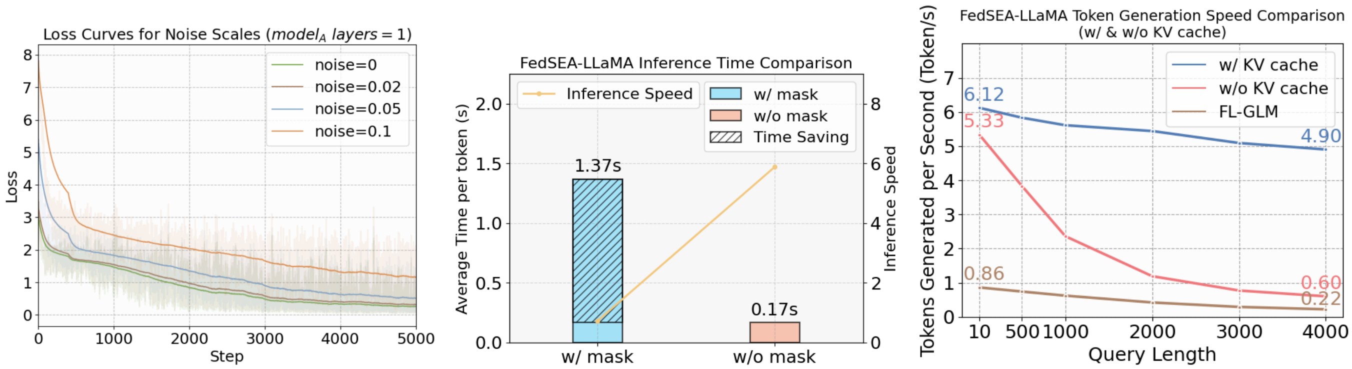
Figure 4: Computation load on client side of centralized LLaMA2-7B and FedSEA-LLaMA on ReCoRD dataset.

Noise Scale	$Blocks_A$ = 1	$Blocks_A$ = 2	$Blocks_A$ = 3
0.0	77.48/40.82	77.82/39.24	77.89/41.34
0.02	77.89/38.09	77.87/39.66	78.61/42.39
0.05	74.57/29.17	77.07/38.41	78.95/41.03
0.1	59.11/8.18	72.45/23.71	75.68/34.10

Table 2: Different noise scales and blocks number of $Blocks_A$ analysis(F1a / EM) on MultiRC dataset.

Effects of Noise Scales Since noise injection into the hidden states during forward propagation inevitably impacts the performance of distributed LLMs, we conducted extensive experiments to investigate how noise levels affect model performance. The corresponding experimental results are presented in Figure 5a and Table 2. Figure 5a demonstrates that when the numbers of blocks in $Blocks_A$ and $Blocks_C$ are consistent, the convergence speed of FedSEA-LLaMA decreases as the magnitude of injected noise increases. When the average absolute value of the noise is small (e.g., below 0.02), the loss curve remains close to that of the noise-free baseline. Table 2 indicates that under fixed noise levels, increasing the number of blocks in $Blocks_A$ consistently improves downstream performance in terms of both F1 and EM scores. Notably, when the noise amplitude reaches 0.1, only the single-block configuration of $Blocks_A$ shows a significant drop in downstream performance, whereas the three-block version of $Blocks_A$ demonstrates greater robustness.

Model Reverse Attack To assess how split points and noise scale affect the security of FedSEA-LLaMA, we evaluate the attack performance of the adversarial model under varying configurations of $Blocks_A$ depth ($Blocks_A$ = 1, 2, 3) and noise (scale = 0, 0.02, 0.5). Attacks use full-parameter fine-tuning (lr=2e-5), with reconstruction quality measured by BLEU-4 and ROUGE-2. FedBert reports a Rouge-2 of 7.053 and a Bleu-4 of 28.57. Experiment results of FedSEA-LLaMA are shown in Table 3. From the results, we can see that when the client holds only the embedding layer (similar to FedBERT), the attack model achieves relatively high BLEU and ROUGE scores. However, under the FedSEA-LLaMA framework, where the client also retains a portion of the LLM blocks in addition to the embedding layer, all at-



(a) Loss Curves for Noise Scales ($Blocks_A$=1, $Blocks_C$=1)

(b) Inference Speed Comparison (w/ vs. w/o Attention-Mask).

