Noise Added to Hidden States

The magnitude of noise injection significantly influences the stability and convergence speed of model training. Given that noise is directly injected into the hidden state h_A, the relative magnitude between the noise and h_A naturally affects the model's performance. To quantify this effect, we first fine-tune the FedSEA-LLaMA model on the MultiRC (Khashabi et al. 2018) with a batch size of 2 and a learning rate of 1×10^{-5}, analyzing the average absolute magnitude of all elements within the hidden state h_A. According to Figure 7, the average magnitude of the hidden state h_A is approximately 0.02 (num_{Blocks_A}=1). The average magnitude of the hidden state h_A increases with the number of blocks in $Blocks_A$.

Based on this observation, we subsequently inject noise at various scales around this value during training to investigate how the noise magnitude affects both the convergence speed and downstream task performance. Training is conducted for 5,000 steps with FP16 precision. We use LoRA with a rank of 8 and a scaling factor $\alpha = 16$. The learning rate is 1e-5, and the batch size is 2. Corresponding experimental results are presented in Figure 6a-6c and Table 2.

A horizontal comparison reveals that, under the same noise level, increasing the number of blocks in $Blocks_A$ consistently accelerates convergence. Table 2 indicates that 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. This can be explained by the average magnitude shown in Figure 7: as the depth of $Blocks_A$ increases, the ratio between the average absolute value of the hidden state h_A and the noise amplitude also increases, thereby improving the signal-to-noise ratio (SNR). This enhanced SNR is precisely what contributes to the better convergence and stronger robustness observed with a larger number of $Blocks_A$.

Prompt

Table 5 illustrates examples of cloze-style questions and corresponding answers derived from each task. (1)**ReCoRD** tests reading comprehension with common sense reasoning by resolving ambiguous entities in passages. (2)**COPA** assesses causal reasoning by choosing the more likely cause or effect for a given premise. (3)**WSC** focuses on pronoun disambiguation requiring deep contextual understanding. (4)**RTE** evaluates textual entailment by determining if one sentence follows, contradicts, or is unrelated to another. (5)**BoolQ** involves yes/no questions answered using evidence from context passages. (6)**WiC** tests word sense disambiguation by checking if a word has the same meaning in two sentences. (7)**CB** deals with entailment in complex sentences with nuanced reasoning. (8)**MultiRC** features multi-answer questions based on multi-sentence contexts, requiring integration of dispersed information.

Table 6 illustrates examples of fine-tuning templates of CoQA and XSum.

Parallel Training

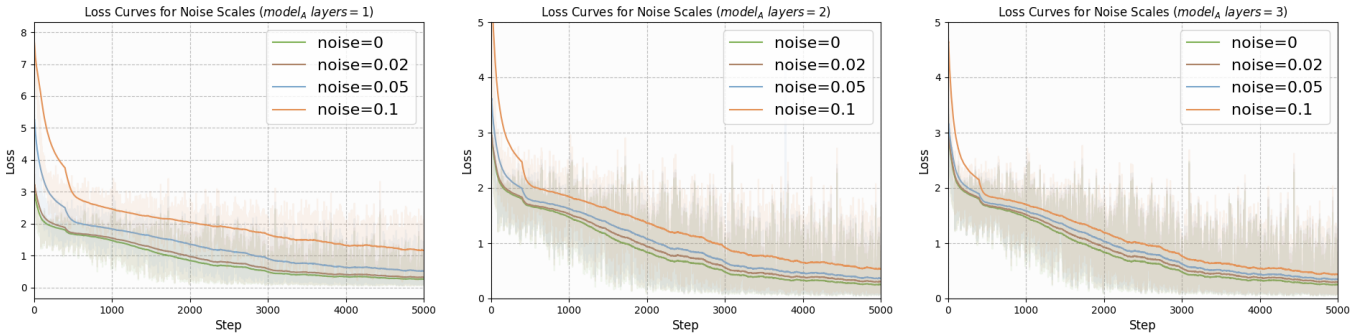
In multi-client settings, the FedSEA-LLaMA framework accommodates three training paradigms: sequential training, client-batch parallelism, and server-hierarchical parallelism, each designed for different system configurations and resource availabilities.

Sequential Training The client begins by passing hidden states through its input blocks $Blocks_A$ and forwards the result to the server. The server processes the input through $Blocks_B$ and returns the output states. The client then computes the loss with $Blocks_C$, sends the gradients to the server for model updates, and receives the updated gradients to adjust its input blocks. This process continues until all clients have completed their respective training phases for the current round. Although this method is inherently slower due to its serialized nature, it significantly reduces the computational burden on the server, making it a suitable choice for scenarios where server-side resources are limited.

