

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εργαστήριο Μικροϋπολογιστών και Ψηφιακών Συστημάτων

Routing-Aware Federated Fine-Tuning of Mixture-of-Experts LLMs

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ξένου Εμμανουήλ

Επιβλέπων: Δημήτριος Σούντρης
Καθηγητής Ε.Μ.Π.

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ (Microlab)
Αθήνα, Μάρτιος 2026



Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Μικροϋπολογιστών και Ψηφιακών Συστημάτων

Routing-Aware Federated Fine-Tuning of Mixture-of-Experts LLMs

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ξένου Εμμανουήλ

Επιβλέπων: Δημήτριος Σούντρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11η Μαρτίου 2026.

.....	.....	.....
Δημήτριος Σούντρης	Σωτήριος Ξύδης	Γεώργιος Ζερβάκης
Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2026

.....

Εμμανουηλ Ξενος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός
και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © – All rights reserved Ξένος Εμμανουήλ, 2026.
Με επιφύλαξη παντός δικαιώματος.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Στον πατέρα μου και στη μνήμη της μητέρας μου

Περίληψη

Η ραγδαία εξάπλωση των Μεγάλων Γλωσσικών Μοντέλων (LLMs) έχει οδηγήσει στην υιοθέτηση αραιών αρχιτεκτονικών, όπως η Μείξη Ειδικών (Mixture-of-Experts - MoE), οι οποίες αποσυνδέουν τον αριθμό των παραμέτρων του μοντέλου από το κόστος εκτέλεσης. Ωστόσο, η εκπαίδευση (training) και μικρορύθμιση (finetuning) αυτών των μοντέλων σε περιβάλλον Ομόσπονδης Μάθησης (Federated Learning) παρουσιάζει σημαντικές προκλήσεις λόγω του τεράστιου όγκου των συνολικών παραμέτρων και των περιορισμών σε εύρος ζώνης, ενέργεια και διαθέσιμους πόρους των συμμετεχόντων (clients). Επιπλέον, η ανομοιογένεια που χαρακτηρίζει τους συμμετέχοντες σε περιβάλλον Ομόσπονδης Εκπαίδευσης (π.χ. διαφορετική μνήμη VRAM και υπολογιστική ισχύς) καθιστά την τυπική εκπαίδευση MoE μη πρακτική.

Στην παρούσα διπλωματική εργασία, προτείνουμε το **FedMoECap**, ένα νέο πλαίσιο σχεδιασμένο για την αποδοτική μικρορύθμιση (finetuning) μοντέλων MoE. Εισάγουμε ένα κριτήριο **Παγώματος βάσει Ενεργοποίησης** (Activation Freeze Criterion) που ιεραρχεί τους σημαντικότερους ειδικούς (experts) σε όλα τα επίπεδα (layers) του δικτύου, βάσει ενός ποσοστού παγώματος (freeze rate) που κάθε client ορίζει ανάλογα με τους διαθέσιμους πόρους του. Στους επιλεγμένους ειδικούς εφαρμόζουμε **Επιλεκτική LoRA** (Selective LoRA), ώστε να περιοριστεί περαιτέρω ο αριθμός των εκπαιδεύσιμων παραμέτρων. Επιπλέον, αναπτύσσουμε τρεις προσαρμοστικές στρατηγικές (Static, Pruning και Rolling) για τη διαχείριση της σύγκλισης των experts.

Η πειραματική μας αξιολόγηση σε συσκευές άκρου **NVIDIA Jetson AGX Orin** χρησιμοποιώντας το μοντέλο **OLMoE-1B-7B** και **NVIDIA A100** GPU σε σύστημα **NVIDIA DGX** δείχνει ότι το FedMoECap μειώνει το κόστος επικοινωνίας έως και **95%** και τη συνολική κατανάλωση ενέργειας κατά **50.8%** σε σύγκριση με την τυπική μέθοδο που εκπαιδεύει όλους τους ειδικούς, διατηρώντας ταυτόχρονα ανταγωνιστική ακρίβεια σε benchmarks όπως τα PIQA, BoolQ, CommonsenseQA και Winogrande. Τα αποτελέσματα αυτά επιβεβαιώνουν ότι το FedMoECap γεφυρώνει επιτυχώς το χάσμα μεταξύ των τεράστιων μοντέλων MoE και των περιορισμένων πόρων του Federated Learning.

Λέξεις Κλειδιά — Μεγάλα Γλωσσικά Μοντέλα, Μείξη Ειδικών, Ομόσπονδη Μάθηση, Παραμετρικά Αποδοτική Μικρορύθμιση, Υπολογιστική Άκρου.

Abstract

The rapid proliferation of Large Language Models (LLMs) has led to the adoption of sparse architectures, such as Mixture-of-Experts (MoE), which decouple model capacity from inference cost. However, training and fine-tuning these models in a Federated Learning (FL) setting presents significant challenges due to the massive parameter footprint and the bandwidth, energy, and resource constraints of participating clients. Furthermore, the system heterogeneity inherent in FL participants (e.g., varying VRAM and compute capabilities) makes standard MoE training impractical.

In this thesis, we propose **FedMoECap**, a novel framework designed for efficient fine-tuning of MoE models. We introduce an **Activation Freeze Criterion** that prioritizes the most important experts across all layers of the network based on a per-client freeze rate determined by each device's hardware budget. We then apply **Selective LoRA** to the chosen experts, further reducing the trainable parameter count. Finally, we develop three adaptive strategies (Static, Pruning, and Rolling) to manage expert convergence throughout training.

Our experimental evaluation on **NVIDIA Jetson AGX Orin** edge devices using the **OLMoE-1B-7B** model and an **NVIDIA A100 (40 GB)** GPU hosted on an **NVIDIA DGX** system demonstrates that FedMoECap reduces communication overhead by up to **95%** and total energy consumption by **50.8%** compared to the standard method that trains all experts, while maintaining competitive accuracy on benchmarks such as PIQA, BoolQ, CommonsenseQA, and Winogrande. These results confirm that FedMoECap successfully bridges the gap between massive MoE models and the constraints of Federated Learning.

Keywords — Large Language Models, Mixture-of-Experts, Federated Learning, Parameter-Efficient Fine-Tuning, Edge Computing.

Ευχαριστίες

Θα ήθελα καταρχάς να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. **Δημήτριο Σούντρη** για την ανάθεση αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να εργαστώ πάνω σε σύγχρονα θέματα όπως τα Mixture-of-Experts LLMs και το Federated Learning. Τον ευχαριστώ επίσης θερμά για το ειλικρινές ενδιαφέρον που έδειξε όχι μόνο πριν ή κατά την διάρκεια της διπλωματικής μου, αλλά και για το ενδιαφέρον που έδειξε για το μέλλον μου μετά την ολοκλήρωση της διπλωματικής μου.

Ευχαριστώ θερμά τους διδάκτορες **Δημοσθένη Μασούρο** και **Γεώργιο Αρμενιάκο** για την καθοδήγησή τους, την τεχνική υποστήριξη, το ενδιαφέρον που έδειξαν και τις εποικοδομητικές συζητήσεις που είχα μαζί τους κατά την διπλωματική μου.

Θα ήθελα να ευχαριστήσω το NVIDIA's Academic Grant Program, το AWS GRNet, τον επίκουρο καθηγητή Ζερβάκη Γεώργιο και το Πανεπιστήμιο Πατρών για την παροχή πόρων για την τέλεση των πειραμάτων της διπλωματικής μου.

Ευχαριστώ θερμά την οικογένειά μου για όσα μου έχουν προσφέρει όλα αυτά τα χρόνια και για την στήριξή τους ώστε να έχω την δυνατότητα να φτάσω ως στην ολοκλήρωση των σπουδών μου.

Τέλος, ευχαριστώ τους φίλους μου για την ηθική συμπαράσταση, την υπομονή τους και τις αμέτρητες αξέχαστες στιγμές που περάσαμε μαζί.

Ξένος Εμμανουήλ
Αθήνα, Μάρτιος 2026

Contents

Περίληψη vii

Abstract ix

Ευχαριστίες xi

Contents xiii

List of Figures xv

List of Tables xvii

Εκτεταμένη Ελληνική Περίληψη 1

1 Introduction 11
- 1.1 The Rise of Federated Learning . . . 11
- 1.2 System Heterogeneity: The Defining Challenge of Federated Learning . . . 11
- 1.3 The Promise and Challenges of Mixture-of-Experts in Federated Fine-Tuning . . . 12
- 1.4 The Gap and the Proposed Solution . . . 13
- 1.5 Contributions . . . 13
- 1.6 Thesis Structure . . . 14

2 Theoretical Background 15
- 2.1 The Transformer Architecture . . . 15
  - 2.1.1 Self-Attention Mechanism . . . 15
  - 2.1.2 Multi-Head Attention . . . 15
  - 2.1.3 The Transformer Block . . . 16
- 2.2 Mixture-of-Experts (MoE) . . . 16
  - 2.2.1 Top-K Routing Mechanism . . . 17
- 2.3 Parameter-Efficient Fine-Tuning (PEFT) . . . 17
  - 2.3.1 Low-Rank Adaptation (LoRA) . . . 17
- 2.4 Federated Learning (FL) . . . 18
  - 2.4.1 The Federated Averaging (FedAvg) Algorithm . . . 18
  - 2.4.2 Challenges in Federated Learning . . . 19
- 2.5 Model Quantization . . . 19

