This is an early access version of a paper accepted for publication at the 3rd IEEE International Conference on Federated Learning Technologies and Applications (FLTA25).

FedLoRASwitch: Efficient Federated Learning via LoRA Expert Hotswapping and Routing

Joakim Flink
Mälardalen University
Västerås, Sweden
joakim.flink@mdu.se

Bostan Khan
Mälardalen University
Västerås, Sweden
bostan.khan@mdu.se

Masoud Daneshtalab
Mälardalen University
Västerås, Sweden
masoud.daneshtalab@mdu.se

***Abstract*—Adapting large language models (LLMs) to diverse tasks in federated settings is hindered by communication overhead, privacy constraints, and client heterogeneity. We introduce FedLoRASwitch, a framework that trains multiple specialised Low-Rank Adaptation (LoRA) experts on distributed clients *without* sharing raw data. A lightweight Transformer router selects the single most relevant expert from five categories (including a general category and a web-search augmented category) based on the highest confidence score per query, with a routing step latency of approximately $35\,\mathrm{ms}$ on a single A5000 GPU. The selected expert's LoRA weights (quantized to 4-bit, payload approx. $25\,\mathrm{MB}$) are then loaded *on-the-fly* into a frozen $0.5\,\mathrm{B}$-parameter base model. A retrieval-augmented web search capability is included, ensuring access to newly published knowledge. Experiments on CodeAlpaca (code generation), GSM8K (mathematics), Indian-History, and MuskumPillerum (general Q&A) show that four exemplar experts boost performance from $1.97\,\%$ to $16.22\,\%$ exact-match on GSM8K, while reducing communicated bytes per client per round by approximately $40\times$ compared with sending full 16-bit model parameters. Beyond accuracy, FedLoRASwitch enables cost-effective personalisation: institutions can host private, domain-specific LoRAs instead of repeatedly training full models. Previous work has validated federated learning, LoRA, and Mixture-of-Experts principles *individually*; FedLoRASwitch is, to our knowledge, the first system that combines these concepts to train multiple distinct LoRA experts in a federated manner and dynamically select one for inference. The framework offers a pragmatic path to high-performance, privacy-preserving, and resource-efficient multi-expert LLM adaptation.**

***Index Terms*—Federated Learning, Large Language Models, Mixture of Experts, Low-Rank Adaptation (LoRA), Parameter-Efficient Fine-Tuning (PEFT), Model Routing.**

I. Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities [1], yet adapting them to specific downstream tasks, particularly in distributed environments, remains a hurdle. Full fine-tuning is often computationally prohibitive in federated learning (FL) settings due to massive parameter counts and communication bottlenecks, compounded by data privacy constraints that prevent data centralization [2]. Parameter-Efficient Fine-Tuning (PEFT) methods like Low-Rank Adaptation (LoRA) [3] mitigate this by freezing the base LLM and training only small, low-rank adapter modules, drastically reducing trainable parameters and communication overhead.

While FL enables collaborative training on decentralized data, adapting a single global model to heterogeneous client data distributions (e.g., code vs. math vs. general knowledge) can lead to negative interference and suboptimal performance. Mixture-of-Experts (MoE) [4] models offer a solution by employing specialized subnetworks (experts) activated conditionally via a routing mechanism, increasing capacity without proportionally increasing computation. However, applying MoE directly in FL is challenging due to data locality constraints and the complexities of distributed expert training and aggregation. In this paper, we propose FedLoRASwitch, a framework that combines the strengths of FL, LoRA, and MoE principles in a novel, modular fashion. Instead of a monolithic MoE model, we train an example setup of multiple domain-specific LoRA experts (e.g., *Code*, *Math*, *Indian-history*, *General*) collaboratively across federated clients. A key innovation is a lightweight Transformer router [5] that learns to predict the most relevant expert for each query at inference time. Selected LoRA experts, being small modules, are then dynamically *hot-swapped*—merged with the frozen base model on-the-fly using an efficient LLM serving engine like TGI [6]. This allows for flexible, conditional computation tailored to each query while minimizing memory usage, as only active experts need to be loaded.

In our approach, we utilize the Flower federated learning system [7], where clients train specific experts based on data availability, ensuring focused updates. For aggregation, we employ the FedAvg [2] method, though it can be readily substituted with others, such as FedAdam [8]. Additionally, a web search mechanism allows the system to ingest external, up-to-date information for knowledge-intensive queries. This setup is particularly valuable for organizations such as businesses, schools, and hospitals, where privacy and personalization are crucial. By leveraging federated learning [2], LoRAs [3], and Mixture-of-Experts (MoE) [4] principles, FedLoRASwitch offers a practical and efficient approach to creating personalized and private language models. This collaborative training method not only enhances performance but also ensures cost-effectiveness in terms of computational power and communication savings.

Our contributions in this work are summarized as follows:

- **FedLoRASwitch Framework:** A collaborative learning system for training multiple LoRA experts on decentral-

This is an early access version of a paper accepted for publication at the 3rd IEEE International Conference on Federated Learning Technologies and Applications (FLTA25).

ized data and combining their usage dynamically via a router.

- **Routing Transformer:** An efficient and specialized router for low-latency expert selection.
- **LoRA Expert Hotswapping:** An inference mechanism leveraging a modern serving engine for on-demand LoRA merging, optimizing memory and flexibility.

The remainder of this paper details related work (Section II), the FedLoRASwitch methodology (Section III), results (Section IV), discussion (Section V), and conclusion (Section VI).

