NEURAL INFORMATION PROCESSING SYSTEMS

LLM at Network Edge: A Layer-wise Efficient Federated Fine-tuning Approach

Jinglong Shen[1], Nan Cheng[1], Wenchao Xu[2], Haozhao Wang[3], Yifan Guo[1], Jiajie Xu[1]

[1] School of Telecommunications Engineering, Xidian University

[2] Department of Computing, The Hong Kong Polytechnic University

[3] School of Computer Science and Technology, Huazhong University of Science and Technology

西安電子科技大學 XIDIAN UNIVERSITY

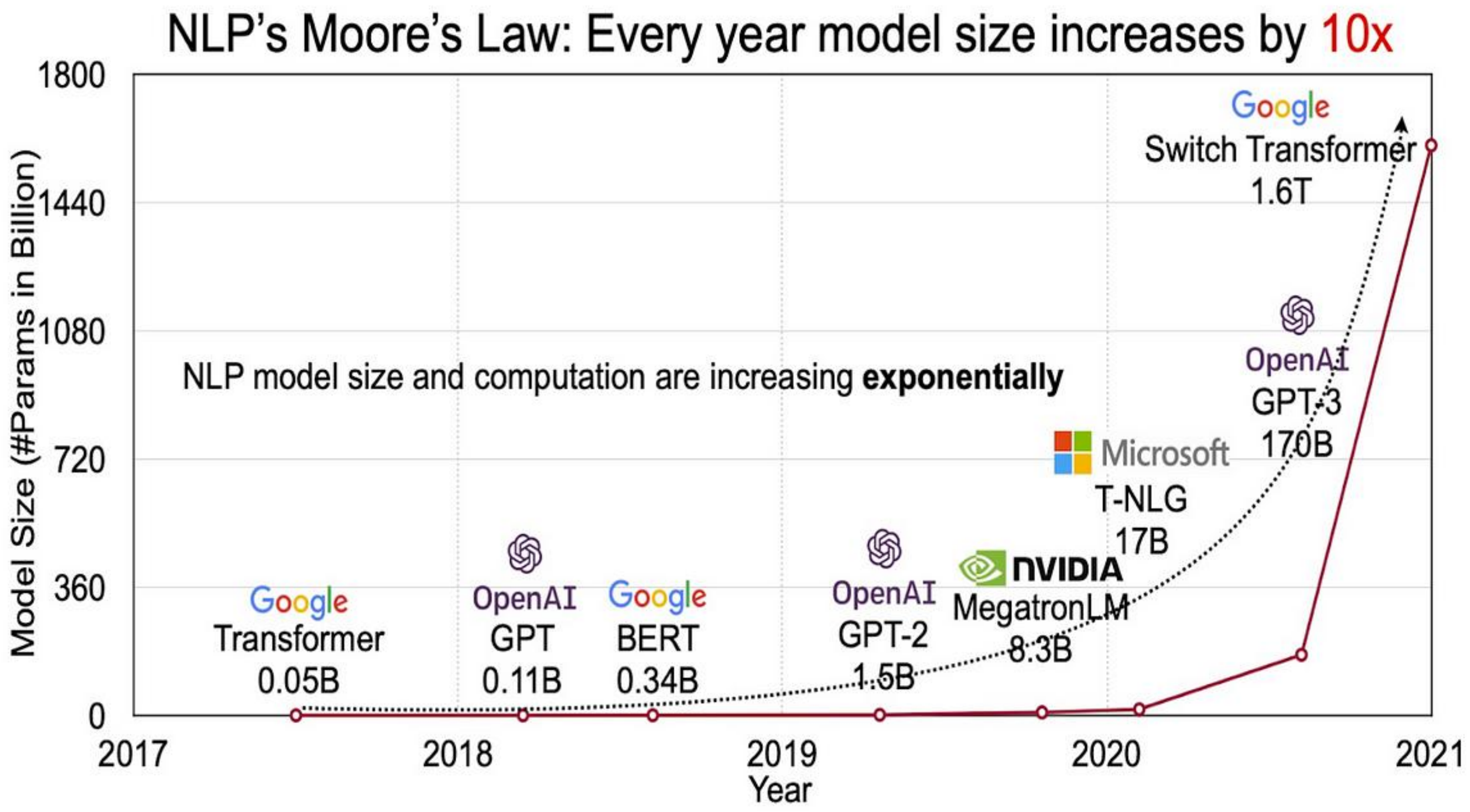
THE HONG KONG POLYTECHNIC UNIVERSITY 香港理工大學

華中科技大學 HUAZHONG UNIVERSITY SCIENCE AND TECHNOLOGY

Outline

Part I: Introduction

Today's Large Models



- The scaling up of model parameters has led to the emergence of many capabilities for large models that are not available to small models.

The Chinchilla Law

- Hoffmann et al. from the DeepMind team proposed chinchilla's law in 2022 [1], which models the relationship between model performance and two main factors: model size (N), and data size (D). The researchers obtained a fitting formula as follows

$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$

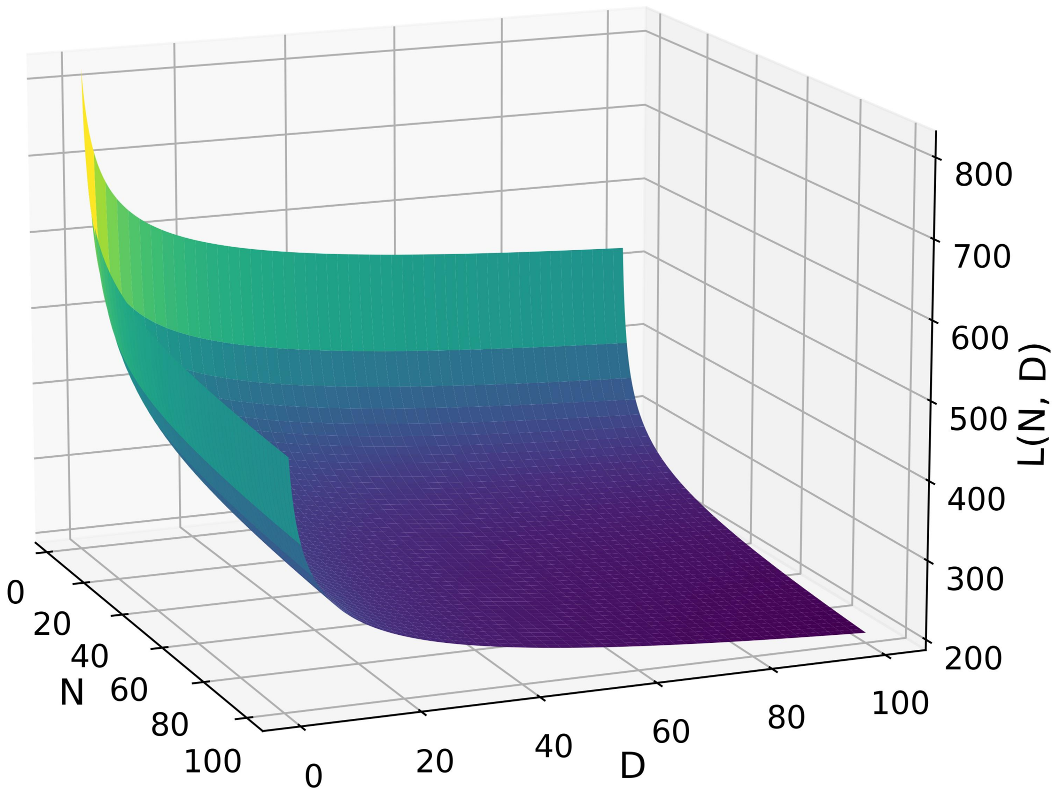
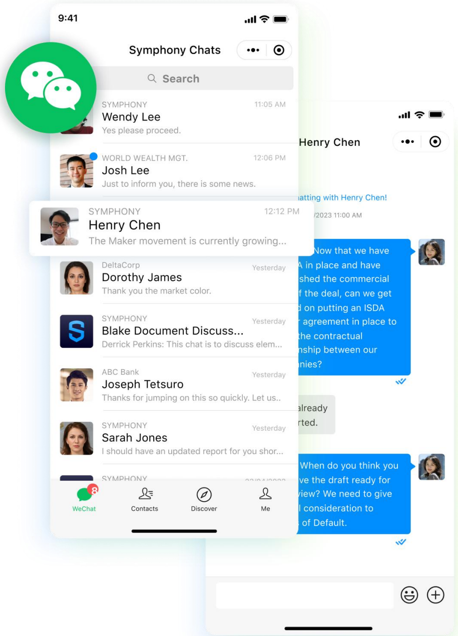