3 Related Work 21
- 3.1 Federated Fine-Tuning of Large Language Models . . . 21
- 3.2 Parameter-Efficient Fine-Tuning in Federated Settings . . . 22
- 3.3 Expert-Aware Fine-Tuning of Mixture-of-Experts Models . . . 22
- 3.4 Resource-Aware and Heterogeneous Federated Learning . . . 23
- 3.5 Mixture of Experts and Quantization . . . 24
- 3.6 Mixture of Experts and Federated Learning . . . 24

3.7 Summary and Positioning . . . 25

4 Motivation **27**
4.1 Observation 1: Expert Routing Is Sparse, Skewed, and Task-Dependent . . . 27
4.2 Observation 2: Expert Importance Is a Global Property, Not a Per-Layer One . . . 28
4.3 Observation 3: The Pre-Trained Router Is a Reliable and Stable Signal . . . 29
4.4 Observation 4: Hardware Heterogeneity Demands a First-Class Parameter . . . 29
4.5 Observation 5: Expert Utility Diminishes During Training . . . 30
4.6 Summary of Design Principles . . . 31

5 Methodology **33**
5.1 Problem Formulation and Design Objectives . . . 33
5.2 The Activation Freeze Criterion . . . 34
5.2.1 Expert Activation Profiling . . . 34
5.2.2 Global Expert Ranking and Selection . . . 34
5.2.3 Frozen Router . . . 35
5.3 Selective LoRA for MoE Fine-Tuning . . . 35
5.4 Per-Client Freeze Rate and Heterogeneity Handling . . . 36
5.5 Adaptive Convergence Management . . . 37
5.5.1 Convergence Criterion . . . 37
5.5.2 FedMoECap-S: Static Selection . . . 38
5.5.3 FedMoECap-P: Convergence-Based Pruning . . . 38
5.5.4 FedMoECap-R: Rolling Expert Swapping . . . 38
5.6 Hybrid Aggregation for the Rolling Variant . . . 39
5.7 The Full FedMoECap Training Protocol . . . 39
5.8 Summary . . . 40

6 Experimental Evaluation **41**
6.1 Experimental Setup . . . 41
6.1.1 Model . . . 41
6.1.2 Hardware Infrastructure . . . 41
6.1.3 Datasets and Evaluation . . . 42
6.1.4 Federated Setup and Hyperparameters . . . 42
6.1.5 Evaluation Metrics . . . 42
6.2 Model Performance . . . 43
6.2.1 Edge Deployment: NVIDIA Jetson AGX Orin . . . 44
6.2.2 Server Deployment: NVIDIA A100 . . . 45
6.3 Design and Hyperparameter Analysis . . . 46
6.3.1 Impact of the Freeze Rate . . . 46
6.3.2 Impact of the Convergence Threshold on FedMoECap-P . . . 47
6.3.3 Impact of the Convergence Threshold on FedMoECap-R . . . 47
6.3.4 Impact of the Patience on FedMoECap-P . . . 48
6.3.5 Impact of the Patience on FedMoECap-R . . . 50
6.3.6 ESFT vs Activation Freeze Criterion . . . 51
6.3.7 FedMoECap-P with Freeze Rate 0 vs FedAvg-LoRA vs FedMoECap-S . . . 51
6.4 Pareto Analysis . . . 52

7 Conclusion **55**
7.1 Summary of Contributions . . . 55
7.2 Key Findings . . . 56
7.3 Limitations . . . 57
7.4 Future Work . . . 57
7.5 Closing Remarks . . . 58

Bibliography **59**

List of Figures

2.1.1 Transformer Architecture of Encoder-Decoder models [15]. Decoder-only models such as GPT, Gemini, and Claude use only the right (decoder) stack. 16

2.2.1 Mixture-of-Experts layer compared to a standard dense Feed-Forward layer. In the MoE layer, a router selects only a sparse subset of experts per token, keeping inference cost low while scaling total model capacity [11]. 17

2.4.1 Basic Federated Learning workflow. Clients train locally on private data and upload only model updates to a central server, which aggregates them into a new global model before broadcasting it back. Image source: [16]. 18

3.3.1 Comparison of fine-tuning paradigms. Full-Parameter Fine-Tuning (FFT) trains all weights, Low-Rank Adaptation (LoRA) injects trainable low-rank matrices into frozen layers, and Expert-Specialized Fine-Tuning (ESFT) selectively trains only the most relevant experts per layer. FedMoECap extends this line of work to the federated setting. 23

4.1.1 Difference in expert routing probability between a Dolly-15K split and a Rosetta Code split on OLMoE-1B-7B. Red cells indicate experts more active on Rosetta Code; blue cells indicate experts more active on Dolly15K. The large-magnitude divergence across layers confirms that expert routing is strongly task-dependent. 28

4.3.1 Difference in expert routing probability between two non-IID splits of the Dolly-15K dataset assigned to different federated clients, on OLMoE-1B-7B. Even within the same dataset, the skewed local distributions produce divergent routing patterns across layers, motivating the decision to freeze the router throughout training. 30

4.5.1 Relative parameter update magnitude per expert across federated rounds. Each colored line tracks one expert; The black dashed line (right axis) shows the number of still-trainable experts per round. Experts converge at different rates, motivating a dynamic pruning strategy that retires each expert individually as its updates fall below a threshold. 31

5.2.1 Stability and time cost of the Activation Freeze Criterion profiling pass as a function of subset size, averaged over Meta-MathQA, Rosetta Code, and Alpaca datasets. Solid lines (left axis) show the average intersection ratio of expert selections produced by three independent random subsets of each size: a ratio of 1.0 means all three draws selected identical experts. Dashed lines (right axis) show the average wall-clock time of the profiling pass. The Activation Freeze Criterion (green) achieves intersection ratios above 0.85 at just 50 samples and above 0.93 at 100 samples, while its time cost remains comparable to ESFT (red) at every subset size. 35

5.2.2 Routing probability heatmaps comparing the expert selection patterns of the Activation Freeze Criterion and ESFT, derived from mathematical routing probabilities. Selection density and distribution are discussed in Section 5.2. 36

5.2.3 Heatmaps computed from actual token routing counts recorded during the profiling pass, confirming the practical reproduction of the mathematical selection patterns. . . 37

5.7.1 Overview of the FedMoECap training protocol. The **Client Side Workflow** (top-left) shows the per-round steps executed on each participating device, while the **Server Workflow** (bottom) shows the aggregation and convergence management steps. Step 1 (expert profiling and selection) is performed only once at the first communication round. Coloured arrows distinguish the three variants: black arrows denote steps shared by all variants, solid blue arrows denote the FedMoECap-S path (steps 7 → 8a → 2), dashed green arrows denote the FedMoECap-P path (steps 7 → 8b → 9a → 2), and dotted red arrows denote the FedMoECap-R path (steps 7 → 8b → 9b → 2). The heterogeneous client panel (top-right) illustrates that different clients may operate with different freeze rates f_r, reflected in the varying fraction of coloured (trainable) experts across client models. 40

6.2.1 Accuracy of each method across all four benchmarks. FedMoECap-S closely approaches the FedAvg-LoRA (All Experts) ceiling, while FedMoECap-P and FedMoECap-R occupy progressively lower positions. The missing bar for FedMoECap-R on PIQA indicates a result not yet available at the time of writing. 44

6.3.1 Left: trainable experts per round for different patience values on FedMoECap-P. Right: allocated VRAM per round. The expert curve is shifted later with higher patience, confirming that the apparent VRAM reduction is an averaging artefact. 49

6.4.1 Pareto front: accuracy vs. total communication bandwidth on the Jetson AGX Orin (CommonsenseQA, left) and the NVIDIA A100 (BoolQ, right). The dashed line connects Pareto-optimal configurations. Points below the frontier are dominated. 53

6.4.2 Pareto front: accuracy vs. total energy consumption on the Jetson AGX Orin (CommonsenseQA, left) and accuracy vs. training time per client on the NVIDIA A100 (BoolQ, right). The dashed line connects Pareto-optimal configurations. 53

List of Tables

0.0.1 Σύγκριση ακρίβειας στα τέσσερα benchmarks αξιολόγησης. Το σύμβολο "−" υποδηλώνει αποτέλεσμα που δεν ήταν διαθέσιμο κατά τη συγγραφή. 5
0.0.2 Αποδοτικότητα συστήματος στο NVIDIA Jetson AGX Orin (εργασία CommonsenseQA). Τα ποσοστά σε παρενθέσεις εκφράζουν μεταβολή σε σχέση με τη βάση FedAvg-LoRA (All Experts). 5
0.0.3 Αποδοτικότητα συστήματος στο NVIDIA A100 (εργασία BoolQ). Τα ποσοστά σε παρενθέσεις εκφράζουν μεταβολή σε σχέση με τη βάση FedAvg-LoRA (All Experts). 6
0.0.4 Επίδραση του ποσοστού παγώματος στο FedMoECap-S (CommonsenseQA, NVIDIA Jetson AGX Orin). Τα ποσοστά εκφράζουν μεταβολή σε σχέση με την προεπιλογή 0.90. . . . 6
0.0.5 Επίδραση της υπομονής P στο FedMoECap-P (CommonsenseQA, NVIDIA Jetson AGX Orin). Τα ποσοστά εκφράζουν μεταβολή σε σχέση με την προεπιλογή $P = 2$. 7

3.7.1 Comparison of related approaches along key design dimensions. ✓ indicates the property is fully satisfied; (✓) indicates profiling is performed but incurs non-negligible overhead (FedMoE requires a costly preliminary fine-tuning stage; HFedMoE re-profiles at every mini-batch); × indicates the property is not addressed. 25