II. Related Work

A. Federated Learning for LLM Adaptation

FL enables privacy-preserving collaborative training [9]. Full federated training of LLMs faces communication and computation challenges. PEFT methods, especially LoRA [3], have emerged as solutions, drastically reducing the parameters exchanged in FL. Works like FlexLoRA [10] address client heterogeneity by allowing variable LoRA ranks. FedSA-LoRA [11] proposes selective aggregation, sharing only part of the LoRA matrices (e.g., "A" matrices) to enhance privacy and efficiency. FewFedPIT extends federated instruction tuning (FedIT) to better handle few-shot settings and privacy risks [12]. Our work builds on federated LoRA but focuses on managing task heterogeneity via multiple explicit experts and routing.

B. Mixture-of-Experts (MoE) and Routing

MoE models increase capacity efficiently by activating sparse expert subnetworks per input via a routing mechanism [4]. Sparsely-gated MoEs achieve massive effective capacity with modest computational overhead. Switch Transformers simplify this by routing each token to a single expert [13]. These are typically trained centrally. FedLoRASwitch adapts the MoE concept to FL using modular LoRA experts trained decentrally and combined at inference via a BERT-based router [5].

C. Dynamic Model Composition and Inference Efficiency

Efficient LLM serving systems like vLLM [14], Lorax [15], and TGI [6] optimize inference through techniques like advanced KV cache management, continuous batching, and model quantization. Crucially, they support dynamic loading and merging of adapters like LoRA. FedLoRASwitch utilizes TGI due to its straightforward setup and widespread adoption, though other engines with similar capabilities could be employed. It exploits this capability for its expert *hotswapping* mechanism, loading only necessary LoRA modules per request, thereby minimizing memory overhead compared to maintaining separate models or a full MoE architecture. We also incorporate a simple retrieval-augmentation mechanism, similar in spirit to RAG systems, using web search.

FedLoRASwitch uniquely integrates these threads: federated PEFT, modular MoE principles, and efficient dynamic inference serving.

III. Methodology

A. System Overview

FedLoRASwitch operates in two primary phases: a federated training phase for developing specialized LoRA experts, and a dynamic inference phase where these experts are utilized. During training (illustrated in Figure 1), multiple clients collaboratively train various domain-specific LoRA experts (e.g., for *Code*, *Math*, *Indian-history*, *General* queries) using their local data. This process is coordinated by a central server which employs the FedAvg protocol [2]. A key aspect of our framework is the dynamic scheduling of expert training by the server; the decision to train a particular expert in a given round is based on factors such as the availability of new relevant data reported by clients and the time elapsed since an expert's last update (staleness).

During inference (detailed in Figure 2), an incoming query is first processed by a lightweight router. This router selects the LoRA expert deemed most relevant to the query. The weights of this selected LoRA module are then merged on-the-fly with the frozen base LLM. The combined, temporarily specialized model then generates the response. The system also incorporates an extended functionality where selection of a designated *WebSearch* expert by the router triggers a retrieval-augmented web search. This allows the system to incorporate up-to-date external information for knowledge-intensive queries, with the search results augmenting the input to the *General* LoRA expert. This separation of the web search functionality addresses potential latency and API cost considerations.

B. Architectural Components

The FedLoRASwitch framework is built upon several key architectural components:

1) Base Language Model: We employ Qwen2.5-0.5B [16], a capable $0.5\,\mathrm{B}$-parameter LLM, as the foundational model. This base model is kept frozen throughout both training and inference phases, meaning its original weights are not altered. It is operated with 8-bit quantization (via `bitsandbytes` in TGI)..

2) Domain-Specific LoRA Experts: Low-Rank Adaptation (LoRA) modules serve as lightweight, specialized experts. In our experimental setup, we developed four such LoRA experts (rank $r = 32$), each targeting a specific domain: *Code* (trained on CodeAlpaca [17]), *Math* (trained on GSM8K [18]), *Indian-history* (available on Hugging Face [19]), and *General* (trained on MuskumPillerum dataset [20]). These LoRA modules are quantized to 4-bit for efficient transfer and storage. Each module introduces only a few million trainable parameters, which are applied to specific layers (typically query and value projection matrices in attention mechanisms) of the base LLM.

3) Expert Selection Router: A lightweight Transformer model, specifically a fine-tuned `albert-base-v1` [5], functions as the expert selection router. Its role is to classify incoming user queries into one of five predefined categories, each corresponding to a LoRA expert or a specialized processing path: *Code*, *Math*, *Indian-history*, *General*, and *WebSearch*.