Fig: The landscape of Chinchilla law.

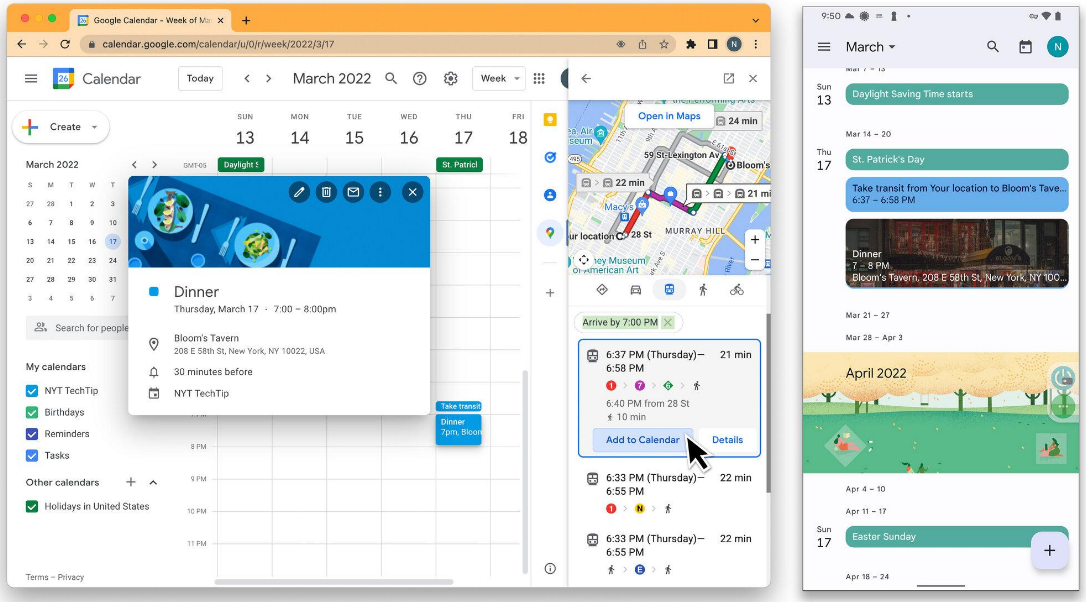
The data size plays an important role in the performance of LLM.

[1] Hoffmann, Jordan, et al. "Training compute-optimal large language models." *arXiv preprint arXiv:2203.15556* (2022).

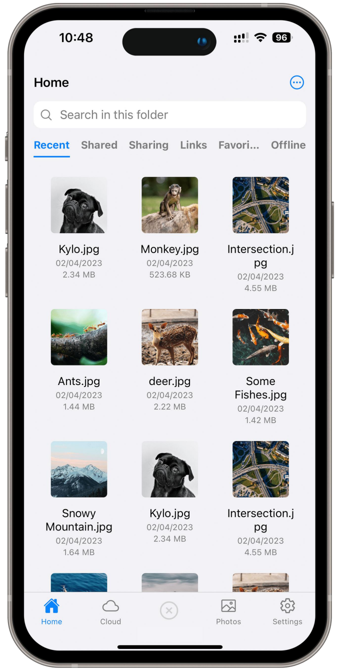
Privacy Concerns



WeChat messages



Calendar schedule



Images & videos

- Valuable data may exist only on the client side, which may contain sensitive information.
- How can we accomplish model training or fine-tuning while ensuring user privacy?

Federated Learning

- Federated learning was first proposed by Google in 2017 to address privacy concerns [1].

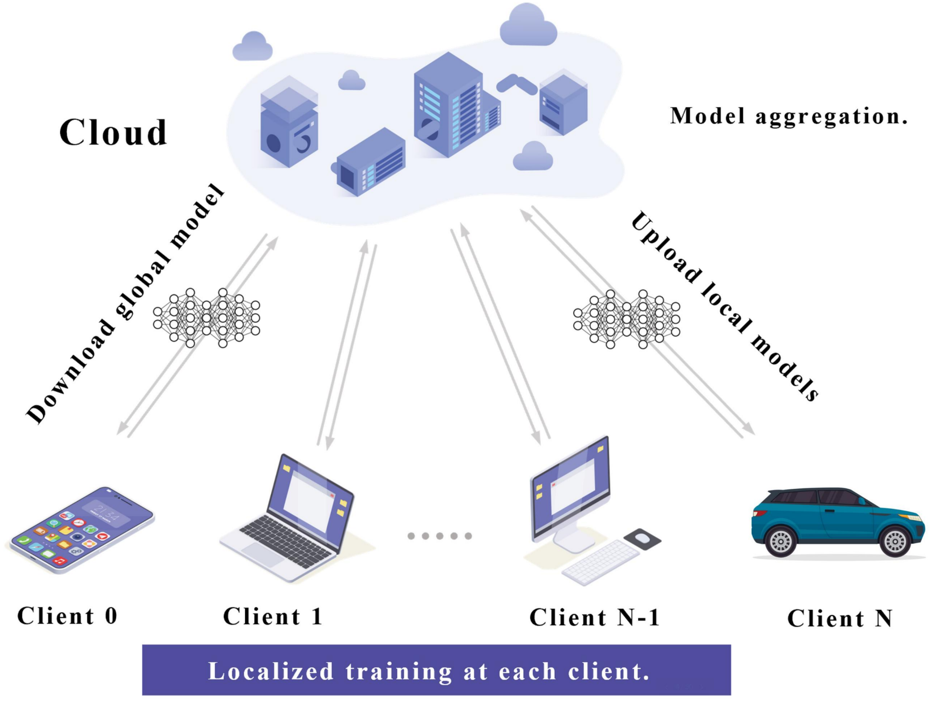


Fig: The training process of federated learning

Training Process

- Cloud distribute initialized global model.
- Each client conducts training using their local datasets.
- Each client uploads trained local model to cloud for aggregation.
- Cloud distribute aggregated model.
- Repeat step 2-4 until converge.

[1] McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." *Artificial intelligence and statistics*. PMLR, 2017.

■ The involved clients have limited resources

- The computational resources of users' personal devices do not match the growing model sizes to accommodate large models for federated training.