6.2.1 Accuracy comparison on PIQA, CommonsenseQA (CommQA), BoolQ, and Winogrande. The symbol '−' indicates that the experiment for that configuration was not yet completed at the time of writing. 43
6.2.2 System efficiency on the NVIDIA Jetson AGX Orin (CommonsenseQA task). Percentages in parentheses indicate change relative to FedAvg-LoRA (All Experts). 44
6.2.3 System efficiency on the NVIDIA A100 (BoolQ task). Percentages in parentheses indicate change relative to FedAvg-LoRA (All Experts). Training time is averaged across clients. 45
6.3.1 Sensitivity to the expert freeze rate on FedMoECap-S (CommonsenseQA, NVIDIA Jetson AGX Orin). Percentages are relative to the default freeze rate of 0.90. 46
6.3.2 Sensitivity to the expert freeze rate on FedMoECap-S (BoolQ, NVIDIA A100). Percentages are relative to the default freeze rate of 0.90. 46
6.3.3 Sensitivity to the convergence threshold τ on FedMoECap-P (CommonsenseQA, NVIDIA Jetson AGX Orin). Percentages are relative to the default threshold of 0.15. . 47
6.3.4 Sensitivity to the convergence threshold τ on FedMoECap-P (BoolQ, NVIDIA A100). Percentages are relative to the default threshold of 0.15. 47
6.3.5 Sensitivity to the convergence threshold τ on FedMoECap-R (CommonsenseQA, NVIDIA Jetson AGX Orin). Percentages are relative to the default threshold of 0.15. . 48
6.3.6 Sensitivity to the convergence threshold τ on FedMoECap-R (BoolQ, NVIDIA A100). Percentages are relative to the default threshold of 0.15. 48
6.3.7 Sensitivity to the patience P on FedMoECap-P (CommonsenseQA, NVIDIA Jetson AGX Orin). Percentages are relative to the default patience of 2. 48
6.3.8 Sensitivity to the patience P on FedMoECap-P (BoolQ, NVIDIA A100). Percentages are relative to the default patience of 2. 49

6.3.9 FedMoECap-P with patience $P = 4$ vs FedMoECap-S as reference, both with an initial freeze rate of 0.90. Values indicate the change relative to FedMoECap-S (BoolQ on NVIDIA A100 and CommonsenseQA on NVIDIA Jetson AGX Orin). 49
6.3.10 Sensitivity to the patience P on FedMoECap-R (CommonsenseQA, NVIDIA Jetson AGX Orin). Percentages are relative to the default patience of 2. 50
6.3.11 Sensitivity to the patience P on FedMoECap-R (BoolQ, NVIDIA A100). Percentages are relative to the default patience of 2. 50
6.3.12 Accuracy comparison between the Activation Freeze Criterion and ESFT as the expert selection method within FedMoECap-S, with the number of trainable experts matched between the two criteria. 51
6.3.13 Comparison of FedMoECap-P with freeze rate 0 against FedAvg-LoRA and FedMoECap-S (CommonsenseQA, NVIDIA Jetson AGX Orin). Percentages are relative to FedAvg-LoRA (All Experts). 52
6.3.14 Comparison of FedMoECap-P with freeze rate 0 against FedAvg-LoRA and FedMoECap-S (BoolQ, NVIDIA A100). Percentages are relative to FedAvg-LoRA (All Experts). 52

Εκτεταμένη Ελληνική Περίληψη

Εισαγωγή και Κίνητρο

Τα Μεγάλα Γλωσσικά Μοντέλα (Large Language Models, LLMs) έχουν μεταμορφώσει ριζικά το τοπίο της τεχνητής νοημοσύνης, επιτυγχάνοντας ανθρώπινο επίπεδο απόδοσης σε πλήθος γλωσσικών εργασιών. Ωστόσο, η ταχεία ανάπτυξη και αξιοποίησή τους συνοδεύεται από σοβαρές απαιτήσεις υπολογιστικών πόρων, τόσο κατά το στάδιο της εκπαίδευσης όσο και κατά την εξειδίκευση (fine-tuning) σε νέες εργασίες. Παράλληλα, η αυξανόμενη ανάγκη για προστασία της ιδιωτικότητας και διατήρηση των δεδομένων τοπικά στις συσκευές των χρηστών αναδεικνύει τη σημασία της **Ομοσπονδιακής Μάθησης** (Federated Learning, FL) ως εναλλακτικής προσέγγισης για κατανεμημένη εκπαίδευση χωρίς κοινή χρήση ευαίσθητων δεδομένων.

Μια ιδιαίτερα ελπιδοφόρα αρχιτεκτονική για τον περιορισμό του υπολογιστικού κόστους των LLMs είναι η αρχιτεκτονική **Mixture of Experts** (MoE). Σε ένα μοντέλο MoE, κάθε επίπεδο του transformer περιέχει πολλαπλά εξειδικευμένα υποδίκτυα, τους *experts*, ενώ ένας μηχανισμός δρομολόγησης (router/-gating network) κατευθύνει κάθε token εισόδου μόνο στους πλέον κατάλληλους experts. Αυτή η *αραιή ενεργοποίηση* επιτρέπει στο μοντέλο να διαθέτει τεράστιο συνολικό αριθμό παραμέτρων — άρα και μεγάλη χωρητικότητα γνώσης — χωρίς να αυξάνει αναλογικά τον υπολογιστικό φόρτο ανά token.

Παρά τα πλεονεκτήματά τους, τα μοντέλα MoE εισάγουν νέες προκλήσεις στο πλαίσιο της Ομοσπονδιακής Μάθησης. Αρχικά, ο συνολικός αριθμός παραμέτρων ενός μοντέλου MoE υπερβαίνει κατά πολύ εκείνον ισοδύναμων πυκνών μοντέλων, καθιστώντας αδύνατη την πλήρη εκπαίδευσή τους σε συσκευές αιχμής (edge devices) με περιορισμένη μνήμη GPU (VRAM). Επιπλέον, η μεταφορά ενημερώσεων για το σύνολο των παραμέτρων σε κάθε γύρο επικοινωνίας θα απαιτούσε τεράστιο εύρος ζώνης. Τέλος, η ετερογένεια των συσκευών σε ένα ομοσπονδιακό σύστημα —όπου ενδέχεται να συνυπάρχουν ισχυροί διακομιστές με φορητές ή ενσωματωμένες συσκευές— καθιστά αδύνατη μια ενιαία στρατηγική εκπαίδευσης.

Η παρούσα εργασία αντιμετωπίζει αυτές ακριβώς τις προκλήσεις μέσω της πρότασης ενός νέου πλαισίου ομοσπονδιακής εξειδίκευσης, του **FedMoECap**.

Θεωρητικό Υπόβαθρο

Ομοσπονδιακή Μάθηση

Η Ομοσπονδιακή Μάθηση αναπτύχθηκε ως απάντηση στην ανάγκη εκπαίδευσης μοντέλων μηχανικής μάθησης χωρίς κεντρική συλλογή δεδομένων. Στον θεμελιώδη αλγόριθμο **FedAvg**, ο κεντρικός διακομιστής διανέμει το παγκόσμιο μοντέλο στους clients, οι οποίοι εκπαιδεύουν τοπικά και επιστρέφουν μόνο τις ενημερώσεις βαρών. Ο διακομιστής ενοποιεί τις ενημερώσεις μέσω σταθμισμένου μέσου όρου. Αυτή η προσέγγιση διασφαλίζει ότι τα ακατέργαστα δεδομένα παραμένουν στις τοπικές συσκευές, κάτι ιδιαίτερα σημαντικό σε εφαρμογές υγείας, νομικής ή οικονομικής φύσης και γενικότερα εφαρμογές που είναι αναγκαία η προστασία των δεδομένων των clients.

Παρά την αποδοτικότητά της, η ομοσπονδιακή μάθηση αντιμετωπίζει σοβαρές προκλήσεις: τα δεδομένα στους clients συνήθως δεν είναι ανεξάρτητα και ισόνομα κατανεμημένα (non-IID), γεγονός που οδηγεί σε

αποκλίνουσες τοπικές ενημερώσεις. Επιπλέον, η μεγάλη ετερογένεια μεταξύ clients — τόσο ως προς τις δυνατότητες υλικού όσο και ως προς το μέγεθος και την κατανομή των τοπικών δεδομένων — δυσχεραίνει τη σύγκλιση.

Αποδοτική Εξειδίκευση Παραμέτρων (PEFT)

Η πλήρης εκπαίδευση ενός LLM απαιτεί τεράστιους υπολογιστικούς πόρους. Για το λόγο αυτό, αναπτύχθηκαν μέθοδοι **Parameter-Efficient Fine-Tuning (PEFT)**, που τροποποιούν μόνο ένα μικρό υποσύνολο παραμέτρων. Η πλέον διαδεδομένη από αυτές είναι το **LoRA** (Low-Rank Adaptation), το οποίο εισάγει εκπαιδεύσιμες μήτρες χαμηλής τάξης παράλληλα με τις παγωμένες παραμέτρους του μοντέλου. Αντί να τροποποιεί απευθείας μια μήτρα βαρών W, το LoRA εκπαιδεύει δύο μήτρες A και B τέτοιες ώστε $\Delta W = BA$, όπου η τάξη $r \ll \min(d, k)$. Αυτό επιτρέπει σημαντική μείωση τόσο του υπολογιστικού κόστους όσο και του κόστους επικοινωνίας σε ομοσπονδιακό πλαίσιο, καθώς μόνο οι μήτρες A και B χρειάζεται να μεταδίδονται.