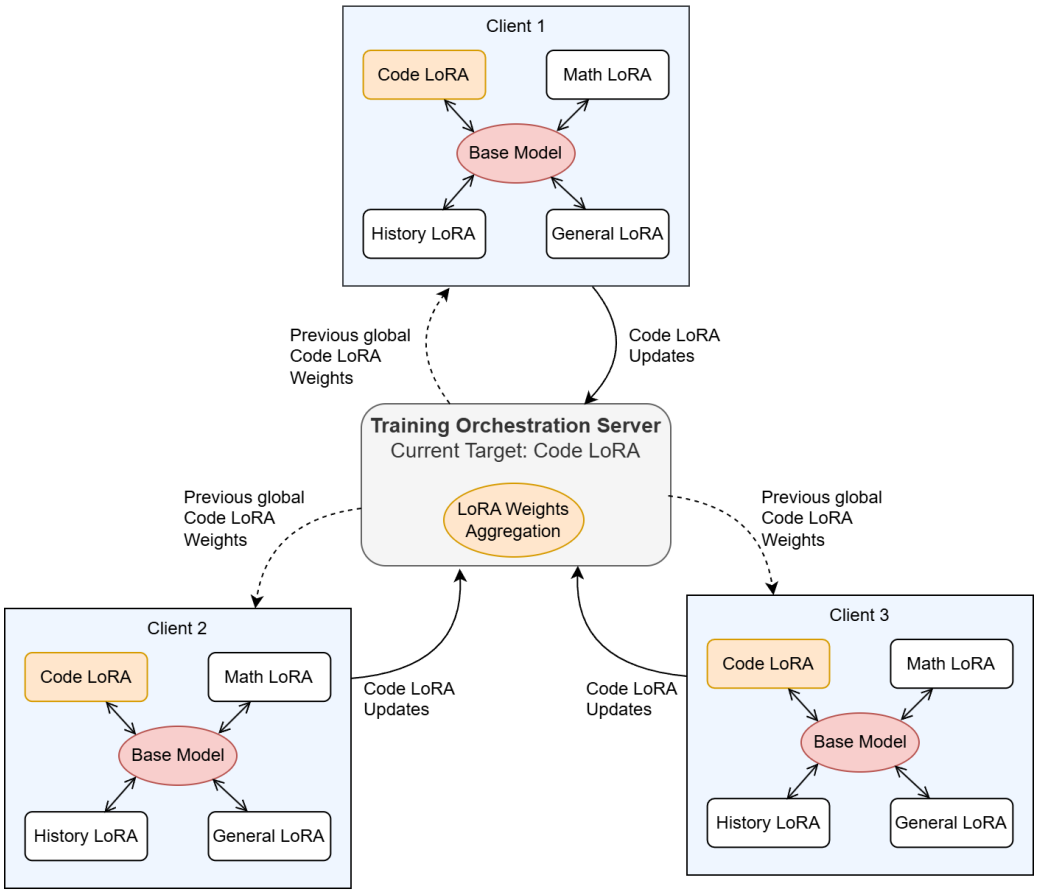

Fig. 1. Illustration of a single federated training round in FedLoRASwitch, coordinated by the Training Orchestration Server. For the current round, the *Code LoRA* expert is targeted for training. The server dynamically selects which expert to train based on factors like data availability and the time since the expert was last updated. Clients with relevant data fine-tune these weights locally and send their resulting *Code LoRA* updates to the server, where they are aggregated to produce an improved global *Code LoRA* expert.

C. Phase 1: Federated Training of Experts

The development of specialized LoRA experts and the expert selection router is achieved through a federated learning process, detailed in Algorithm 1 and illustrated in Figure 1.

1) Dynamic Training Orchestration: The central server coordinates the training process. It dynamically schedules which LoRA expert (E_{target}) is to be trained in any given set of federated rounds (R_{total}). This decision, detailed as part of Algorithm 1 (lines 3-7), is based on a combination of factors: primarily the availability of new, relevant data on client devices (which clients might report in a privacy-preserving manner, e.g., counts of new samples per expert domain) and the staleness of each expert (i.e., the time elapsed or number of rounds since its last update). The server calculates a priority score for each expert based on these factors and selects the one with the highest score for the current training cycle.

2) Client-Side LoRA Fine-tuning: For each training round $r \in [1, R_{total}]$ for a targeted expert E_{target}: The server selects a subset of clients (S_r) to participate. Clients $k \in S_r$ participate only if they possess a sufficient amount of relevant domain data (experimentally set at over 2000 samples for our setup). Each selected client k receives the current global weights $w_r^{E_{target}}$ of the targeted LoRA expert. Using their local, labeled dataset D_k, clients fine-tune these weights for a number of local epochs, resulting in updated local LoRA expert weights $w_{r+1,k}^{E_{target}}$.

3) Server-Side Aggregation and Distribution: The updated local LoRA expert weights $w_{r+1,k}^{E_{target}}$ (or their deltas) computed by participating clients are transmitted to the server. Each client sends approximately 25 MB of data per round, which includes the 4-bit LoRA parameters and any necessary communication overhead. The server then aggregates these updates using the FedAvg algorithm [2]: $w_{r+1}^{E_{target}} = \sum_{k \in S_r} \frac{n_k}{N} w_{r+1,k}^{E_{target}}$, where n_k is the number of samples on client k and $N = \sum_{k \in S_r} n_k$. The resulting updated global LoRA expert model $w_{r+1}^{E_{target}}$ is then distributed back to the clients for the next round of training or for use in inference. In our experiments, training was conducted for 100 rounds per expert with $K = 3$ clients, taking approximately 2 h per expert. The aggregated LoRA expert updates handled by the