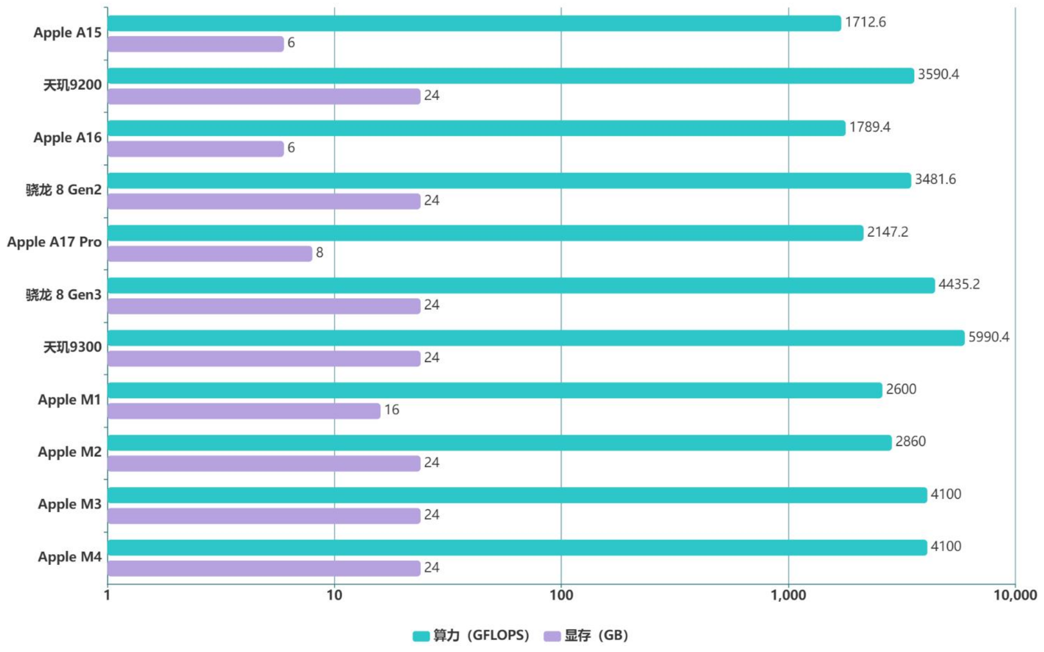


Fig: Current personal devices' resource

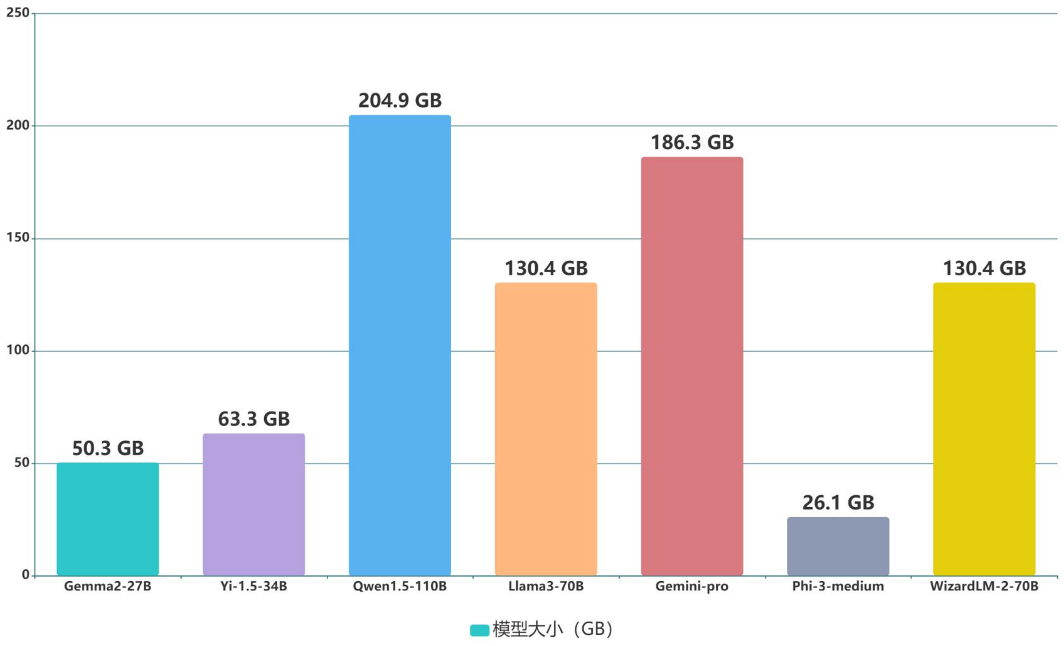


Fig: Current LLMs' size

PEFT methods can reduce resource requirements

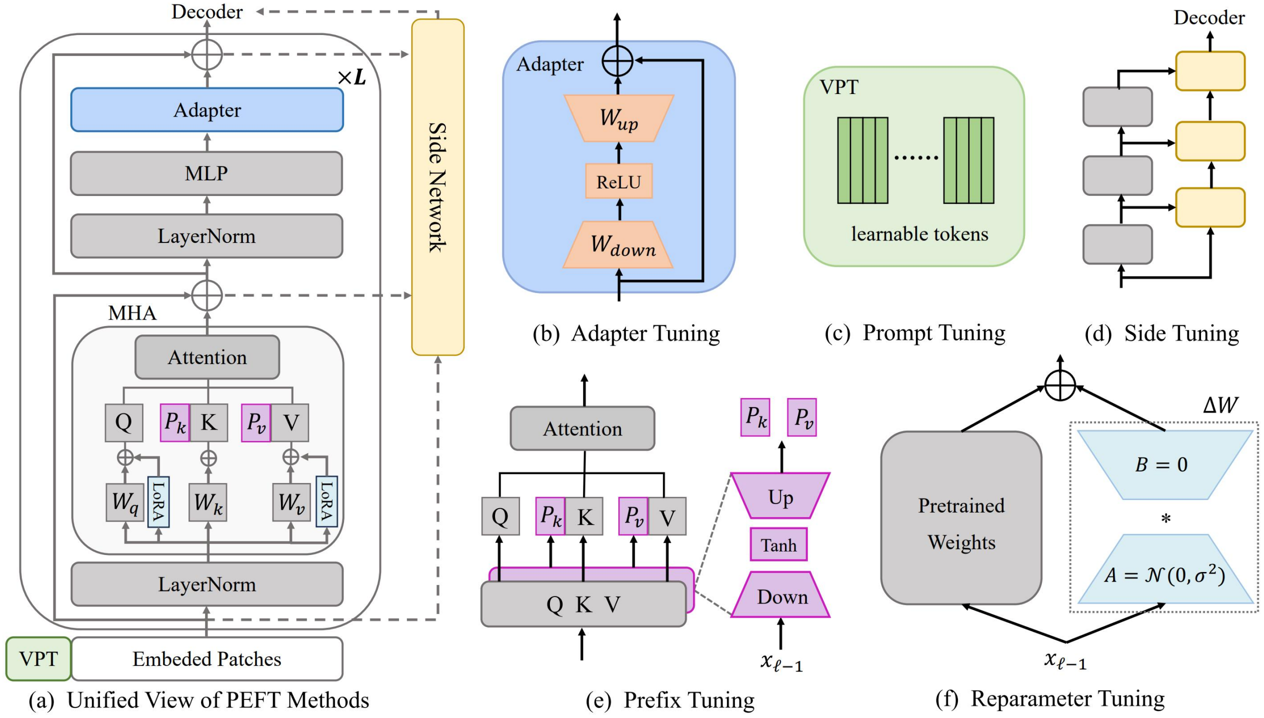


Fig: The detailed architecture of various PEFT methods. [1]