Αρχιτεκτονική Mixture of Experts

Σε ένα μοντέλο MoE, το υπο-επίπεδο feed-forward κάθε επιπέδου αντικαθίσταται από N παράλληλους experts. Ένα δίκτυο gating δρομολογεί κάθε token στους top-K_r experts, υπολογίζοντας για κάθε expert μια βαθμολογία gating $G(x)_i$ μέσω softmax. Η έξοδος του επιπέδου είναι το σταθμισμένο άθροισμα των εξόδων των επιλεγμένων experts: $y = \sum_{i \in \mathcal{K}} G(x)_i \cdot E_i(x)$. Εφόσον μόνο $K_r \ll N$ experts ενεργοποιούνται ανά token, το μοντέλο διατηρεί τον αριθμό των ενεργών floating-point operations στο επίπεδο ενός μικρότερου πυκνού μοντέλου, ενώ παράλληλα έχει πρόσβαση στην χωρητικότητα ενός πολύ μεγαλύτερου.

Σχετικές Εργασίες

Στο πλαίσιο της ομοσπονδιακής εξειδίκευσης LLMs, οι πρώτες προσεγγίσεις όπως τα FedAvg και FedProx αντιμετωπίζουν τα μοντέλα ως πυκνά δίκτυα, αγνοώντας εντελώς την αραιή δομή του MoE. Η μέθοδος FedIT εφαρμόζει ομοσπονδιακή εκπαίδευση σε σενάρια instruction tuning, αλλά επίσης δεν εκμεταλλεύεται τον δρομολογητή του MoE.

Από την πλευρά των μεθόδων ενήμερων για τους experts (expert-aware), η **ESFT** επιλέγει για εκπαίδευση μόνο τους experts που ενεργοποιούνται συχνότερα για μια συγκεκριμένη εργασία. Ωστόσο, χρησιμοποιεί κριτήριο επιλογής *ανά επίπεδο*, αναγκάζοντας την εκπαίδευση να κατανεμηθεί σε όλα τα επίπεδα ανεξάρτητα από τη σχετικότητά τους, και προϋποθέτει κεντρική πρόσβαση στα δεδομένα.

Άλλες εργασίες όπως το **FedMoE**, το **HFedMoE** και το **A^{3}SMoE** συνδυάζουν MoE με ομοσπονδιακή μάθηση, αλλά καθεμία αφήνει ανοικτές βασικές προκλήσεις: το FedMoE υπόκειται σε ακριβό διπλό στάδιο αρχικοποίησης και κριτήριο επιλογής ανά επίπεδο· το HFedMoE μετράει την σημαντικότητα των experts σε κάθε mini-batch και σε κάθε γύρο· το A^3SMoE προσαρμόζει τον αριθμό ενεργοποιημένων experts ανά client αλλά δεν διαχειρίζεται τη σύγκλισή τους. Καμία από αυτές τις μεθόδους δεν προσφέρει *διαχείριση σύγκλισης* experts κατά τη διάρκεια της εκπαίδευσης. Ακριβώς αυτό το κενό καλείται να καλύψει το FedMoECap.

Μεθοδολογία: Το Πλαίσιο FedMoECap

Το **FedMoECap** είναι ένα ολοκληρωμένο πλαίσιο ομοσπονδιακής εξειδίκευσης για μοντέλα MoE, σχεδιασμένο γύρω από τρεις βασικούς στόχους: **αποδοτικότητα** (μείωση VRAM, χρόνου, κόστους επικοινωνίας και ενέργειας), **εκπαίδευση βάσει της χρησιμότητας των experts** (εκπαίδευση μόνο των πλέον σχετικών experts για κάθε εργασία), και **υποστήριξη ετερογένειας** (συμμετοχή clients με διαφορετικούς υπολογιστικούς πόρους στον ίδιο γύρο εκπαίδευσης).

Κριτήριο Παγώματος Ενεργοποίησης (Activation Freeze Criterion)

Το πρώτο στοιχείο του FedMoECap είναι ένα κριτήριο επιλογής experts βασισμένο στις στατιστικές δρομολόγησης τους. Κατά τον πρώτο γύρο ομοσπονδιακής εκπαίδευσης, κάθε client εκτελεί μια *σκιαγράφιση* (profiling) πάνω στα τοπικά του δεδομένα, καταγράφοντας τη συχνότητα ενεργοποίησης κάθε expert.

Σε αντίθεση με προγενέστερες μεθόδους που εφαρμόζουν κριτήριο επιλογής ανά επίπεδο (per-layer), το FedMoECap *ομαδοποιεί όλους τους experts* από όλα τα επίπεδα σε μια ενιαία λίστα και επιλέγει τους top-$(1 - f_k)\%$ ανεξάρτητα από τη θέση τους στο δίκτυο:

$$E_{\text{train}} = \text{Top}_{(1-f_k)\%} \left(\bigcup_{l=1}^{L} E_l \right)$$

όπου $f_k \in [0, 1)$ είναι το **ποσοστό παγώματος** (freeze rate) του client k. Αυτή η παγκόσμια κατάταξη επιτρέπει το *πάγωμα ολόκληρων επιπέδων* όταν κανένας expert τους δεν κατατάσσεται ψηλά, βελτιστοποιώντας το παγκόσμιο προϋπολογισμό εκπαίδευσης σε αντίθεση με μεθόδους ανά επίπεδο, όπου μπορεί experts με χαμηλή συχνότητα ενεργοποιήσης να επιλεχθούν για να ξεπεράσουν το κατώφλι που έχει τεθεί στο επίπεδό τους.

Εφόσον ο δρομολογητής παραμένει παγωμένος καθ' όλη τη διάρκεια της εκπαίδευσης, η συμπεριφορά δρομολόγησης — και άρα η κατάταξη των experts — δεν αλλάζει, οπότε το αποτέλεσμα της αρχικής σκιαγράφισης παραμένει έγκυρο για όλους τους γύρους.

Επιλεκτικό LoRA

Αφού προσδιοριστεί το σύνολο εκπαιδεύσιμων experts, το FedMoECap εφαρμόζει LoRA *αποκλειστικά* σε αυτούς. Για κάθε εκπαιδεύσιμο expert e στο επίπεδο l, εισάγεται ένας adapter χαμηλής τάξης: $W_{l,e} = W^0_{l,e} + B_{l,e}A_{l,e}$, ενώ οι παγωμένοι experts δεν λαμβάνουν κανέναν adapter και δεν συμβάλλουν με εκπαιδεύσιμες παραμέτρους.

Επειδή στο μοντέλο OLMoE-1B-7B περίπου 93% των παραμέτρων βρίσκονται στα βάρη των experts, ένα ποσοστό παγώματος $f_k = 0.75$ συνδυασμένο με τάξη LoRA $r = 8$ αποφέρει μείωση των εκπαιδεύσιμων παραμέτρων κατά $\approx 75\%$ σε σχέση με την εφαρμογή LoRA σε όλους τους experts. Κατά την ομοσπονδιακή ενοποίηση, μόνο οι μήτρες LoRA $\{B_{l,e}, A_{l,e}\}$ των εκπαιδεύσιμων experts μεταδίδονται, μειώνοντας δραστικά το εύρος ζώνης ανά γύρο.

Διαχείριση Ετερογένειας Clients

Κάθε client ορίζει ανεξάρτητα το ποσοστό παγώματος f_k του. Clients με περιορισμένη μνήμη ή εύρος ζώνης μπορούν να θέσουν υψηλό f_k (π.χ. 0.95), εκπαιδεύοντας μόνο ένα ελάχιστο υποσύνολο experts και μεταδίδοντας πολύ μικρές ενημερώσεις. Αντίθετα, πιο ισχυροί clients μπορούν να θέσουν χαμηλότερο f_k (π.χ. 0.80), συνεισφέροντας περισσότερο στην εκπαίδευση. Αυτό επιτρέπει σε συσκευές πολύ διαφορετικής ισχύος — όπως ένα NVIDIA Jetson AGX Orin και ένας διακομιστής DGX — να συμμετέχουν στον ίδιο γύρο εκπαίδευσης.

Κριτήριο Σύγκλισης και Τρεις Στρατηγικές

Ένα βασικό καινοτόμο στοιχείο του FedMoECap είναι η ικανότητά του να ανιχνεύει τη σύγκλιση κάθε expert και να ενεργεί αναλόγως. Μετά από κάθε γύρο ενοποίησης, ο διακομιστής υπολογίζει τη *σχετική ενημέρωση παραμέτρων* κάθε εκπαιδεύσιμου expert:

$$\delta^{(t)}_{l,e} = \frac{\left\| \theta^{(t)}_{l,e} - \theta^{(t-1)}_{l,e} \right\|_2}{\left\| \theta^{(t)}_{l,e} \right\|_2 + \varepsilon}$$

Ένας expert θεωρείται συγκλίνων εάν $\delta^{(t)}_{l,e} < \tau$ για P συνεχόμενους γύρους (μηχανισμός *υπομονής*, patience).

Πάνω σε αυτό το κριτήριο, το FedMoECap προσφέρει τρεις στρατηγικές:

FedMoECap-S (Στατική). Η επιλογή experts πραγματοποιείται άπαξ κατά τον πρώτο γύρο και παραμένει αμετάβλητη για ολόκληρη την εκπαίδευση. Η στρατηγική αυτή είναι η απλούστερη και η πλέον αποδοτική ως προς το εύρος ζώνης, αποτελώντας ιδανική επιλογή για αναπτύξεις με αυστηρό και γνωστό εκ των προτέρων προϋπολογισμό επικοινωνίας.