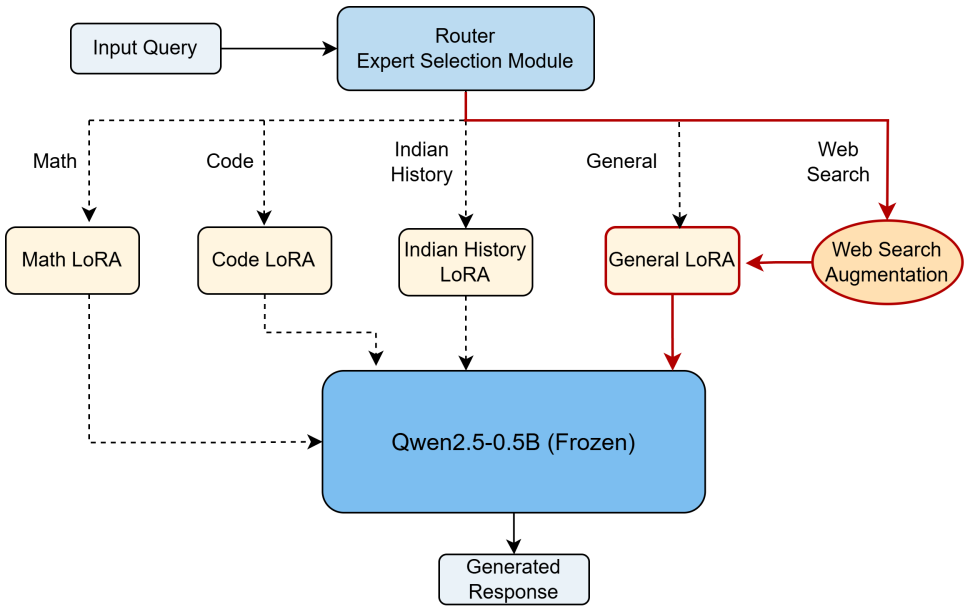

Fig. 2. FedLoRASwitch inference pipeline. The solid red line illustrates an example flow where the router selects the *Web Search* option, leading to query augmentation before processing by the *General* LoRA expert.

server amounted to approximately 25 MB per round per expert, totaling about 2.5 GB of data processed by the server for the complete training of one expert.

4) Router Training: The expert selection router (M_{router}) is trained centrally after the initial set of LoRA experts has been developed (this can also be done periodically). This supervised training process uses a labeled dataset $D_{router} = \{(q_i, y_i)\}$, where q_i are sample queries and y_i are their corresponding true expert categories (*Code*, *Math*, *Indian-history*, *General*, or *WebSearch*). For our experiments, D_{router} was constructed from a small, unused portion of the other domain-specific training datasets, augmented with synthetic data generated by a GPT-3 series model. Specifically, we used 400 samples from each of the four primary expert categories. We supplemented this with 639 synthetic data samples: 400 focused on *WebSearch* queries, 98 for *General*, 81 for *Math*, and approximately 30 each for *Code* and *Indian-history*. In practical deployments, D_{router} would ideally be derived from real user interactions or client-provided examples. The router M_{router} is trained by minimizing a cross-entropy loss function, with early stopping to prevent overfitting.

D. Phase 2: Dynamic Inference with Expert Hotswapping

Once the LoRA experts and the router are trained, the system is ready for inference, as depicted in Figure 2 and outlined in Algorithm 2. The inference pipeline is designed for dynamic specialization per query.

1) Query Processing and Expert Selection: When an input query q is received, it is first processed by the expert selection router M_{router}. The router classifies the query and outputs confidence scores for each of the five predefined expert categories. The expert category C_{sel} with the highest confidence score is chosen, and its corresponding LoRA expert E_{sel} (or the *WebSearch* processing path) is selected.

2) WebSearch-Augmented Response Generation: If the selected category C_{sel} is *WebSearch*, a dedicated web search function is triggered using query q. This function retrieves relevant, up-to-date information I_{web} from external sources. The original query q is then augmented with I_{web} to form $q' = q + I_{web}$. This augmented query q' is subsequently processed by the *General* LoRA expert ($E_{sel} = E_{General}$) to generate a contextually enriched response.

3) On-the-fly LoRA Adaptation and Response Generation: For queries routed to a domain-specific LoRA expert E_{sel} (e.g., *Code*, *Math*, *Indian-history*, or *General* directly, or *General* after web search augmentation):

1) **LoRA Merging:** The weights $w^{E_{sel}}$ of the selected LoRA expert are retrieved from storage (payload approx. 25 MB each, due to 4-bit quantization). These are merged on-the-fly with the weights of the frozen base LLM M_{base}, creating a temporarily specialized model $M_{adapted} = M_{base} + w^{E_{sel}}$.
2) **Generation:** $M_{adapted}$ processes the input query (q or q') and generates the response A.
3) **Reset:** After response generation, the LoRA weights $w^{E_{sel}}$ are unmerged from M_{base}, restoring it to its original frozen state, ready for the next query.

This hotswapping mechanism allows the system to dynamically adapt to the specific nature of each query using a shared base model and lightweight, swappable expert modules, optimizing memory usage and computational resources.