Client-batch Parallelism The client-batch parallel approach enhances training throughput by expanding the effective batch size at the server, as shown in Figure 8a. During each training round, the server collects intermediate hidden states from all participating clients. If each of the M clients processes a local batch of size b, the hidden states sent to the server will each have the shape $(b, \textit{seq\_length}, \textit{hidden\_size})$. The server concatenates these hidden states along the batch dimension, forming a combined tensor of shape $(b \times M, \textit{seq\_length}, \textit{hidden\_size})$. This aggregated batch is then used to perform a joint forward and backward pass on the server model, allowing it to compute gradients more efficiently and update shared parameters.

Server-hierarchical Parallelism In a server-hierarchical parallel setup, the central server spawns multiple sub-server model instances, each dedicated to a specific client, illustrated in Figure 8b. These instances run in parallel, enabling simultaneous local training for all clients. Typically, after a fixed number of local updates, the central server synchronizes the training progress by aggregating model parameters across all sub-servers. This is usually done via weighted averaging of both server-side and client-side model parameters. This strategy enables significant improvements in training efficiency and scalability, especially in environments where the server has access to multiple GPUs or multi-threaded execution capabilities.

For communication cost analysis, we randomly select 1,000 samples from the ReCoRD dataset. All training uses a batch size of 1 with FP16 precision for efficiency. LoRA is applied with a rank of 8 and a scaling factor of 16, targeting attention modules during fine-tuning. The model is partitioned with Blocks_A=1 and Blocks_C=1. Each experiment is repeated five times, and we report the mean and standard deviation of the total communication time in Table 7. Results show that sequential training is the slowest, client-batch training is faster, and server hierarchical training is the fastest. Notably, when the server can handle larger batches, increasing the number of clients in client-batch training effectively reduces training time.



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

(b) Loss Curves for Noise Scales (Blocks_A=2, Blocks_C=1)

(c) Loss Curves for Noise Scales (Blocks_A=3, Blocks_C=1)

Figure 6: FedSEA-LLaMA: Noise Injection across different number of blocks in $Blocks_A$ on MultiRC dataset.

Dataset	Task	Cloze Question	Answers
ReCoRD	Question answering	[passage p] [cloze question q]	Answer candidates
COPA	Causal reasoning	"[choice c1]" or"[choice c2]"? [premise p], so [M].	c1/c2
WSC	Coreference resolution	[sentence s] The pronoun '*p*' refers to [M].	Noun n
RTE	Textual entailment	"[hypothesis h]"? [M] "[premise p]"	"yes"/"no"
BoolQ	Question answering	[passage p]. Question: q? Answer:[M].	"yes" / "no"
WiC	Word sense disambiguation	"[sentence s1]"/"[sentence s2]"Similar sense of [word w]? [M].	"yes"/"no"
CB	Textual entailment	"[hypothesis h]"? [M], "[premise p]"	"yes"/"no"/"maybe"
MultiRC	Question answering	[passage p]. Question: q? Is it [answer a]? [M].	"yes"/"no"

Table 5: Cloze questions and answers for the 8 SuperGLUE tasks

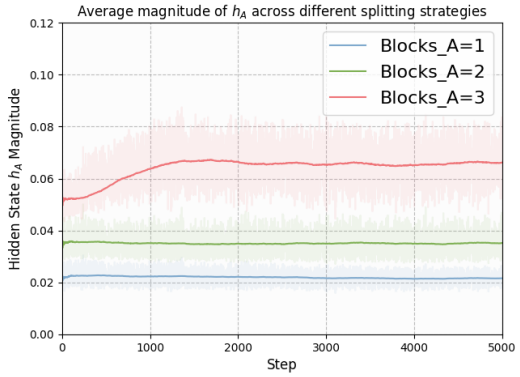


Figure 7: Avg magnitude of hidden state h_A (Blocks_C =1) on MultiRC dataset.

Parameter Setting

FedSEA-LLaMA and Baselines

Different hyperparameters are explored to obtain the best performance of FedSEA-LLaMA and Centralized LLaMA. Specifically, we tune the LoRA rank (r) and scaling factor (alpha) as pairs, trying combinations including (4, 32), (8, 32), (16, 32), (32, 32), and (8, 16). For the learning rate (lr), values of 1e-5, 2e-5, and 3e-5 are tested. The final parameter settings were selected based on the model's validation performance. The final parameters are shown in Table 8.

Different Noise Scales and Layers of $Block_A$ Experiment

We use a LoRA rank of 8 with a scaling factor of 16, a learning rate of 1e-5, and FP16 precision to investigate the effects of different noise scales and the number of blocks in $Blocks_A$ on downstream performance.