FedMoECap-P (Κλάδεμα βάσει Σύγκλισης). Μόλις ένας expert ικανοποιήσει το κριτήριο σύγκλισης, παγώνεται μόνιμα και αφαιρείται από το ενεργό σύνολο εκπαίδευσης. Το αποτέλεσμα είναι ένα μονοτονικά μειούμενο σύνολο εκπαιδεύσιμων παραμέτρων: κόστος επικοινωνίας, VRAM και χρόνος εκπαίδευσης μειώνονται σταδιακά σε κάθε γύρο. Η στρατηγική αυτή ενδείκνυται όταν η ελαχιστοποίηση κόστους ενέργειας ή συνολικού χρόνου εκπαίδευσης είναι ο κύριος στόχος.

FedMoECap-R (Κυλιόμενη Αντικατάσταση). Αντί να παγώνει οριστικά τους συγκλίνοντες experts, η στρατηγική Rolling τους *αντικαθιστά* με προηγουμένως παγωμένους experts από τη λίστα κατάταξης. Κάθε φορά που ένας expert συγκλίνει, ο επόμενος σε σειρά κατάταξης expert ενεργοποιείται με νέους adapters LoRA. Αυτό επιτρέπει τελικά στο FedMoECap-R να εκπαιδεύσει ένα *μεγαλύτερο ποσοστό* των συνολικών experts σε σχέση με τις άλλες δύο στρατηγικές, διατηρώντας σταθερό τον αριθμό εκπαιδεύσιμων experts ανά γύρο. Προκειμένου να αποφευχθεί η καταστροφική λήθη (catastrophic forgetting), εφαρμόζεται ένας **υβριδικός κανόνας ενοποίησης** που συνδυάζει τις ενημερώσεις από clients που εκπαιδεύουν ακόμη έναν expert με εκείνες από clients που τον έχουν ήδη παγώσει.

Πειραματική Αξιολόγηση

Πειραματικό Σενάριο

Όλα τα πειράματα χρησιμοποιούν το **OLMoE-1B-7B-0125**, ένα αραιό μοντέλο MoE του Allen Institute for AI με ≈ 1 δισεκατομμύριο ενεργές παραμέτρους ανά πάσο εμπρόσθιας διάδοσης και 7 δισεκατομμύρια παραμέτρους συνολικά. Για τη μείωση του αποτυπώματος μνήμης, εφαρμόζεται ποσοτικοποίηση **INT8** σε όλα τα βάρη.

Η υποδομή υλικού αποτελείται από δύο επίπεδα που αντιπροσωπεύουν τα δύο άκρα του φάσματος ομοσπονδιακών αναπτύξεων. Το **NVIDIA Jetson AGX Orin** αποτελεί αντιπροσωπευτική ενσωματωμένη συσκευή αιχμής με ενοποιημένη αρχιτεκτονική μνήμης και αυστηρούς περιορισμούς ισχύος, ενώ το **NVIDIA A100 (40 GB)** σε διακομιστή DGX προσομοιώνει έναν ισχυρό θεσμικό client.

Η αξιολόγηση πραγματοποιείται σε τέσσερα benchmarks λογικής κοινής αίσθησης (commonsense reasoning): **PIQA** (δίτιμες επιλογές φυσικής διαίσθησης), **CommonsenseQA** (πολλαπλής επιλογής με 5 απαντήσεις), **BoolQ** (ναι/όχι ερωτήσεις) και **Winogrande** (ανάλυση αναφοράς). Για τα PIQA, CommonsenseQA και BoolQ χρησιμοποιείται κατανομή Dirichlet ($\alpha = 0.3$) για τη δημιουργία μη-IID τοπικών συνόλων δεδομένων, ενώ το Winogrande αξιολογείται σε IID ρύθμιση. Ο αριθμός γύρων ομοσπονδιακής επικοινωνίας κυμαίνεται από 12 έως 15, με προεπιλεγμένο ποσοστό παγώματος $f_k = 0.90$, κατώφλι σύγκλισης $\tau = 0.15$ και υπομονή $P = 2$.

Ακρίβεια Μοντέλου

Το **FedAvg-LoRA (All Experts)**, το οποίος εφαρμόζει LoRA σε όλους τους experts σε κάθε γύρο, χρησιμεύει ως κύρια βάση σύγκρισης. Συγκρίνεται επίσης με το αρχικό **Βασικό Μοντέλο** χωρίς καμία εξειδίκευση. Ο Πίνακας 0.0.1 συνοψίζει την ακρίβεια όλων των μεθόδων στα τέσσερα benchmarks.

Τα αποτελέσματα αποκαλύπτουν τέσσερις βασικές παρατηρήσεις.

Πρώτον, και οι τρεις παραλλαγές του FedMoECap **υπερτερούν σημαντικά του Βασικού Μοντέλου** σε κάθε benchmark. Στο CommonsenseQA, το FedMoECap-S βελτιώνει την ακρίβεια κατά

Μέθοδος	**PIQA (%)**	**CommQA (%)**	**BoolQ (%)**	**Wino (%)**
Βασικό Μοντέλο	78.73	52.33	69.97	68.82
FedAvg-LoRA (All Experts)	80.85	70.43	84.28	79.48
FedMoECap-S	80.00	67.24	82.11	76.16
FedMoECap-P	79.92	62.41	77.43	72.22
FedMoECap-R	–	65.60	80.24	74.66

Table 0.0.1: Σύγκριση ακρίβειας στα τέσσερα benchmarks αξιολόγησης. Το σύμβολο "–" υποδηλώνει αποτέλεσμα που δεν ήταν διαθέσιμο κατά τη συγγραφή.

σχεδόν 15 ποσοστιαίες μονάδες (από 52.33% σε 67.24%), επιβεβαιώνοντας ότι ακόμη και η επιλεκτική εκπαίδευση του 10% των experts παράγει ουσιαστικό μαθησιακό σήμα.

Δεύτερον, το **FedMoECap-S** επιτυγχάνει την υψηλότερη ακρίβεια μεταξύ των τριών παραλλαγών και κλείνει το μεγαλύτερο μέρος του χάσματος από τη βάση σε όλα τα benchmarks. Στο PIQA, φτάνει στο 80.00%, μόλις 0.85 μονάδες κάτω από το 80.85% της πλήρους βάσης, παρά το γεγονός ότι εκπαιδεύει μόνο το 10% των experts. Τα χάσματα στο BoolQ (2.17 μονάδες) και στο CommonsenseQA (3.19 μονάδες) είναι επίσης μικρά. Το μεγαλύτερο χάσμα εμφανίζεται στο Winogrande (3.32 μονάδες).

Τρίτον, το **FedMoECap-P** παρουσιάζει μεγαλύτερη πτώση ακρίβειας σε σχέση με το FedMoECap-S σε όλα τα benchmarks. Η διαφορά είναι μεγαλύτερη στο CommonsenseQA (62.41% έναντι 67.24%, διαφορά 4.83 μονάδων) και μικρότερη στο PIQA (79.92% έναντι 80.00%, μόλις 0.08 μονάδες). Αυτό αντικατοπτρίζει το προοδευτικό πάγωμα experts της στρατηγικής Pruning: καθώς οι experts συγκλίνουν και αφαιρούνται οριστικά από την εκπαίδευση, ο αριθμός εκπαιδεύσιμων experts μειώνεται, περιορίζοντας την ικανότητα προσαρμογής στη μικτή κατανομή εργασιών.

Τέταρτον, το **FedMoECap-R** καταλαμβάνει σταθερά μια ενδιάμεση θέση μεταξύ FedMoECap-S και FedMoECap-P. Στο CommonsenseQA σκοράρει 65.60%, στο BoolQ 80.24% και στο Winogrande 74.66%, ξεπερνώντας το FedMoECap-P κατά 1.64, 1.87 και 1.50 μονάδες αντίστοιχα. Αυτή η κατάταξη αντικατοπτρίζει την ικανότητα της Rolling στρατηγικής να εκθέτει μεγαλύτερο ποσοστό του συνόλου experts σε εκπαίδευση, αντικαθιστώντας τους συγκλίνοντες experts με νέους αντί να τους παγώνει οριστικά.

Η συνολική εικόνα ακρίβειας απεικονίζεται στο Σχήμα 6.2.1 της κύριας εργασίας, όπου φαίνεται καθαρά η ιεράρχηση FedAvg-LoRA > FedMoECap-S > FedMoECap-R > FedMoECap-P > Βασικό Μοντέλο.

Αποδοτικότητα Συστήματος: Συσκευή Αιχμής (Jetson)

Ο Πίνακας 0.0.2 αναφέρει την κατανάλωση πόρων για την εργασία CommonsenseQA στο NVIDIA Jetson AGX Orin, το πλέον απαιτητικό σενάριο λόγω των αυστηρών περιορισμών ισχύος και θερμοκρασίας της συσκευής.

Μέθοδος	**VRAM**	**Εύρος ζώνης (MB)**	**Χρόνος (s/batch)**	**Ενέργεια (Wh)**
FedAvg-LoRA (All Experts)	–	37,972	16.04	123.15
FedMoECap-S	−9.1%	3,801 (−90.0%)	13.05 (−18.6%)	104.01 (−15.5%)
FedMoECap-P	−9.2%	1,603 (−95.8%)	13.00 (−19.0%)	60.56 (−50.8%)
FedMoECap-R	−8.8%	3,835 (−89.9%)	13.22 (−17.6%)	109.05 (−11.4%)

Table 0.0.2: Αποδοτικότητα συστήματος στο NVIDIA Jetson AGX Orin (εργασία CommonsenseQA). Τα ποσοστά σε παρενθέσεις εκφράζουν μεταβολή σε σχέση με τη βάση FedAvg-LoRA (All Experts).