(c) Inference Speed Comparison (w/ vs. w/o KV Cache).

Figure 5: (a)Different noise injection on MultiRC dataset. (b)Ablation study on attention-mask. (c)Ablation study on KV cache.

Noise scale	$Blocks_A$**=1**	$Blocks_A$**=2**	$Blocks_A$**=3**
0.0	0.051/0.003	0.056/0.003	0.069/0.016
0.02	0.050/0.002	0.052/0.003	0.084/0.044
0.05	0.050/0.003	0.051/0.002	0.067/0.004

Table 3: Security analysis (Rouge-2 F1 / BlEU-4) of FedSEA-LLaMA on Record dataset.

$Blocks_A$	$Blocks_C$ **= 1**	$Blocks_C$ **= 2**	$Blocks_C$ **= 3**
1	80.34/44.60	80.63/46.59	80.90/47.11
2	80.41/45.86	80.62/45.96	79.94/45.02
3	81.17/47.22	80.74/46.59	80.20/45.33

Table 4: Different model partition analysis(F1a/EM) on MultiRC dataset. No Noise is added to the hidden states.

tack metrics approach zero. These findings demonstrate that FedSEA-LLaMA effectively mitigates model inversion attacks, thereby preserving the privacy of client-side data.

Efficiency Optimization Figure 5b shows a comparison of the time consumption between transferring the attention-mask and transferring only the actual input length during distributed training. The bar chart represents the average time spent per token in FedSEA-LLaMA's forward propagation, while the line chart represents the number of tokens generated per second(context length=500). We can see that transferring the compressed information of the attention-mask plays a significant role in reducing the latency of distributed LLMs, with an average time reduction of 87.6%.

As shown in Figure 5c, FedSEA-LLaMA demonstrates superior performance compared to FL-GLM in distributed inference, achieving over 7× speedup. Without the KV cache, the inference speed decreases significantly with longer queries, due to the increasing number of K and V vectors. In contrast, with the KV cache technique, the client only needs to send the intermediate vector of the current single token to the server, while reusing previously generated K and V vectors. This effectively avoids redundant computations. When the query length reaches 4000, the inference speed of FedSEA-LLaMA with KV cache is 8.2 times that without it. As query length increases further, this advantage continues to expand, greatly alleviating the computational burden and operational latency of distributed LLMs.

Adaptive Splitting The choice of model partitioning has a significant impact on the computational burden at the client side. To evaluate whether FedSEA-LLaMA maintains robust performance under different partitioning strategies, we investigate its downstream task performance when clients hold different numbers of blocks in $Blocks_A$ and $Blocks_C$ ($Blocks_A$ = 1, 2, 3 and $Blocks_C$ = 1, 2, 3). This setting reflects realistic constraints where clients possess limited computational resources. Table 4 shows F1a/EM scores across different partitioning schemes. Performance variations across partitions are minimal, indicating that FedSEA-LLaMA is robust to different split configurations. This flexibility allows resource-constrained clients to adjust block allocation based on their downstream task requirements and resource availability, without significant accuracy loss.

Conclusion

In this work, we propose FedSEA-LLaMA, a novel federated split learning framework tailored for LLMs. By securely offloading the majority of model computation to a central server while maintaining privacy-sensitive components on local clients, FedSEA-LLaMA addresses the core limitations of traditional federated split learning approaches. Through secure end-to-end propagation with noise injection, inference acceleration, and adaptive partitioning tailored to downstream tasks and hardware, FedSEA-LLaMA demonstrates significant advances in security, efficiency, and adaptability. This federated split LLMs framework not only keeps users' private data confined to their local devices but also significantly shortens training and inference time, making it better suited for handling the scale and complexity of LLMs. In future work, we aim to support more base models and deploy the FedSEA-LLaMA framework in scenarios where data privacy is a critical concern.

Acknowledgements

This work was funded by the National Natural Science Foundation of China (NSFC) under Grants No. 62406013,No.62388101, the Beijing Advanced Innovation Center Funds for Future Blockchain and Privacy Computing(GJJ-24-034), and the Fundamental Research Funds for the Central Universities.