IV. Experimental Evaluation

This section details the experimental setup used to evaluate FedLoRASwitch and presents the results obtained across

Algorithm 1 FedLoRASwitch: Federated Training Phase

Server Initializes: Global LoRA experts E_j for $j = 1, ..., N$; Router M_{router} (can be pre-trained or trained after initial expert versions); Staleness counter $T_j = 0$ for each expert E_j.
for each training cycle $c = 1, \ldots, C_{total}$ **do**
  {Server Selects Target Expert E_{target}}
  for each expert E_j **do**
    Server requests/receives data availability metric DA_j from clients (e.g., total new samples for E_j).
    Calculate priority score $P_j = \alpha \cdot DA_j + \beta \cdot T_j$.
  end for
  $E_{target} = \text{argmax}_{E_j} P_j$.
  Reset staleness for selected expert: $T_{target} = 0$.
  Increment staleness for other experts: $T_j = T_j + 1$ for $j \neq target$.
  for each round $r = 1, \ldots, R_{total}$ for E_{target} **do**
    Server selects subset of clients S_r.
    for each client $k \in S_r$ **in parallel do**
      Client k downloads $w_r^{E_{target}}$ (current global LoRA weights for E_{target}).
      Client k fine-tunes $w_r^{E_{target}}$ on local data D_k to get $w_{r+1,k}^{E_{target}}$.
      Client k sends $w_{r+1,k}^{E_{target}}$ (or $\Delta w_k = w_{r+1,k}^{E_{target}} - w_r^{E_{target}}$) to server.
    end for
    Server aggregates updates: $w_{r+1}^{E_{target}} = \text{FedAvg}(\{w_{r+1,k}^{E_{target}}\}_{k \in S_r})$.
    Server updates global model for E_{target} with $w_{r+1}^{E_{target}}$.
  end for
end for
Router Training (Centralized, Post-Expert Training / Periodic):
Server collects/creates labeled dataset $D_{router} = \{(q_i, y_i)\}$.
Train M_{router} on D_{router} using cross-entropy loss and early stopping.

Algorithm 2 FedLoRASwitch: Dynamic Inference Phase

Input: User query q.
Server Components: Trained LoRA experts $E_1, ..., E_N$; Router M_{router}; Base LLM M_{base}.
Router M_{router} processes q: $C_{sel} = \text{argmax}_{C_i} P(C_i|q)$.
Let E_{sel} be the expert corresponding to C_{sel}.
if C_{sel} is *WebSearch* **then**
  $I_{web} = \text{PerformWebSearch}(q)$.
  $q' = \text{AugmentQuery}(q, I_{web})$.
  $E_{sel} = E_{General}$.
  Query for generation becomes q'.
else
  Query for generation remains q.
end if
Retrieve LoRA weights $w^{E_{sel}}$ for the selected expert E_{sel}.
$M_{adapted} = \text{Merge}(M_{base}, w^{E_{sel}})$. {On-the-fly hotswap}
Generate response $A = M_{adapted}$(query for generation).
Unmerge$(M_{base}, w^{E_{sel}})$. {Reset base model}
Output: Response A.

various performance metrics and efficiency benchmarks.

A. Experimental Setup

1) Datasets: We utilized several datasets to train and evaluate the domain-specific LoRA experts:

- **Code Generation (*Code* expert):** CodeAlpaca-20k dataset (using approximately half, 10k samples) [17]. Performance was assessed qualitatively using a Win-Lose-Tie (WLT) methodology.
- **Indian History Q&A (*Indian-history* expert):** BashitAli/Indian_history dataset on Hugging Face [19]. Performance was also assessed qualitatively via WLT.
- **Mathematical Reasoning (*Math* expert):** GSM8K dataset [18]. Performance was measured using the Exact Match (EM) metric.

The *General* expert was trained on the MuskumPillerum dataset [20].

2) Evaluation Metrics: Our evaluation employed a range of metrics tailored to different aspects of the system:

- **Exact Match (EM):** Used for the GSM8K dataset [18] to evaluate mathematical reasoning, where only the final numerical answer's correctness is considered.
- **Win-Lose-Tie (WLT):** A qualitative comparison framework used for code generation and Indian-history queries. Five SOTA LLMs served as judges, independently assessing whether FedLoRASwitch or the base model provided a more correct or contextually appropriate answer, or if their performance was comparable. For this qualitative assessment, we focused on code generation (using CodeAlpaca) and Indian history Q&A. These diverse topics were selected to demonstrate FedLoRASwitch's ability to enhance answer quality for both common, structured query types (code) and more specialized, knowledge-intensive domains (Indian history). The complete lists of questions used are provided in Appendix B of our supplementary material, available at https://doi.org/10.5281/zenodo.15551664.
- **Router Accuracy:** Measured on a held-out set of one hundred samples per expert category. These samples were not used during router training. Accuracy is the percentage of queries correctly routed to their intended expert category.
- **Inference Latency:** The time taken to process a query and generate a response.
- **VRAM Usage:** Memory footprint of the models during operation.
- **Communication Cost:** The amount of data transmitted during federated training rounds.

For the WLT evaluation on Indian-history, the targeted set comprised six questions.

3) FedLoRASwitch Configuration: The base LLM for our FL experiments was Qwen2.5-0.5B-Instruct [16] with 8 bit quantization. While this specific model was used, the FedLoRASwitch framework is designed to be compatible with various base models and sizes. The LoRA experts were configured with a rank $r = 32$ and quantized to 4-bits for communication and storage. For response generation, all LoRAs had an output token limit of 256, except for the *Code* LoRA, which was set to 512 tokens to accommodate longer code snippets. Specific pre-prompts were used for each LoRA expert to prime them for their respective domains. While most were straightforward to guide the AI to the correct focus area, the pre-prompt for the *Code* expert was intentionally more detailed to illustrate the use of advanced prompting techniques. The full list of pre-prompts is available in Appendix A of our supplementary material, accessible online at https://doi.org/10.5281/zenodo.15551664.

B. Performance Evaluation of Specialized Experts

This subsection presents the core performance results of FedLoRASwitch, demonstrating the effectiveness of using specialized LoRA experts.