Τα αποτελέσματα αναδεικνύουν τρεις διαφορετικές πηγές εξοικονόμησης.

Η πιο εντυπωσιακή είναι η **μείωση του εύρους ζώνης επικοινωνίας**. Το FedMoECap-S και το FedMoECap-R μειώνουν το εύρος ζώνης κατά ≈ 90%, ενώ το FedMoECap-P το μειώνει κατά 95.8%, ρίχνοντας τη συνολική κίνηση από ≈ 38 GB σε κάτω από 2 GB. Η μείωση αυτή οφείλεται αποκλεισ-

τικά στο γεγονός ότι μόνο οι LoRA adapters των ενεργά εκπαιδευόμενων experts (10% του συνόλου) μεταδίδονται, χωρίς ποτέ να αποστέλλονται LoRA adapters των παγωμένων experts.

Η δεύτερη σημαντική πηγή εξοικονόμησης είναι η **κατανάλωση ενέργειας**. Το FedMoECap-P επιτυγχάνει μείωση 50.8%, περίπου υποδιπλασιάζοντας την ενέργεια που καταναλώνεται κατά τη διάρκεια ολόκληρης της εκπαίδευσης. Αυτή η προαναφερθήσα εξοικονόμηση εξηγείται από το προοδευτικό πάγωμα experts: καθώς οι γύροι εξελίσσονται, ο ενεργός φόρτος εκπαίδευσης συρρικνώνεται, μειώνοντας σταδιακά την κατανάλωση ισχύος. Μάλιστα λόγω του προοδευτικού παγώματος μπορούν να μειωθούν και οι συνολικοί γύροι εκπαίδευσης, αφού μπορεί στους τελευταίους γύρους να έχουν συγκλίνει όλοι οι experts που έχουν επιλεχθεί από το Κριτήριο Παγώματος Ενεργοποίησης σε έναν ή περισσότερους clients. Το FedMoECap-S εξοικονομεί 15.5% και το FedMoECap-R 11.4%.

Τέλος, και οι τρεις παραλλαγές μειώνουν τη χρησιμοποιούμενη **VRAM** κατά $\approx 9\%$ και τον **χρόνο ανά batch** κατά 17–19%, αποτελέσματα σημαντικά για μια συσκευή όπου κάθε megabyte και κάθε δευτερόλεπτο υπολογισμού σχετίζονται άμεσα με το κατά πόσον είναι εφικτή η ομοσπονδιακή μάθηση.

Αποδοτικότητα Συστήματος: Διακομιστής (A100)

Ο Πίνακας 0.0.3 αναφέρει την κατανάλωση πόρων για την εργασία BoolQ στο NVIDIA A100. Σε αυτή την πλατφόρμα, η VRAM δεν αποτελεί δεσμευτικό περιορισμό, οπότε οι κρίσιμες μετρικές είναι το εύρος ζώνης, η καθυστέρηση ανά batch και ο συνολικός χρόνος εκπαίδευσης ανά client.

Μέθοδος	**VRAM**	**Εύρος ζώνης (MB)**	**Χρόνος (s/batch)**	**Χρόνος/client (s)**
FedAvg-LoRA (All Experts)	–	37,963	8.69	2,923.42
FedMoECap-S	−7.4%	3,853 (−89.8%)	7.24 (−16.7%)	2,422.52 (−17.1%)
FedMoECap-P	−13.5%	1,789 (−95.3%)	6.71 (−22.8%)	1,528.05 (−47.2%)
FedMoECap-R	−7.3%	3,917 (−89.7%)	7.23 (−16.8%)	2,422.24 (−17.1%)

Table 0.0.3: Αποδοτικότητα συστήματος στο NVIDIA A100 (εργασία BoolQ). Τα ποσοστά σε παρενθέσεις εκφράζουν μεταβολή σε σχέση με τη βάση FedAvg-LoRA (All Experts).

Τα αποτελέσματα στον A100 επιβεβαιώνουν και επεκτείνουν τα ευρήματα από το Jetson. Η μείωση εύρους ζώνης 90–95% διατηρείται πλήρως σε υλικό επιπέδου διακομιστή, αποδεικνύοντας ότι η εξοικονόμηση αυτή είναι δομική ιδιότητα του αλγορίθμου και όχι αρτεφάκτο συγκεκριμένου υλικού.

Η πιο αξιοσημείωτη παρατήρηση στον A100 είναι η **μείωση του χρόνου εκπαίδευσης** που επιτυγχάνει το FedMoECap-P: ο μέσος χρόνος ανά client μειώνεται κατά 47.2%, από 2,923 δευτερόλεπτα σε μόλις 1,528 δευτερόλεπτα. Αυτό είναι άμεσο αποτέλεσμα του προοδευτικού παγώματος experts: στους τελευταίους γύρους, ελάχιστοι experts (ή και κανένας) παραμένουν ενεργοί, μειώνοντας δραστικά τον υπολογιστικό φόρτο ανά batch.

Ανάλυση Σχεδιαστικών Επιλογών και Υπερπαραμέτρων

Ποσοστό Παγώματος. Ο Πίνακας 0.0.4 εξετάζει τρεις τιμές γύρω από την προεπιλογή $f_k = 0.90$ για το FedMoECap-S (CommonsenseQA, Jetson).

Ποσοστό Παγώματος	**VRAM**	**Εύρος ζώνης (MB)**	**Χρόνος (s/batch)**	**Ενέργεια (Wh)**	**Ακρίβεια (%)**
0.80 (Περισσότερη εκπ.)	+1.2%	7,645 (+101.1%)	13.33 (+2.1%)	106.88 (+2.8%)	69.37
0.90 (Προεπιλογή)	–	3,801	13.05	104.01	67.24
0.95 (Υψηλό πάγωμα)	−0.6%	1,913 (−49.7%)	12.74 (−2.4%)	102.53 (−1.4%)	66.34

Table 0.0.4: Επίδραση του ποσοστού παγώματος στο FedMoECap-S (CommonsenseQA, NVIDIA Jetson AGX Orin). Τα ποσοστά εκφράζουν μεταβολή σε σχέση με την προεπιλογή 0.90.

Τα αποτελέσματα αποκαλύπτουν ένα συνεπές μοτίβο: το **εύρος ζώνης** είναι μακράν η πιο ευαίσθητη μετρική στο ποσοστό παγώματος. Διπλασιασμός του αριθμού εκπαιδεύσιμων experts (από $f_k = 0.90$ σε 0.80) διπλασιάζει επίσης το εύρος ζώνης (+101%), ενώ το κέρδος ακρίβειας είναι σεμνό (+2.13 μονάδες).

Η ενέργεια και ο χρόνος batch παραμένουν σχεδόν αμετάβλητοι (+2.1% και +2.8% σε Jetson και A100 αντίστοιχα), επιβεβαιώνοντας ότι η κατανάλωση ισχύος οφείλεται κυρίως στον σταθερό υπολογιστικό φόρτο του συστήματος και όχι στις LoRA ενημερώσεις. Από την άλλη, αύξηση σε $f_k = 0.95$ υποδιπλασιάζει το εύρος ζώνης (−49.7%) αλλά μειώνει την ακρίβεια κατά 0.90 μονάδες, επιβεβαιώνοντας το 0.90 ως την καλύτερη ισορροπία.

Κατώφλι Σύγκλισης τ. Το κατώφλι τ ελέγχει πόσο επιθετικά το σύστημα ανιχνεύει τη σύγκλιση ενός expert. Για το **FedMoECap-P**, η επίδρασή του είναι ιδιαίτερα σημαντική: αυστηρότερο $\tau = 0.10$ αποδίδει +2.62 μονάδες ακρίβειας στο CommonsenseQA και +3.00 στο BoolQ σε σχέση με χαλαρότερο $\tau = 0.20$, αλλά αυξάνει το εύρος ζώνης κατά 26.6% (Jetson) και 33.6% (A100) και την ενέργεια κατά 58.3%. Αντίθετα, για το **FedMoECap-R** το τ είναι σχεδόν αδιάφορο: η μέγιστη διαφορά ακρίβειας μεταξύ των τιμών $\tau \in \{0.10, 0.15, 0.20\}$ είναι 0.82 μονάδες στο CommonsenseQA και 1.59 μονάδες στο BoolQ, ενώ μετρικές πόρων παραμένουν σχεδόν αμετάβλητες, αποδεικνύοντας ότι η μηχανή αντικατάστασης experts του Rolling διατηρεί σταθερό φόρτο ανεξάρτητα από το τ.

Υπομονή P. Ο Πίνακας 0.0.5 παρουσιάζει την επίδραση της υπομονής P για το FedMoECap-P (CommonsenseQA, Jetson).

Υπομονή (P)	VRAM	Εύρος ζώνης (MB)	Χρόνος (s/batch)	Ενέργεια (Wh)	Ακρίβεια (%)
2 (Προεπιλογή)	–	1,603	13.00	60.56	62.41
3	−3.8%	1,921 (+19.8%)	12.47 (−4.1%)	70.68 (+16.7%)	63.64
4	−2.4%	2,238 (+39.6%)	12.64 (−2.8%)	74.00 (+22.2%)	64.29

Table 0.0.5: Επίδραση της υπομονής P στο FedMoECap-P (CommonsenseQA, NVIDIA Jetson AGX Orin). Τα ποσοστά εκφράζουν μεταβολή σε σχέση με την προεπιλογή $P = 2$.