References

Abedi, A.; and Khan, S. S. 2024. FedSL: Federated split learning on distributed sequential data in recurrent neural networks. *Multimedia Tools and Applications*, 83(10): 28891–28911.

Asnani, V.; Yin, X.; Hassner, T.; and Liu, X. 2023. Reverse engineering of generative models: Inferring model hyperparameters from generated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Bai, J.; Chen, D.; Qian, B.; Yao, L.; and Li, Y. 2024. Federated fine-tuning of large language models under heterogeneous language tasks and client resources. *arXiv e-prints*, arXiv–2402.

Beutel, D. J.; Topal, T.; Mathur, A.; Qiu, X.; Fernandez-Marques, J.; Gao, Y.; Sani, L.; Kwing, H. L.; Parcollet, T.; Gusmão, P. P. d.; and Lane, N. D. 2020. Flower: A Friendly Federated Learning Research Framework. *arXiv preprint arXiv:2007.14390*.

Chen, J.; Yan, H.; Liu, Z.; Zhang, M.; Xiong, H.; and Yu, S. 2024. When federated learning meets privacy-preserving computation. *ACM Computing Surveys*, 56(12): 1–36.

Clark, K. 2017. Computing neural network gradients. *unpublished Journal*.

Du, Z.; Qian, Y.; Liu, X.; Ding, M.; Qiu, J.; Yang, Z.; and Tang, J. 2022. GLM: General Language Model Pretraining with Autoregressive Blank Infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 320–335.

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

Fan, T.; Kang, Y.; Ma, G.; Fan, L.; Chen, K.; and Yang, Q. 2024. FedCoLLM: A Parameter-Efficient Federated Cotuning Framework for Large and Small Language Models. *arXiv preprint arXiv:2411.11707*.

Fan, T.; Ma, G.; Kang, Y.; Gu, H.; Song, Y.; Fan, L.; Chen, K.; and Yang, Q. 2025. FedMKT: Federated Mutual Knowledge Transfer for Large and Small Language Models. In *Proceedings of the 31st International Conference on Computational Linguistics*, 243–255.

Fang, Y.; Zhang, H.; Chen, H.; Ding, Z.; Long, B.; Lan, Y.; and Zhou, Y. 2022. From spoken dialogue to formal summary: An utterance rewriting for dialogue summarization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 3859–3869.

Ghimire, B.; and Rawat, D. B. 2022. Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things. *IEEE Internet of Things Journal*, 9(11): 8229–8249.

GLM, T.; Zeng, A.; Xu, B.; Wang, B.; Zhang, C.; Yin, D.; Zhang, D.; Rojas, D.; Feng, G.; Zhao, H.; et al. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. In *International conference on machine learning*, 2790–2799. PMLR.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2): 3.

Huang, W.; Ye, M.; Shi, Z.; Wan, G.; Li, H.; Du, B.; and Yang, Q. 2024. Federated learning for generalization, robustness, fairness: A survey and benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Ji, S.; Pan, S.; Long, G.; Li, X.; Jiang, J.; and Huang, Z. 2019. Learning private neural language modeling with attentive aggregation. In *2019 International joint conference on neural networks (IJCNN)*, 1–8. IEEE.

JianHao, Z.; Lv, C.; Wang, X.; Wu, M.; Liu, W.; Li, T.; Ling, Z.; Zhang, C.; Zheng, X.; and Huang, X.-J. 2024. Promoting Data and Model Privacy in Federated Learning through Quantized LoRA. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 10501–10512.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Kuang, W.; Qian, B.; Li, Z.; Chen, D.; Gao, D.; Pan, X.; Xie, Y.; Li, Y.; Ding, B.; and Zhou, J. 2024. 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*, 5260–5271.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, 74–81.

Liu, J.; Ren, J.; Jin, R.; Zhang, Z.; Zhou, Y.; Valduriez, P.; and Dou, D. 2024. Fisher Information-based Efficient Curriculum Federated Learning with Large Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 10497–10523.

Liu, J.; Zou, Y.; Zhang, H.; Chen, H.; Ding, Z.; Yuan, C.; and Wang, X. 2021. Topic-Aware Contrastive Learning for Abstractive Dialogue Summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 1229–1243.