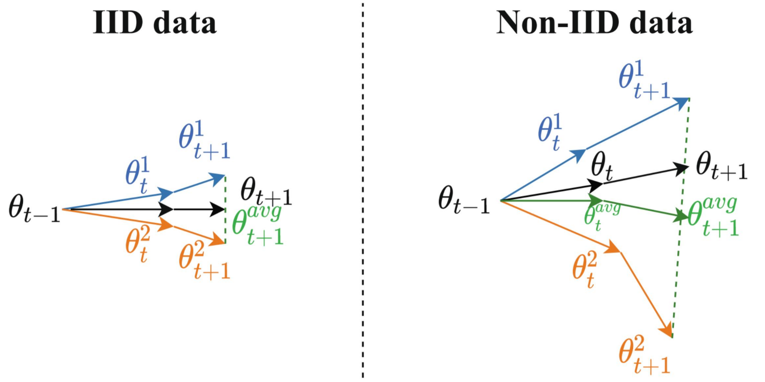
- A wide variety of PEFT methods have emerged, which can effectively reduce the number of trainable parameters and hence the computational overhead.
- While most PEFT methods primarily target parameter efficiency, they still incur **significant memory overhead** during training because the model parameters still need to be fully loaded into the GPU.

- Is PEFT methods can perfectly handle this issue?

[1] Xin, Yi, et al. "Parameter-efficient fine-tuning for pre-trained vision models: A survey." *arXiv preprint arXiv:2402.02242* (2024).

Data Heterogeneity

- Data heterogeneity leads to poor convergence and may cause clients with important data to drop out of training.



	bird	deer	frog	ship
Client 1				
Client 2				

Fig: Non-IID data

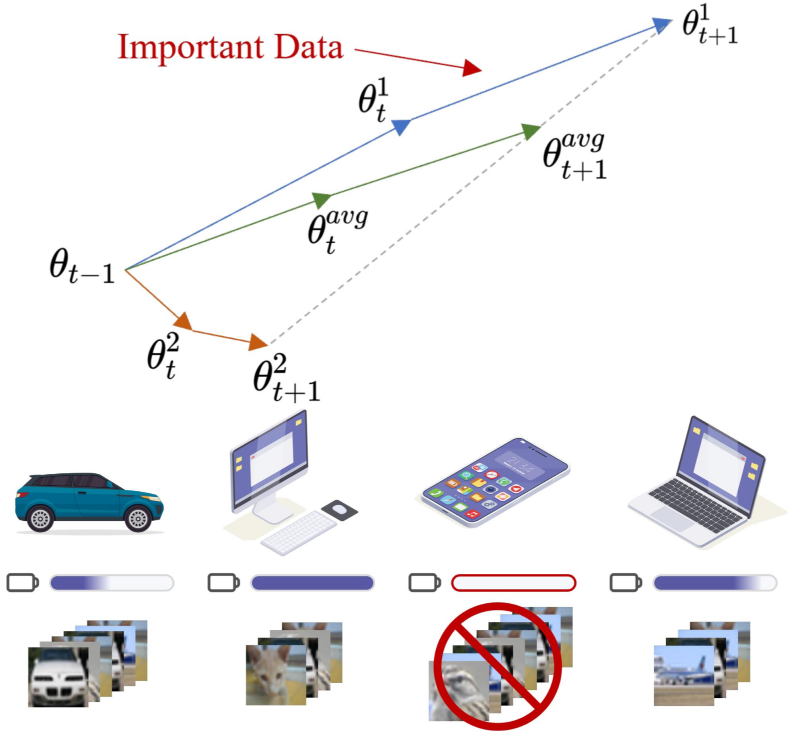
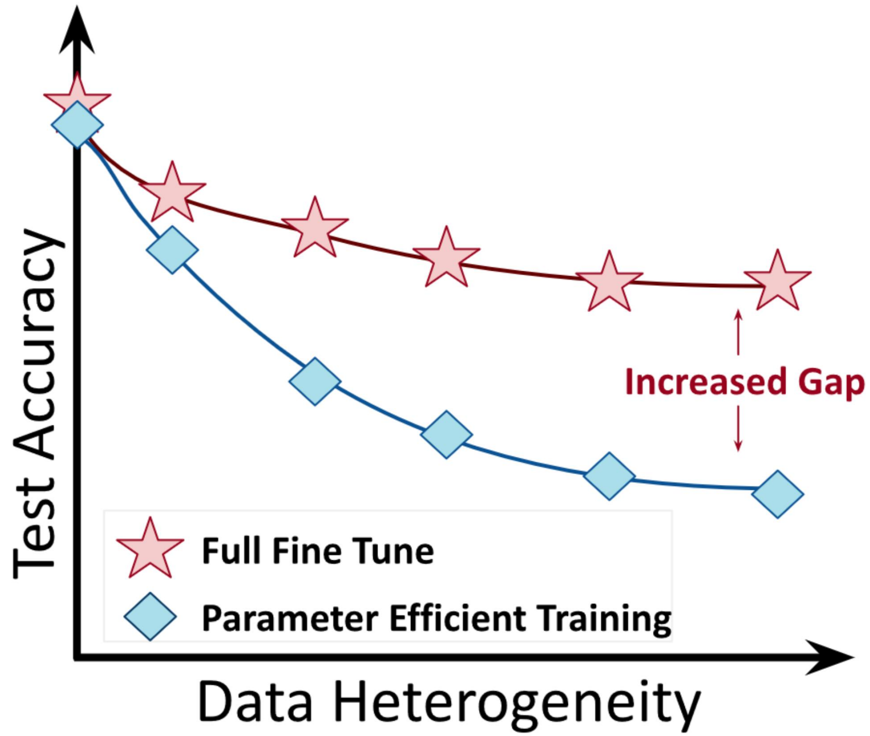


Fig: Important data absence

PEFT methods is vulnerable to data heterogeneity



- The impact of client data distribution on the performance of **full fine-tuning vs. PEFT**. While heterogeneity has an adverse effect on both of them, **parameter efficient methods are more vulnerable and experience more accuracy drop** in more heterogeneous settings.[1]

Can we mitigate client resource cost while maintain model's performance?

[1] Babakniya, Sara, et al. "SLoRA: Federated parameter efficient fine-tuning of language models." *arXiv preprint arXiv:2308.06522* (2023).

Client Heterogeneity

- The clients in the FL system may differ significantly in terms of computational capability and battery level.