Μεγαλύτερη υπομονή βελτιώνει σταδιακά την ακρίβεια (από 62.41% για $P = 2$ έως 64.29% για $P = 4$, +1.88 μονάδες), διότι οι experts πρέπει να επιδείξουν πιο εμπεδωμένη σύγκλιση πριν παγώσουν. Το κόστος είναι αύξηση του εύρους ζώνης κατά $\approx 20\%$ ανά επιπλέον στάδιο υπομονής (+19.8% για $P = 3$, +39.6% για $P = 4$) και αντίστοιχη αύξηση ενέργειας. Σημαντική παρατήρηση: οι φαινομενικές μειώσεις VRAM και χρόνου batch για υψηλές τιμές υπομονής είναι *στατιστική ψευδαίσθηση*. Καθώς η υπομονή αυξάνεται, οι experts καθυστερούν να παγώσουν, οπότε η εκπαίδευση διανύει περισσότερους γύρους στους οποίους ελάχιστοι experts παραμένουν ενεργοί. Αυτοί οι γύροι χαμηλής δραστηριότητας «τραβούν προς τα κάτω» τους μέσους όρους ανά γύρο για VRAM και χρόνο batch, χωρίς να αντικατοπτρίζουν πραγματικές εξοικονομήσεις κατά τη διάρκεια της ενεργής φάσης εκπαίδευσης.

Μια ενδιαφέρουσα σύγκριση είναι το FedMoECap-P με $P = 4$ έναντι FedMoECap-S, καθώς το $P = 4$ επιτυγχάνει την υψηλότερη ακρίβεια μεταξύ όλων των ρυθμίσεων FedMoECap-P. Σε σχέση με το FedMoECap-S, το FedMoECap-P ($P = 4$) μειώνει το εύρος ζώνης κατά 37–41% και τον χρόνο εκπαίδευσης κατά 25–35%, με κόστος ακρίβειας 2–3 μονάδων, αποτελώντας ένα χρήσιμο και διακριτό σημείο στην καμπύλη αποδοτικότητας–ακρίβειας.

Υπομονή P για το FedMoECap-R. Σε έντονη αντίθεση με το FedMoECap-P, η αύξηση της υπομονής στο FedMoECap-R αφήνει ουσιαστικά αμετάβλητες *όλες* τις μετρικές πόρων: το εύρος ζώνης μεταβάλλεται κατά $\leq 0.6\%$, η ενέργεια κατά $\leq 2.3\%$ και ο χρόνος εκπαίδευσης κατά $\leq 0.5\%$ σε σχέση με την προεπιλογή $P = 2$. Αυτό δεν είναι στατιστικό αρτεφάκτο: ο μηχανισμός αντικατάστασης experts του Rolling διατηρεί σταθερό τον αριθμό ενεργών experts ανά γύρο ανεξάρτητα από το πότε κηρύσσεται σύγκλιση. Παρόλα αυτά, η ακρίβεια βελτιώνεται σταθερά: +1.39 μονάδες στο CommonsenseQA (από 65.60% για $P = 2$ σε 66.99% για $P = 4$) και +0.68 μονάδες στο BoolQ, χωρίς ουσιαστικό κόστος πόρων. Το εύρημα αυτό αποκαλύπτει ένα θεμελιώδες χαρακτηριστικό της Rolling στρατηγικής: η *βάθος* εκπαίδευσης στους πλέον task-relevant experts υπερτερεί έναντι της *εύρους* κάλυψης μεγαλύτερου αριθμού experts, εύρημα συνεπές με το κίνητρο του Κριτηρίου Παγώματος Ενεργοποίησης.

Σύγκριση Κριτηρίου Παγώματος Ενεργοποίησης με ESFT. Για την αξιολόγηση της αποτελεσματικότητας της παγκόσμιας κατάταξης experts, συγκρίνουμε το Κριτήριο Παγώματος Ενερ-

γοποίησης με το κριτήριο επιλογής της ESFT, βαθμονομημένο έτσι ώστε ο συνολικός αριθμός εκπαιδεύσιμων experts να είναι ισοδύναμος μεταξύ των δύο μεθόδων. Επειδή τα σύνολα εκπαιδεύσιμων experts είναι ισάριθμα, όλες οι μετρικές πόρων αναμένονται να είναι ουσιαστικά ίσες και η σύγκριση εστιάζει αποκλειστικά στην ακρίβεια. Το Κριτήριο Παγώματος Ενεργοποίησης υπερτερεί κατά +0.49 ποσοστιαίες μονάδες τόσο στο CommonsenseQA (67.24% έναντι 66.75%) όσο και στο BoolQ (82.11% έναντι 81.62%). Το πλεονέκτημα αποδίδεται στην παγκόσμια κατάταξη: αντί να κατανέμει εκπαιδεύσιμα slots ανά επίπεδο ανεξάρτητα από τη συνολική σημαντικότητά τους, το FedMoECap συγκεντρώνει τον προϋπολογισμό εκπαίδευσης στους πλέον ενεργοποιούμενους experts του δικτύου συνολικά, επιτρέποντας ακόμη και το πλήρες πάγωμα επιπέδων με χαμηλή σχετικότητα. Επιπλέον, το ποσοστό παγώματος f_k ελέγχει άμεσα και διαφανώς το κλάσμα των εκπαιδεύσιμων experts, σε αντίθεση με το κατώφλι ανά επίπεδο της ESFT, του οποίου η επίδραση εξαρτάται από τη συγκεκριμένη κατανομή δρομολόγησης.

FedMoECap-P με Ποσοστό Παγώματος 0. Αυτό το πείραμα εξετάζει μια εναλλακτική λειτουργία του FedMoECap-P στην οποία το αρχικό ποσοστό παγώματος τίθεται $f_r = 0$, δηλαδή *όλοι* οι experts είναι εκπαιδεύσιμοι από τον πρώτο γύρο. Αντί να βασίζεται στο Κριτήριο Παγώματος Ενεργοποίησης, η στρατηγική Pruning ανακαλύπτει μόνη της ποιοι experts αξίζει να διατηρηθούν, παγώνοντας προοδευτικά εκείνους που συγκλίνουν. Το πιο αξιοσημείωτο εύρημα είναι ότι το FedMoECap-P $f_r = 0$ **επιτυγχάνει ακρίβεια πολύ κοντά στο FedMoECap-S**: 66.26% έναντι 67.24% στο CommonsenseQA (-0.98 μονάδες) και 81.87% έναντι 82.11% στο BoolQ (-0.24 μονάδες), χωρίς καθόλου χρήση δρομολόγησης για την επιλογή experts. Ωστόσο, αυτή η ισοδυναμία ακρίβειας συνοδεύεται από σημαντικό κόστος πόρων: 17,266 MB εύρους ζώνης στο Jetson (-54.5% έναντι FedAvg, αλλά $4-5\times$ περισσότερο από το FedMoECap-S) και 117.69 Wh ενέργειας, σχεδόν ίδια με τη βάση των 123.15 Wh. Το αποτέλεσμα αυτό επιβεβαιώνει ότι το Κριτήριο Παγώματος Ενεργοποίησης παρέχει εκ των προτέρων την ίδια ευθυγράμμιση experts που η Pruning ανακαλύπτει αργότερα μέσω σύγκλισης, αλλά με κλάσμα του κόστους.

Ανάλυση Μετώπου Pareto

Η ανάλυση μετώπου Pareto παρουσιάζει συνοπτικά το αντάλλαγμα ακρίβειας–αποδοτικότητας για κάθε ρύθμιση που δοκιμάστηκε, εντοπίζοντας τις κυρίαρχες λύσεις. Τα διαγράμματα Pareto για ακρίβεια έναντι εύρους ζώνης και ακρίβεια έναντι ενέργειας/χρόνου εκπαίδευσης (Σχήματα 6.4.1 και 6.4.2 της κύριας εργασίας) αποκαλύπτουν τα εξής:

Στο **μέτωπο εύρους ζώνης**, το FedMoECap-S κυριαρχεί με τις τρεις ρυθμίσεις ποσοστού παγώματος ($f_k \in \{0.80, 0.90, 0.95\}$) να ορίζουν μια ομαλή καμπύλη. Κινούμενοι από $f_k = 0.80$ σε $f_k = 0.95$, το εύρος ζώνης υποδιπλασιάζεται σε κάθε βήμα με οριακή μόνο απώλεια ακρίβειας, επιβεβαιώνοντας ότι το ποσοστό παγώματος αποτελεί έναν ακριβή και καλά συμπεριφερόμενο ρυθμιστή. Το FedMoECap-R δεν επιτυγχάνει Pareto-βέλτιστη απόδοση στον άξονα εύρους ζώνης.

Στο **μέτωπο ενέργειας και χρόνου εκπαίδευσης**, το FedMoECap-P κυριαρχεί σε όλη την περιοχή χαμηλών πόρων (κάτω από ≈ 100 Wh στο Jetson και ≈ 2050 δευτερόλεπτα στον A100), καθώς η προοδευτική μείωση του αριθμού ενεργών experts αποφέρει τις μεγαλύτερες εξοικονομήσεις. Πέρα από αυτές τις τιμές, το FedMoECap-S αποτελεί την καλύτερη επιλογή, ορίζοντας το μέτωπο στην περιοχή υψηλότερης ακρίβειας. Το FedMoECap-R *δεν κυριαρχεί σε κανένα από τα δύο μέτωπα*: στο εύρος ζώνης συγκεντρώνεται σε μια στενή κατακόρυφη ζώνη γύρω από τα 3800–3900 MB, όπου το FedMoECap-S παρέχει ανώτερη ακρίβεια, ενώ στο μέτωπο ενέργειας και χρόνου βρίσκεται πάνω από 100 Wh και 2400 δευτερόλεπτα αντίστοιχα, όπου το FedMoECap-S ήδη υπερτερεί.