1) Mathematical Reasoning (GSM8K): The GSM8K dataset served as the primary benchmark for quantitatively evaluating specialized task performance. For a clear interpretation of the results in Table I, it is essential to note the consistent evaluation conditions: all listed models, including our foundational $0.5\,\mathrm{B}$ Qwen2.5-Instruct model, were operated with 8-bit quantization. This was applied in real-time during inference using the `bitsandbytes` library within the Text Generation Inference (TGI) framework. This practical, efficiency-oriented setup impacts performance on precision-sensitive tasks like GSM8K, generally resulting in lower scores than those achieved with higher-precision (e.g., FP16/BF16) evaluations common in standard model leaderboards.

Operating under these 8-bit quantized conditions, our $0.5\,\mathrm{B}$ base model (detailed as "Qwen 2.5 instruct (Base Model)" in Table I) achieved an exact match score of $1.97\,\%$. FedLoRASwitch, when activating its specialized *Math* LoRA expert in conjunction with this same 8-bit quantized base, demonstrated a striking improvement, reaching $16.22\,\%$ exact match. This constitutes a more than 8-fold increase in performance over its non-specialized counterpart (see Table I), underscoring the substantial advantage of targeted LoRA expertise for complex reasoning, even within a quantized framework. Notably, with this specialization, our $0.5\,\mathrm{B}$ parameter system outperforms other larger models in this comparison, such as the Qwen2.5-Instruct 1.5B and Smolm 1.7B, when all are subjected to the same 8-bit real-time quantization constraints. This highlights the power of focused adaptation to significantly enhance capability within resource-constrained environments.

2) Coding Task Performance (WLT): In the qualitative evaluation of coding tasks using the WLT methodology (Table II), FedLoRASwitch, equipped with its specialized *Code* expert LoRA, demonstrated superior performance. Across the AI evaluators, it secured more "wins" (14 total points) compared to the base model (5 total points). The router successfully directed all six coding questions to the *Code* expert, which was crucial for this enhanced performance. Evaluators noted that FedLoRASwitch often provided more complete, runnable programs or utilized efficient libraries.

TABLE I
GSM8K PERFORMANCE UNDER REAL-TIME 8-BIT QUANTIZATION. MODEL SIZE IN BILLIONS (B) OF PARAMETERS

Name	Size (B)	GSM8K (%)
Qwen 2.5 instruct	7	24.56
Qwen 2.5 instruct	3	17.36
Qwen 2.5 instruct	1.5	4.09
Qwen 2.5 instruct (Base Model)	0.5	1.97
Smolm [21]	1.7	2.96
FedLoRASwitch (0.5B base)	**0.5**	**16.22**

TABLE II
COMPARATIVE EVALUATION OF BASE MODEL VS. FEDLORASWITCH (W-L-T FOR CODING QUESTIONS - 6 QUESTIONS)

Evaluating AI	**Base Wins**	**Tie**	**FedLoRASwitch Wins**
Grok 3 [22]	2	2	2
Claude 3.7 sonnet [23]	0	1	5
ChatGPT o3 [24]	1	3	2
DeepSeek 3 [25]	0	3	3
Gemini 2.5 Pro [26]	2	2	2
Total Points	**5**	**10**	**14**

3) Indian History Query Performance (WLT): For Indian-history questions (Table III), FedLoRASwitch again won the majority of "battles" according to all evaluating AIs, achieving 19 total points against the base model's 3 points. This reinforces the benefit of an expert LoRA when a router trained on relevant data selects the appropriate expert. In this test, the router correctly identified the *Indian History* expert for five out of six questions. For one ambiguous question ("What are the four separate collections included in the Mantra category, and what is their significance?"), the router selected the *General* expert, as this category received the highest confidence score. FedLoRASwitch's answers were often noted as better structured.

TABLE III
COMPARATIVE EVALUATION OF BASE MODEL VS. FEDLORASWITCH (W-L-T FOR INDIAN-HISTORY QUESTIONS - 6 QUESTIONS)

Evaluating AI	**Base Wins**	**Tie**	**FedLoRASwitch Wins**
DeepSeek 3 [25]	0	2	4
Grok 3 [22]	0	1	5
ChatGPT (o3) [24]	1	2	3
Claude 3.7 sonnet [23]	1	2	3
Gemini 2.5 Pro [26]	1	1	4
Total Points	**3**	**8**	**19**

C. Router Performance

The effectiveness of FedLoRASwitch hinges on the router's ability to accurately direct queries to the appropriate expert. Table IV summarizes the router's accuracy on a held-out test set.

TABLE IV
ROUTER ACCURACY BY TARGET EXPERT (ON 100 HELD-OUT SAMPLES PER CATEGORY)

Target Expert	Accuracy (%)
Indian History	94
General	95
Math	100
Code	98
Websearch	97

The router demonstrated high accuracy across all categories, achieving perfect routing for *Math* queries. The average processing time for expert selection was approximately $35.6\,\mathrm{ms}$. Misclassifications typically occurred with ambiguous queries, which were often routed to the *General* expert, or when broadly applicable terms led to selection of the *Websearch* category. These instances suggest areas for future refinement in router training data or logic.