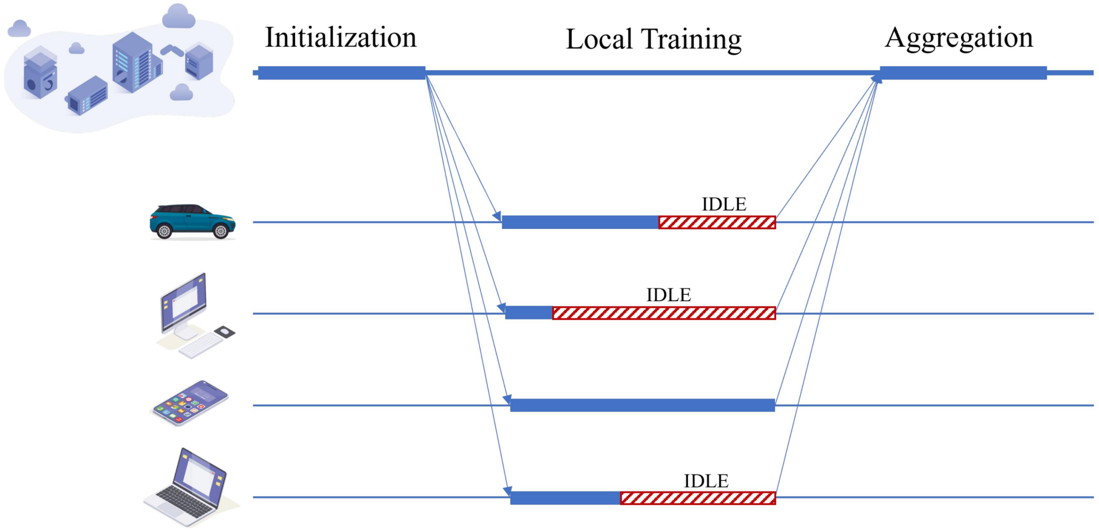


Fig: Straggler effect

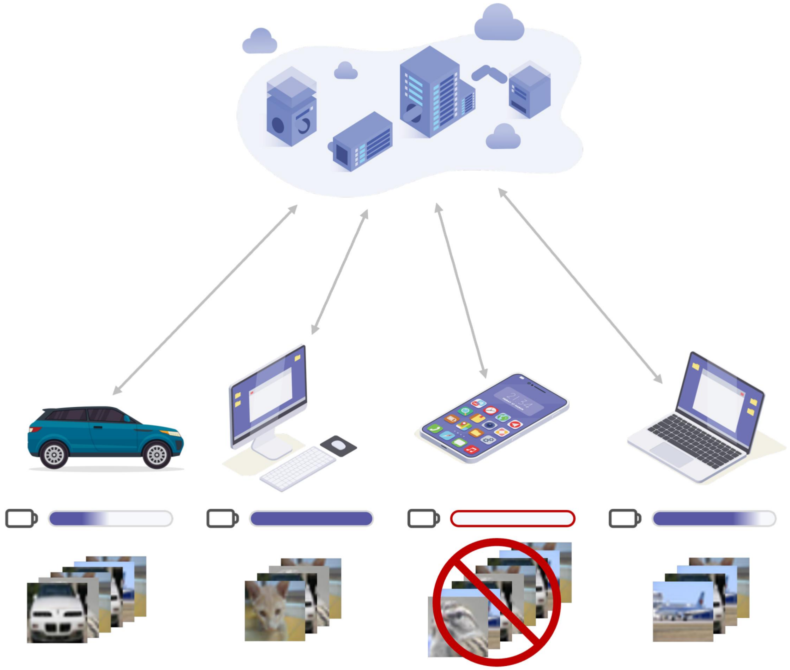
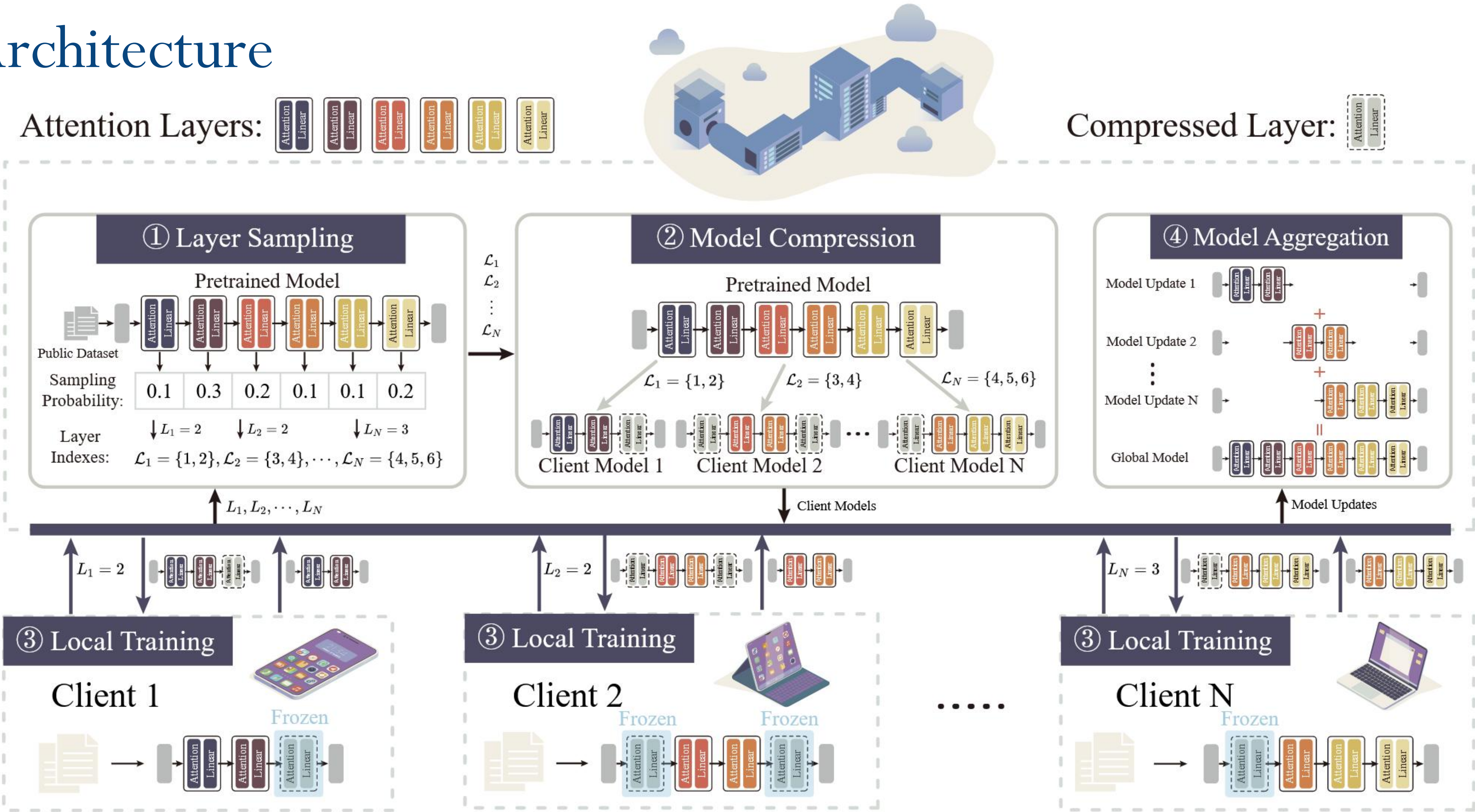


Fig: Client dropout

Part II: Methodology

Architecture



Round Initialization: Layer Sampling

The server selects the layers to be trained in the current round for each client .

Importance-based Sampling Method

- **The importance score of a single parameter**

The importance of a parameter can be quantified by the error induced or remove it [1].

$$\mathcal{I}_m = \left(\mathcal{F}(\mathcal{D}, \Theta) - \mathcal{F}(\mathcal{D}, \Theta \mid_{\theta_m = 0})\right)^2$$

To reduce the complexity, we approximate $\mathcal{I}_m$ in the vicinity of Θ by its first-order Taylor expansion

$$\mathcal{I}_m^{(1)}(\Theta) = \left(g_m \theta_m\right)^2$$

[1] Molchanov, Pavlo, et al. "Importance estimation for neural network pruning." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

Round Initialization: Layer Sampling

The server selects the layers to be trained in the current round for each client.