Matsubara, Y.; and Levorato, M. 2021. Neural compression and filtering for edge-assisted real-time object detection in challenged networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, 2272–2279. IEEE.

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

Narayan, S.; Cohen, S. B.; and Lapata, M. 2018. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 1797–1807.

Pasquini, D.; Ateniese, G.; and Bernaschi, M. 2021. Unleashing the tiger: Inference attacks on split learning. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2113–2129.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.

Rahman, S. A.; Tout, H.; Talhi, C.; and Mourad, A. 2020. Internet of things intrusion detection: Centralized, on-device, or federated learning? *IEEE Network*, 34(6): 310–317.

Reddy, S.; Chen, D.; and Manning, C. D. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7: 249–266.

Shu, Y.; Hu, W.; Ng, S.-K.; Low, B. K. H.; and Yu, F. R. 2024. Ferret: Federated Full-Parameter Tuning at Scale for Large Language Models. In *International Workshop on Federated Foundation Models in Conjunction with NeurIPS 2024*.

Stremmel, J.; and Singh, A. 2021. Pretraining federated text models for next word prediction. In *Advances in Information and Communication: Proceedings of the 2021 Future of Information and Communication Conference (FICC), Volume 2*, 477–488. Springer.

Su, N.; Hu, C.; Li, B.; and Li, B. 2024. TITANIC: Towards production federated learning with large language models. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, 611–620. IEEE.

Thapa, C.; Arachchige, P. C. M.; Camtepe, S.; and Sun, L. 2022. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 8485–8493.

Tian, Y.; Wan, Y.; Lyu, L.; Yao, D.; Jin, H.; and Sun, L. 2022. FedBERT: When federated learning meets pre-training. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4): 1–26.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Wang, A.; Pruksachatkun, Y.; Nangia, N.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2020. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. arXiv:1905.00537.

Wang, Z.; Shen, Z.; He, Y.; Sun, G.; Wang, H.; Lyu, L.; and Li, A. 2024. FLoRA: Federated Fine-Tuning Large Language Models with Heterogeneous Low-Rank Adaptations. *CoRR*.

Wu, F.; Li, Z.; Li, Y.; Ding, B.; and Gao, J. 2024a. Fedbiot: Llm local fine-tuning in federated learning without full model. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3345–3355.

Wu, H.; Li, X.; Zhang, D.; Xu, X.; Wu, J.; Zhao, P.; and Liu, Z. 2024b. CG-FedLLM: How to Compress Gradients in Federated Fune-tuning for Large Language Models. *CoRR*.

Yang, X.; Yu, H.; Gao, X.; Wang, H.; Zhang, J.; and Li, T. 2024. Federated continual learning via knowledge fusion: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 36(8): 3832–3850.

Yatskar, M. 2019. A Qualitative Comparison of CoQA, SQuAD 2.0 and QuAC. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2318–2323.

Ye, R.; Wang, W.; Chai, J.; Li, D.; Li, Z.; Xu, Y.; Du, Y.; Wang, Y.; and Chen, S. 2024. Openfedllm: Training large language models on decentralized private data via federated learning. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, 6137–6147.

You, C.; Chen, N.; Liu, F.; Ge, S.; Wu, X.; and Zou, Y. 2022. End-to-end Spoken Conversational Question Answering: Task, Dataset and Model. In *Findings of the Association for Computational Linguistics: NAACL 2022*, 1219–1232.

Yuan, L.; Wang, Z.; Sun, L.; Yu, P. S.; and Brinton, C. G. 2024. Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*.

Zeng, A.; Liu, X.; Du, Z.; Wang, Z.; Lai, H.; Ding, M.; Yang, Z.; Xu, Y.; Zheng, W.; Xia, X.; et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Zhang, J.; Vahidian, S.; Kuo, M.; Li, C.; Zhang, R.; Yu, T.; Wang, G.; and Chen, Y. 2024. Towards building the federatedgpt: Federated instruction tuning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6915–6919. IEEE.

Zheng, J.-Y.; Zhang, H.; Wang, L.; Qiu, W.; Zheng, H.-W.; and Zheng, Z.-M. 2024. Safely Learning with Private Data: A Federated Learning Framework for Large Language Model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 5293–5306.