D. System Efficiency Analysis

Beyond task performance, we evaluated FedLoRASwitch on key efficiency metrics:

- **Latency:** Including routing and LoRA merging, FedLoRASwitch averaged approximately $1.5\,\mathrm{s}$ per query for 100 output tokens. This was slightly slower than the base model's inference alone (approx. $80\,\mathrm{ms}$ overhead for routing and merging). For the 6 Indian History WLT questions, the base model took 79.9s, while FedLoRASwitch took 86.4s (averaging 14.4s per question for FedLoRASwitch). The higher per-question latency in this specific test is likely due to the generation of longer and more detailed responses than the 100-token benchmark.
- **VRAM Usage:** The system utilized approximately $14.77\,\mathrm{GB}$ of VRAM, compared to $14.51\,\mathrm{GB}$ for the base model alone. This minor increase is a significant advantage for supporting multiple dynamic specializations compared to hosting multiple full models. Hotswapping LoRAs offers memory savings over restarting models with different configurations.
- **Communication Cost:** Each client transmitted approximately $25\,\mathrm{MB}$ of 4-bit LoRA updates per round. This represents an approximate $\mathbf{40}\times$ reduction compared to sending full 16-bit parameters for the $0.5\,\mathrm{B}$ base model (estimated at $1000\,\mathrm{MB}$ or $1\,\mathrm{GB}$ per client per round). Further reductions are possible via compression or selective updates.

The inclusion of web search also successfully augmented answers for factual, out-of-distribution queries, showcasing the system's extensibility.

V. DISCUSSION

Strengths: FedLoRASwitch demonstrates effective specialization by dedicating experts to domains while leveraging a shared base model. The approach inherently supports privacy and decentralization as raw data remains local. It achieves significant efficiency gains in communication (uplink from clients) and memory compared to full-scale fine-tuning. While inference latency is slightly higher than the base model alone due to routing and merging, the performance gains on specialized tasks can be substantial. The modular design promotes adaptability, allowing easy addition or updating of experts, and provides interpretability regarding which expert is used. This makes it feasible to run unique experts directed towards specific fields with small models on hardware-constrained machines.

Scalability: Adding more clients seems feasible due to LoRA's training efficiency, potentially requiring stratified sampling. Adding more experts is architecturally possible, with minimal impact on router inference time and manageable memory overhead, as serving engines like TGI support loading multiple active LoRAs simultaneously (e.g., TGI supports at least 32, and Lorax up to 100 [27], [28]).

Limitations and Future Work: While FedLoRASwitch shows promise, key limitations guide future work. Performance is constrained by the $0.5\,\mathrm{B}$ base model, as larger models amplify FL challenges. Router accuracy is crucial, since misclassifications lead to suboptimal responses; future work includes adaptive expert creation and hierarchical routing. Scaling with the current FedAvg necessitates advanced aggregation strategies to address heterogeneity and convergence [29]. While FL with LoRA offers baseline privacy, LoRA updates lack formal differential privacy, making integration of guarantees like DP or secure aggregation vital [30]. Other promising directions include exploring fine-grained experts, user personalization, RAG integration, and merging multiple LoRAs simultaneously (e.g., combining domain and stylistic experts). A slight inference overhead from routing/merging is a consideration for high-throughput use, though modern serving engines offer mitigation.

VI. CONCLUSION

In response to the pressing challenges of adapting Large Language Models in distributed, privacy-sensitive environments, we introduce FedLoRASwitch. This novel framework uniquely synergizes federated learning, the efficiency of Low-Rank Adaptation, and the targeted intelligence of a Mixture-of-Experts approach, all orchestrated via a lightweight dynamic router. By enabling the collaborative training of multiple specialized LoRA experts on decentralized client data—without sharing raw information—and dynamically hotswapping the most relevant expert into a frozen base model at inference time, FedLoRASwitch delivers substantial performance gains. Our experiments demonstrate a significant uplift in task-specific accuracy, such as an over 8-fold improvement on the GSM8K benchmark (from $1.97\,\%$ to $16.22\,\%$ exact match),

alongside an approximate $\mathbf{40}\times$ reduction in communication overhead per client per round.

FedLoRASwitch is, to our knowledge, the first system to holistically integrate these principles, offering a pragmatic and scalable solution for creating personalized, high-performing, and resource-efficient LLMs. It underscores the power of modularity and conditional computation, paving the way for more accessible and adaptable AI systems across diverse domains and resource constraints. This work not only validates a potent combination of existing techniques but also offers a tangible blueprint for the future of decentralized, specialized, and privacy-preserving intelligent applications, enabling organizations to harness the power of LLMs securely and efficiently.

REFERENCES

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, arXiv:2005.14165. [Online]. Available: https://arxiv.org/abs/2005.14165

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, ser. PMLR, vol. 54, 2017, pp. 1273–1282, arXiv:1602.05629. [Online]. Available: https://arxiv.org/abs/1602.05629

[3] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations (ICLR)*, Apr. 2022, poster.

[4] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," arXiv preprint arXiv:1701.06538, 2017. [Online]. Available: https://arxiv.org/abs/1701.06538

[5] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," in *International Conference on Learning Representations (ICLR)*, 2020, arXiv:1909.11942. [Online]. Available: https://arxiv.org/abs/1909.11942

[6] Hugging Face, "Text generation inference," https://huggingface.co/docs/text-generation-inference/en/index, accessed: May 11, 2025.

[7] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," arXiv preprint arXiv:2007.14390, 2020. [Online]. Available: https://arxiv.org/abs/2007.14390

[8] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," in *International Conference on Learning Representations (ICLR)*, 2021, arXiv:2003.00295. [Online]. Available: https://arxiv.org/abs/2003.00295

[9] F. M. Awaysheh, M. Alazab, S. Garg, D. Niyato, and C. Verikoukis, "Big data resource management & networks: Taxonomy, survey, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2098–2130, 2021.