Importance-based Sampling Method

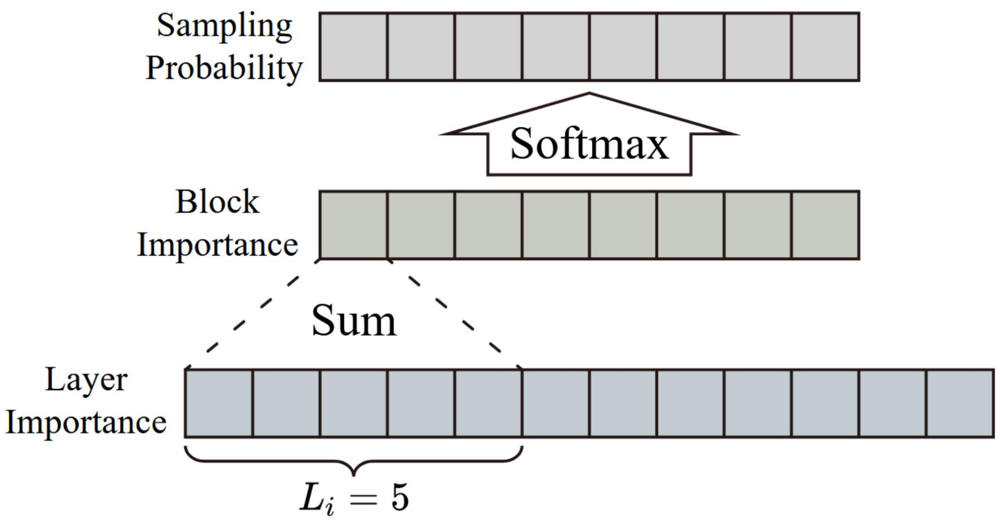
- **The importance score of a layer**

By summing the importance scores of the parameters within the layer, we can obtain the importance of each layer.

$$\mathcal{I}_{\Theta^l} \approx \mathcal{I}^{(1)}_{\Theta^l}(\Theta) = \sum_{m\in\Theta^l} \mathcal{I}^{(1)}_m(\Theta) = \sum_{m\in\Theta^l} (g_m\theta_m)^2$$

The sampling probability for each client can then be expressed as

$$\mathbf{p}_i = \mathrm{Softmax}\left(\{\mathcal{I}_{\bar{\Theta}^k} \mid k = 1, \cdots, L - L_i + 1\}\right)$$

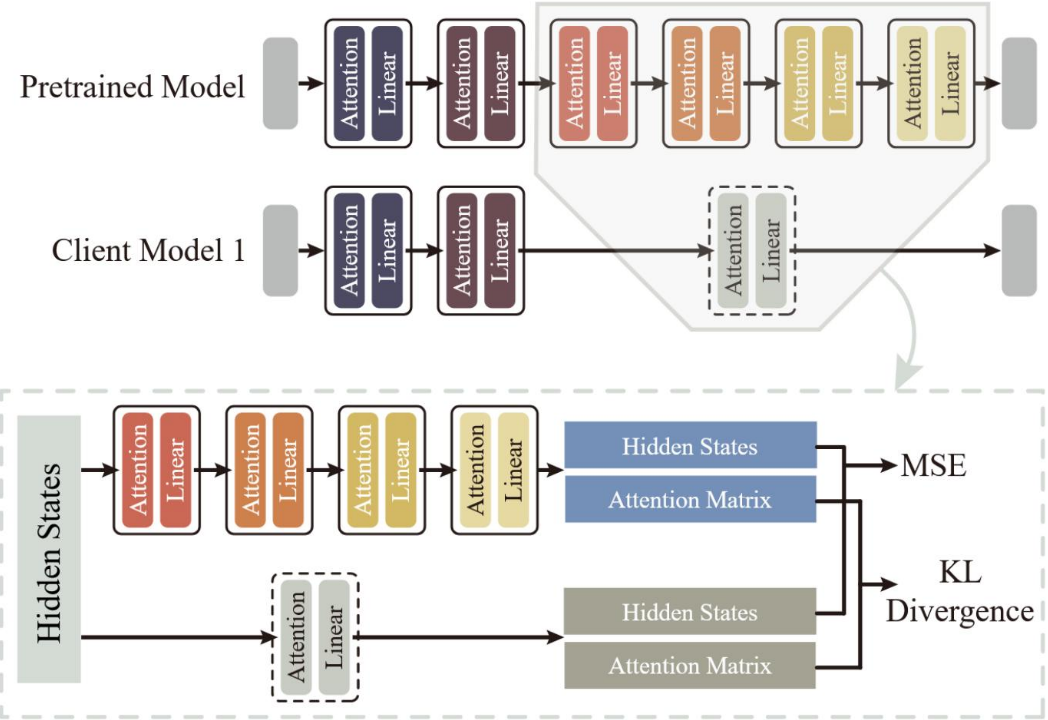


Round Initialization: Model Compression

The server compresses unselected layers to reduce resource consumption.

$$E_{\text{distill}} = (1-\alpha)E_{\text{hidden\_state}} + \alpha E_{\text{attention\_matrix}} = (1-\alpha)\mathbf{MSE}(H_t, H_s) + \alpha\mathbf{KL}(A_t, A_s)$$