Ablation Study on Attention-Mask

The context length is set to 500, and the precision is set to FP16.

Ablation Study on KV Cache

The precision is set to FP16. Different context lengths are set to investigate the inference speed of FedSEA-LLaMA and FL-GLM.

Different Model Partition

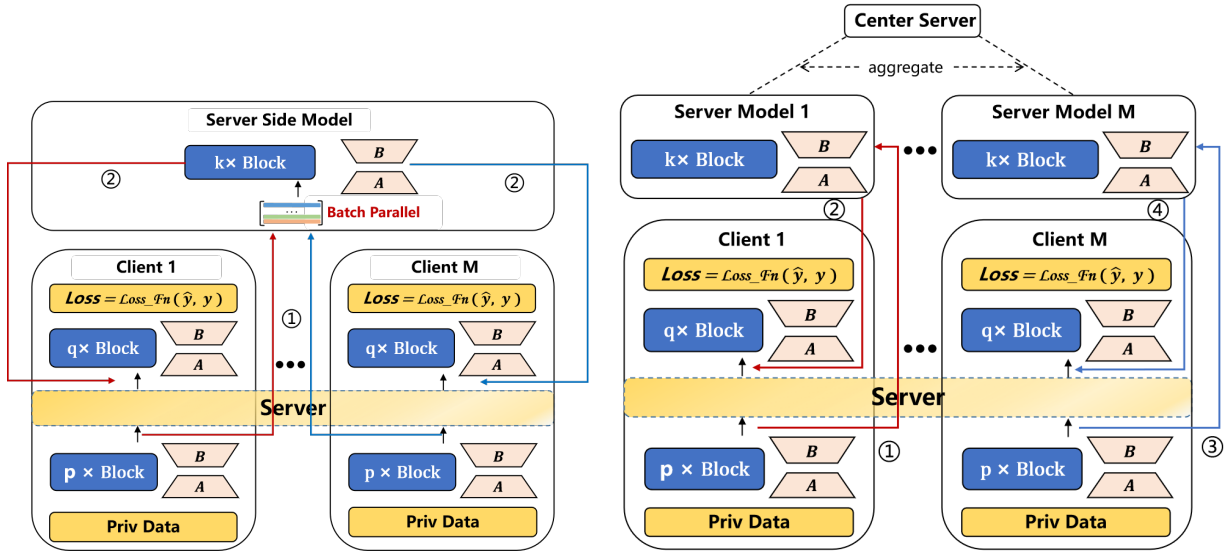
All experimental results were obtained after 8,000 training steps with LoRA rank set to 8, scaling factor set to 16, a batch size of 1, FP16 precision, and a learning rate of 1e-5.

Dataset	Task	Fine-tuning template
CoQA	conversational QA	$< s >$[INST] $<< SYS >>$You are a helpful assistant. You need to answer my question base on the provided story.$<< /SYS >>$ The story is {*story*}.My question is {*question*}.[/INST]{*answer*}$< /s >$
XSum	summarization generation	$< s >$[INST] $<< SYS >>$You are a helpful assistant. You need to summarize an abstract base on the provided passage.$<< /SYS >>$ The passage is {*passage*}.[/INST]{*summarization*} $< /s >$

Table 6: Fine-tuning templates for CoQA and XSum

Strategy	**Sequential training**	**Client-batch training**			**Server-hierachical training**			Centralized
num. of clients	2	2	4	8	2	4	8	-
time(s)	689.2±5.7	630.0±1.8	542.2±3.5	488.6±0.4	348.2±3.8	173.53±3.0	86.4±0.6	381.9±2.4

Table 7: Comparison of training time between different training strategies on ReCoRD dataset.



(a) Client-batch parallel training.

(b) Server-hierachical parallel training.

Figure 8: Multi-client Parallel Training of FedSEA-LLaMA. In (a), the server receives forward-pass states from multiple clients simultaneously, concatenates them into a batch, and performs joint forward propagation. Then, the server splits hidden state h_B and sends them back to different clients. In (b), each $client_i$ is paired with a corresponding sub_server_i. After several training steps, the central server aggregates all sub-servers' models using FedAvg and distributes the updated global server model back to each sub_server_i.

		CoQA	XSum	MultiRC	Record
FedSEA-LLaMA	Lora r & alpha	$(4, 32)$	$(16, 32)$	$(32, 32)$	$(32, 32)$
	lr	1e-4	3e-5	1e-5	2e-5
Centralized-LLaMA	Lora r & alpha	$(8, 16)$	$(32, 32)$	$(32, 32)$	$(8, 32)$
	lr	5e-5	3e-5	1e-5	3e-5

Table 8: Hyperparameters for FedSEA-LLaMA and Baselines