[10] J. Bai, D. Chen, B. Qian, L. Yao, and Y. Li, "Federated fine-tuning of large language models under heterogeneous tasks and client resources," arXiv preprint arXiv:2402.11505, 2024. [Online]. Available: https://arxiv.org/abs/2402.11505

[11] P. Guo, S. Zeng, Y. Wang, H. Fan, F. Wang, and L. Qu, "Selective aggregation for low-rank adaptation in federated learning," arXiv preprint arXiv:2410.01463, 2024. [Online]. Available: https://arxiv.org/abs/2410.01463

[12] Z. Zhang, J. Zhang, J. Huang, L. Qu, H. Zhang, Q. Wang, X. Zhou, and Z. Xu, "FewFedPIT: Towards privacy-preserving and few-shot federated instruction tuning," arXiv preprint arXiv:2403.06131, 2024. [Online]. Available: https://arxiv.org/abs/2403.06131

[13] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Journal of Machine Learning Research*, vol. 23, no. 120, pp. 1–39, 2022.

[14] vLLM Project, "vLLM: A high-throughput and memory-efficient inference and serving engine for LLMs," https://docs.vllm.ai/en/latest/index.html, accessed: May 11, 2025.

[15] Predibase, "LoRAX: Multi-LoRA inference server that scales to 1000s of fine-tuned LLMs," GitHub Repository. https://github.com/predibase/lorax, 2023.

[16] Q. Team, "Qwen2.5: A party of foundation models," September 2024. [Online]. Available: https://qwenlm.github.io/blog/qwen2.5/

[17] S. Sahil, "CodeAlpaca-20k: An instruction-following dataset for code generation," https://huggingface.co/datasets/sahil2801/CodeAlpaca-20k, 2023.

[18] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," arXiv preprint arXiv:2110.14168, 2021, dataset available at https://huggingface.co/datasets/openai/gsm8k. [Online]. Available: https://arxiv.org/abs/2110.14168

[19] BashitAli, "Indian-history," Hugging Face Datasets. https://huggingface.co/datasets/BashitAli/Indian_historyzz, accessed: May 11, 2025.

[20] A. G. Ravi, "General-Knowledge: A dataset of general facts and reasoning questions," https://huggingface.co/datasets/MuskumPillerum/General-Knowledge, 2023.

[21] L. B. Allal, A. Lozhkov, E. Bakouch, L. von Werra, and T. Wolf, "Smollm - blazingly fast and remarkably powerful," 2024. [Online]. Available: https://huggingface.co/blog/smollm

[22] xAI, "Announcing Grok-3: The Next Generation of AI for Everyone," https://x.ai/grok, 2025, accessed: [Current Date, e.g., 30 May 2025].

[23] Anthropic, "Claude 3.7 Sonnet," https://www.anthropic.com/news/claude-3-7-sonnet, 2025, accessed: 30 May 2025.

[24] OpenAI, "Introducing OpenAI o3 and o4-mini," https://openai.com/index/introducing-o3-and-o4-mini/, 2025, accessed: 30 May 2025.

[25] DeepSeek-AI, A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Guo, D. Yang, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, B. Han, H. Xu, H. Wang, H. Zhang, H. Ding, H. Xin, H. Gao, H. Li, H. Qu, J. L. Cai, J. Liang, J. Guo, J. Ni, J. Li, J. Wang, T. Wang, T. Pei, T. Sun, W. L. Xiao, W. Zeng, W. Zhao, W. An, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, X. Q. Li, X. Jin, X. Wang, X. Yu, X. Chen, X. Sun, X. Li, Y. Yu, Y. Ma, Y. Hou, Y. Deng, Y. He, Y. Liu, Y. Wang, Z. S. Wei, Z. Li, Z. Ma, Z. Cai, Z. Peng, and Z. Huang, "Deepseek-v3 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2412.19437

[26] Google DeepMind and Google Cloud, "Gemini 2.5 Pro: Our most advanced reasoning Gemini model," https://gemini.google.com, 2025, accessed: 30 May 2025.

[27] D. Thomas, D. Maniloff, and D. Holtz, "TGI Multi-LoRA: Deploy once, serve 30 models," Hugging Face Blog, Jul. 2024, accessed: May 11, 2025. [Online]. Available: https://huggingface.co/blog/multi-lora-serving

[28] T. Addair, G. Angus, M. Saleh, and W. Abid, "LoRAX: Open source LoRA serving framework for LLMs," Predibase Blog, Nov. 2023, accessed: May 11, 2025. [Online]. Available: https://predibase.com/blog/lorax-the-open-source-framework-for-serving-100s-of-fine-tuned-llms-in

[29] L. Nascimento, F. M. Awaysheh, and S. Alawadi, "Data skew in federated learning: An experimental evaluation on aggregation algorithms," in *2024 2nd International Conference on Federated Learning Technologies and Applications (FLTA)*, 2024, pp. 162–170.

[30] M. Tahir, T. Mawla, F. Awaysheh, S. Alawadi, M. Gupta, and M. Intizar Ali, "Securefedprom: A zero-trust federated learning approach with multi-criteria client selection," *IEEE Journal on Selected Areas in Communications*, vol. 43, no. 6, pp. 2025–2041, 2025.