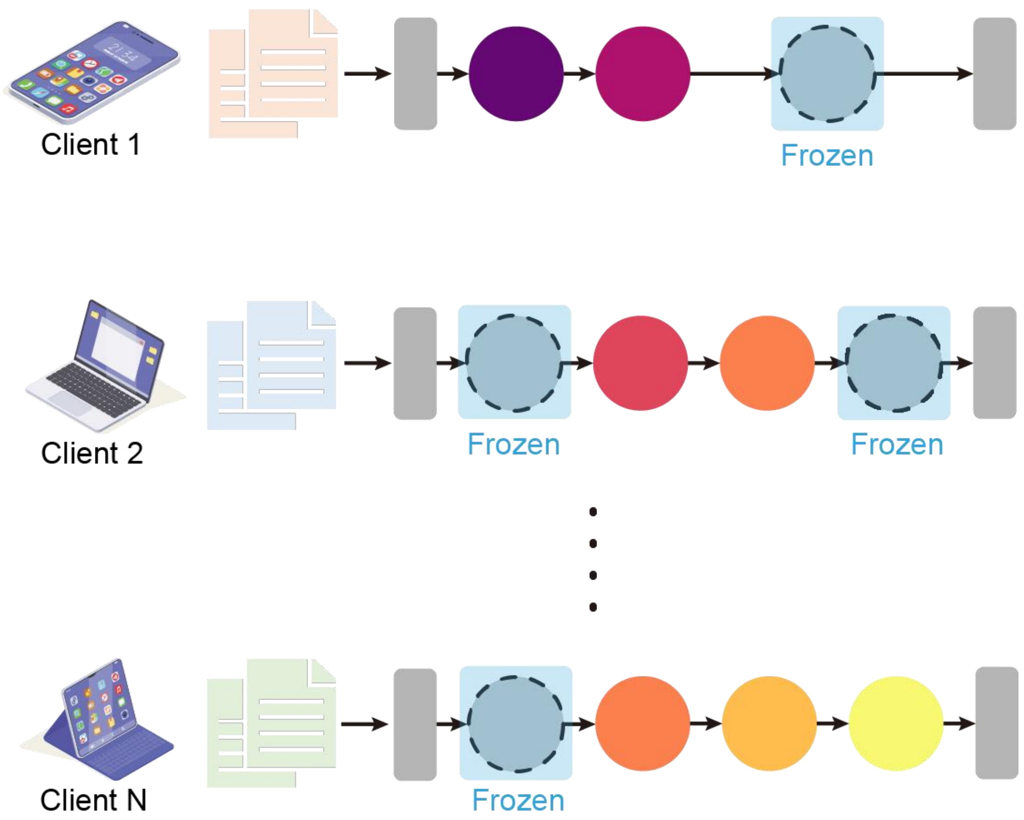
Compression by Knowledge Distillation



1. The server holds a public source proxy dataset for knowledge distillation.
2. The compressed size is determined by a combination of the unselected layer size and the compression rate r: $\left|\check{\Theta}_g^{\mathcal{L}_i^-}\right| = \left\lceil \left|\Theta_g^{\mathcal{L}_i^-}\right| \cdot r \right\rceil$
3. The process of distillation with two losses, the MSE and the KL Divergence.
4. The server sends the compressed model to the corresponding client. $\Theta_i = \{\Theta_g^{\mathcal{L}_i}, \check{\Theta}_g^{\mathcal{L}_i^-}\}$

Local Training

Each client updates some of the layers with the help of the compressed model.

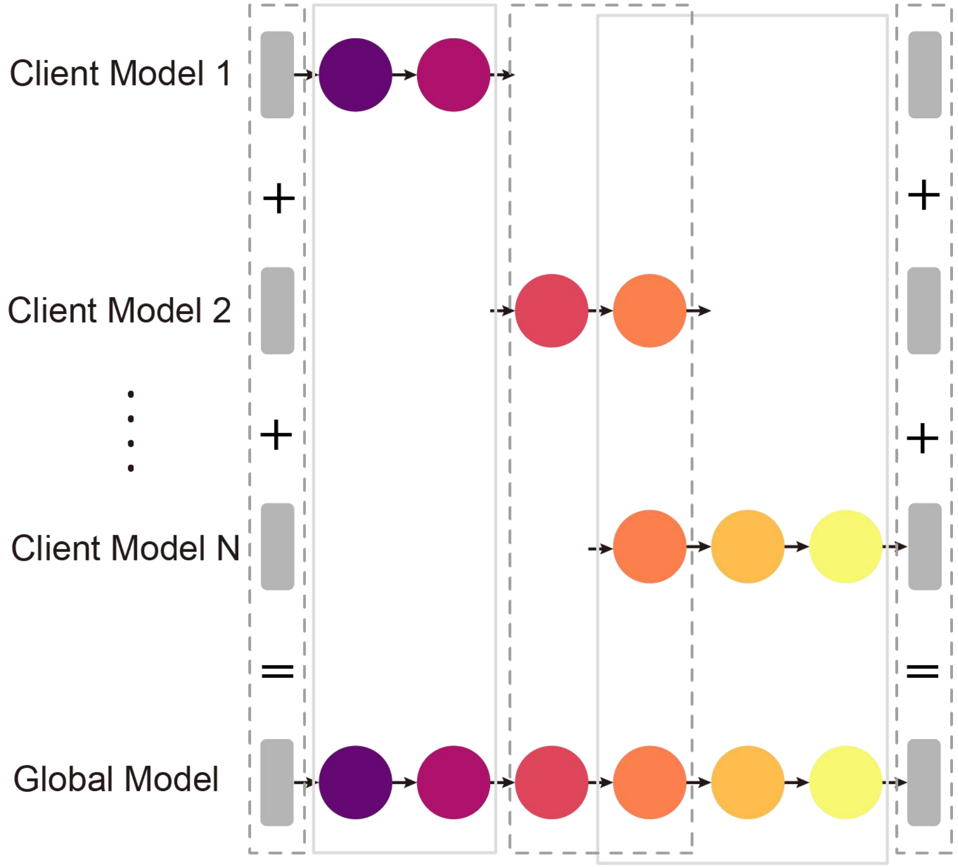


During local training

1. The compressed layer $\breve{\Theta}_{g}^{\mathcal{L}_{\bar{i}}}$ is frozen in order to provide an appropriate training context.
2. Each client updates only the unfrozen layers $\Theta_{g}^{\mathcal{L}_{i}}$ in its local model.
3. Each client uploads the updated layer parameters $\Theta_{i}^{\mathcal{L}_{i}}$ to the server.

Training Process: Aggregation

The server aggregates models in a layer-wise manner to form a parameter-complete model



During model aggregation

1. The server aggregates the received layer updates.
2. Parameters are aggregated by layer and a weighted average is applied to each layer separately.

$$\Theta_g = \{\Theta_g^l \mid l = 1, \cdots, L\} = \left\{ \sum_{i \in \mathcal{S}_l} \frac{D_i}{\sum_{j \in \mathcal{S}_l} D_j} \Theta_i^l \mid l = 1, \cdots, L \right\}.$$

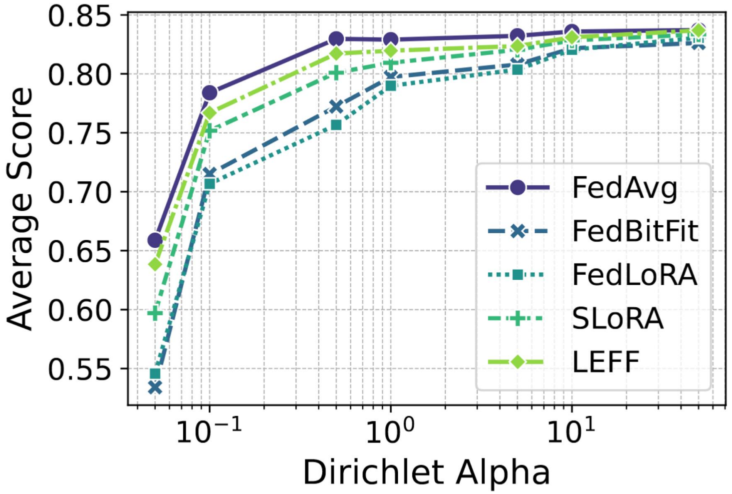
Part III: Evaluation

Performance of LEFF

Setup

- **Dataset:** NLU (GLUE Benchmark: CoLA, MRPC, MNLI, etc); NLG (E2E NLG Challenge).
- **Model:** DeBERTaV3 Base (for NLU tasks); GPT-2 Medium (for NLG tasks).
- **Baselines:** FedAvg (Full-parameter fine-tuning); FedBitFit (Fine-tunes only bias terms); FedLoRA (Parameter-efficient method using low-rank updates); SLoRA (An improved version of FedLoRA).
- **Implementations:** Data is partitioned among clients using a Dirichlet distribution with parameter alpha.

Effect of Data Heterogeneity



- Better performance in data heterogeneous scenarios.

Effect of Client Scale

	Number of Clients				
	8	16	24	32	40
CoLA	61.51	50.21	44.87	35.60	31.26
SST-2	94.30	93.77	93.15	92.30	91.38
MRPC	82.82	77.38	71.44	68.38	64.38
STS-B	86.84	85.90	85.12	80.39	79.35
QQP	88.35	87.32	86.76	86.32	86.00
MNLI	88.20	88.07	87.57	87.36	86.52
QNLI	91.79	91.09	90.45	89.45	89.20
RTE	60.01	59.33	57.16	52.71	50.90
Average	81.73	79.13	77.07	74.06	72.37

- Good adaptability in different client scales.

Performance of LEFF

Computation Resource Overhead

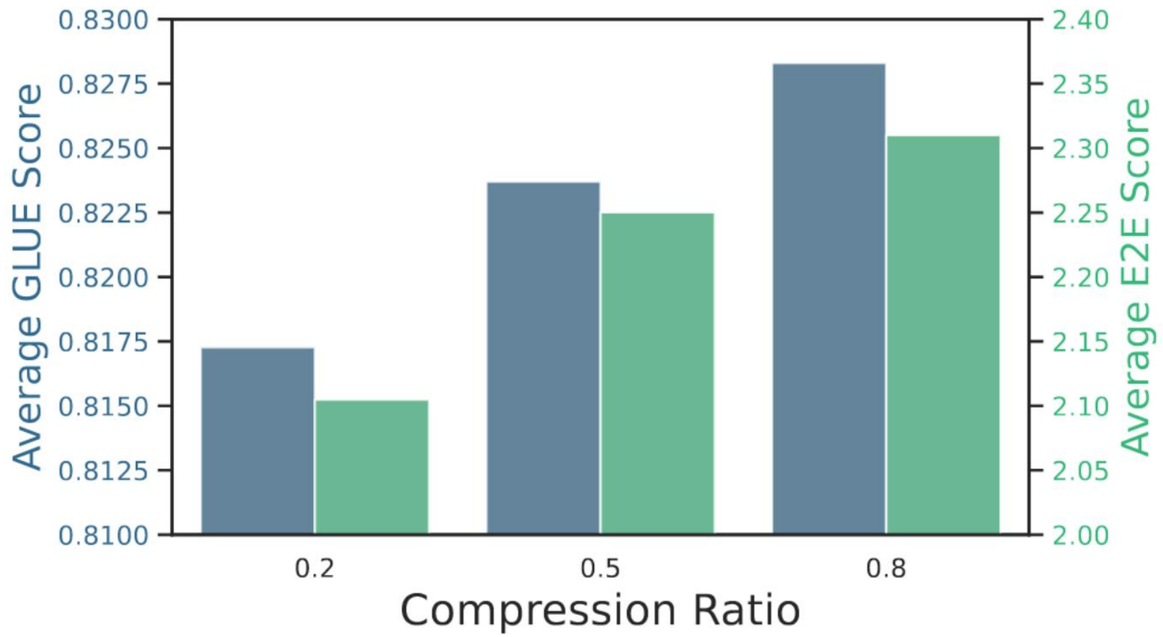
Model	Algorithm	Trainable Params	Peak Memory (GB)
DeBERTaV3-Base	FedAvg	85,648,130	3.841
DeBERTaV3-Base	FedLoRA	1,340,930	2.644
DeBERTaV3-Base	FedBitFit	102,914	2.198
DeBERTaV3-Base	LEFF	7,681,538	2.136
DeBERTaV3-Base	SLoRA	1,340,930	2.644
DeBERTaV3-Large	FedAvg	303,363,074	9.361
DeBERTaV3-Large	FedLoRA	3,557,378	6.660
DeBERTaV3-Large	FedBitFit	272,386	5.488
DeBERTaV3-Large	LEFF	13,649,922	3.005
DeBERTaV3-Large	SLoRA	3,557,378	6.660

Model	Algorithm	Trainable Params	Peak Memory (GB)
GPT2	FedAvg	85,056,000	3.358
GPT2	FedLoRA	811,008	2.476
GPT2	FedBitFit	102,144	2.206
GPT2	LEFF	7,089,408	1.719
GPT2	SLoRA	811,008	2.476
GPT2-Large	FedAvg	708,390,400	15.548
GPT2-Large	FedLoRA	4,055,040	9.739
GPT2-Large	FedBitFit	508,160	8.318
GPT2-Large	LEFF	19,680,000	3.240
GPT2-Large	SLoRA	4,055,040	9.739
Llama-3.1-8B	FedAvg	OOM	OOM
Llama-3.1-8B	FedLoRA	20,971,520	46.868
Llama-3.1-8B	FedBitFit	No Bias	No Bias
Llama-3.1-8B	LEFF	743,452,672	29.881
Llama-3.1-8B	SLoRA	20,971,520	46.868

- Lower GPU memory overhead.

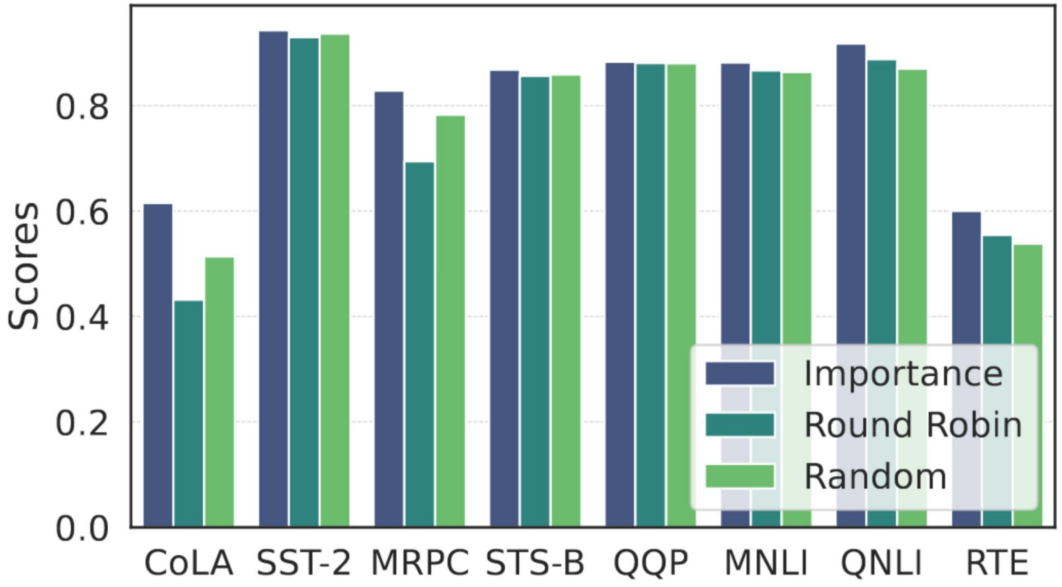
Performance of LEFF

Effect of Different Compression Ratio



- The larger the compression ratio, the better the training results, but it leads to higher computational resource overhead.

Ablation Study of Sampling Method



- The importance sampling approach has better performance compared to the baseline approaches.

Part IV: Conclusion

Summary

1. We introduced **LEFF**, a novel framework for efficient federated fine-tuning of LLMs.
2. LEFF effectively balances computational efficiency with model performance by using selective layer-wise fine-tuning.
3. It is robust to both data and system heterogeneity, making it practical for real-world edge environments.
4. Theoretical analysis guarantees convergence, and empirical results show performance comparable to full fine-tuning.

Limitations and Future Work

1. **Efficiency-Fidelity Trade-off:** The approximation error from compression can create a performance ceiling.
2. **Server-side Load:** LEFF shifts some complexity (importance calculation, compression) to the server.
3. **Future Directions:** Explore more advanced adaptive compression strategies and optimize server-side operations.

Thank You!