Συμπεράσματα και Μελλοντική Εργασία

Η παρούσα διπλωματική εργασία παρουσίασε το **FedMoECap**, ένα ολοκληρωμένο πλαίσιο ομοσπονδιακής εξειδίκευσης για μοντέλα MoE που αντιμετωπίζει συστηματικά το τριπλό πρόβλημα αποδοτικότητας, εκμετάλλευσης της εξειδίκευσης ως προς τους experts και ετερογένειας clients.

Τα κύρια συμπεράσματα είναι: (α) Η παγκόσμια κατάταξη experts υπερτερεί του κριτηρίου ανά επίπεδο εκμεταλλευόμενη αποδοτικότερα τον διαθέσιμο προϋπολογισμό. (β) Τα ποσοστά παγώματος ανά client

αποτελούν έναν ακριβή μηχανισμό για την εξισορρόπηση ακρίβειας–πόρων σε ετερογενείς αναπτύξεις. (γ) Οι στρατηγικές Static, Pruning και Rolling σχηματίζουν μια συνεκτική οικογένεια λύσεων που καλύπτει τις κυρίαρχες λειτουργικές απαιτήσεις στην πράξη.

Το FedMoECap καταδεικνύει ότι η ομοσπονδιακή εξειδίκευση μεγάλων μοντέλων MoE σε ετερογενή, περιορισμένων πόρων περιβάλλοντα είναι εφικτή: μείωση εύρους ζώνης έως 95% και κατανάλωσης ενέργειας άνω του 50% μπορούν να επιτευχθούν διατηρώντας ταυτόχρονα την ακεραιότητα της ιδιωτικότητας και την ανταγωνιστική ακρίβεια.

Μελλοντικές κατευθύνσεις περιλαμβάνουν: βελτίωση της στρατηγικής Rolling με εναλλακτικούς κανόνες αντικατάστασης experts· αυτόματη εκτίμηση του βέλτιστου ποσοστού παγώματος από μετρικές υλικού (latency, μέγεθος δεδομένων)· αξιολόγηση σε άλλες εργασίες, όπως περίληψη και instruction following· και ανάπτυξη σε πραγματικά κατανεμημένα ομοσπονδιακά συστήματα για την εκτίμηση επιπτώσεων πραγματικής καθυστέρησης δικτύου και αποτυχιών clients.

Chapter 1

Introduction

The emergence of Large Language Models (LLMs) has redefined the boundaries of artificial intelligence, providing unprecedented performance in linguistic reasoning and human-machine interaction [1]. As these architectures continue to scale, the process of specializing them for domain-specific tasks via fine-tuning faces two primary hurdles: the massive computational cost of training and the necessity for high-quality, localized datasets [2].

1.1 The Rise of Federated Learning

A fundamental shift in how AI systems are deployed has been driven by the growing awareness of data privacy concerns and the practical reality that valuable training data is rarely centralized. In healthcare, finance, and mobile computing, sensitive records are distributed across thousands of institutions and devices, each bound by legal obligations such as the EU General Data Protection Regulation (GDPR) or the US Health Insurance Portability and Accountability Act (HIPAA), which restrict how personal data may be collected, transferred, and processed [3, 4]. Federated Learning (FL), introduced by McMahan et al. [5], was specifically designed to address this reality. In the FL paradigm, a central server orchestrates training across a population of clients, each of which computes gradient updates on its local private data and transmits only those updates—never the raw data—back to the server. The server aggregates the updates into a refined global model and redistributes it for the next round. This architecture has rapidly gained traction in industry and academia, enabling collaborative model improvement in scenarios where data pooling is legally or logistically impossible.

The appeal of FL extends beyond privacy. In mobile and IoT applications, data is generated continuously at the edge at a rate that makes its transmission to a central datacenter impractical. Training models directly on the devices where data is produced reduces latency, eliminates large-scale data transfer costs, and allows models to adapt to the local context of individual users or environments. These motivations have made FL a cornerstone of modern distributed machine learning [5, 6].

1.2 System Heterogeneity: The Defining Challenge of Federated Learning

Despite its appeal, FL deployments in the real world are far from the idealized setting assumed in early theoretical work. The most persistent and consequential challenge is **system heterogeneity**: the clients participating in a federated federation are not homogeneous machines running under equal conditions, but a diverse mixture of edge devices, mobile phones, embedded systems, and cloud instances with wildly different hardware capabilities [7]. A powerful workstation equipped with a high-end GPU can process hundreds of training samples per second, while a smartphone or an embedded

IoT sensor may struggle to sustain even minimal batch processing under tight memory and energy budgets.

This disparity has direct consequences for training dynamics. In every federated round, the server must wait for all participating clients to finish their local training before aggregation can proceed. The slowest client—the *straggler*—therefore determines the wall-clock time of the entire round [7]. More critically, resource-constrained clients may be entirely unable to load a large model into their limited VRAM, let alone fine-tune it. Standard FL protocols offer no principled mechanism for clients to adapt the training workload to their own hardware budget; a client that cannot execute the prescribed computation simply drops out, reducing the effective size of the federation and potentially introducing selection bias into the global model [8].

Energy consumption is a further dimension of heterogeneity that is often overlooked in the literature. Edge devices such as embedded GPUs operate on constrained power budgets, and prolonged training rounds can drain batteries or exceed thermal design constraints. For MoE models in particular, where the total parameter count is large even when only a fraction of experts are active during inference, the energy cost of full fine-tuning is prohibitive for most edge-class hardware.

1.3 The Promise and Challenges of Mixture-of-Experts in Federated Fine-Tuning

To mitigate the computational burden of fine-tuning, various Parameter-Efficient Fine-Tuning (PEFT) techniques have been developed, most notably Low-Rank Adaptation (LoRA), which significantly lowers the number of trainable parameters while preserving model performance [9, 2]. Parallel to these developments, the research community has embraced sparse architectures, particularly the **Mixture-of-Experts** (MoE) paradigm [10]. MoE models decouple total parameter count from per-token compute costs by activating only a small subset of specialized sub-networks (the *experts*) for each input token [10]. This allows models like **OLMoE-1B-7B** to maintain the inference budget of a 1B parameter model while possessing the knowledge capacity of a 7B parameter architecture [11].

The combination of MoE and FL is a natural target for research, since both paradigms independently reduce computational burden. However, their intersection introduces a compounded set of challenges.

Memory Footprint. While MoE models are efficient at inference time because only a fraction of experts fire per token, their *total* parameter count exceeds that of comparably capable dense models. Fine-tuning an MoE model requires loading all expert weights into VRAM, even those that will not be updated in a given round. For OLMoE-1B-7B, this means managing 7 billion parameters—far beyond the capacity of most edge devices operating in a federated setting.

Communication Overhead. In each federated round, the trainable parameters must be transmitted between clients and the server. For dense fine-tuning of a full MoE model, this implies broadcasting and collecting the weights of all expert networks across all layers at every round. Given that MoE models contain hundreds of expert FFN blocks, the resulting communication volume is enormous, rendering naive federated fine-tuning infeasible on bandwidth-limited connections [7].

Non-IID Routing Divergence. A subtler challenge specific to MoE architectures in FL is that the routing behavior of the gating network is data-dependent. Clients with different local data distributions will activate different subsets of experts during their local training steps. This means that the updates uploaded to the server do not uniformly cover all experts, and different clients may push conflicting gradient signals to the same expert weights, destabilizing convergence [12, 13].

Expert Allocation Under Heterogeneous Resources. Perhaps the most practically severe challenge is that system-heterogeneous clients cannot all afford to fine-tune the same number of experts. A server-class client may be capable of updating all experts per layer, while an edge device may only

sustain gradient computation over a much smaller subset. Standard fine-tuning protocols provide no mechanism for clients to declare an independent per-client training budget, forcing a lowest-common-denominator configuration that either excludes weaker clients or underutilizes stronger ones [7].

1.4 The Gap and the Proposed Solution

While expert-aware methods like **Expert Specialized Fine-Tuning** (ESFT) have been proposed [14] for *centralized* fine-tuning, there are currently a few PEFT approaches that leverage the MoE architecture within the federated paradigm. Most existing federated fine-tuning implementations treat MoE models as dense LLMs, failing to exploit the inherent sparsity of the expert routing signal for resource optimization. The few works that do consider MoE structure in FL rely on per-layer selection policies that ignore the global distribution of expert importance, or require costly preliminary fine-tuning phases that add significant overhead before main training can begin [12] or do constant per batch profiling of expert importance [13].

This thesis addresses the open problem of how to fine-tune a large MoE language model in a federated setting when clients possess heterogeneous hardware capabilities, operate under strict bandwidth and energy budgets, and hold private non-IID data that activates different experts on each participant. We propose **FedMoECap**, a framework that exploits the routing statistics of the MoE gating network to identify the globally most task-relevant experts, applies LoRA selectively to those experts only, and provides each client with the ability to independently specify a freeze rate that calibrates its training workload to its local resource budget. Three complementary convergence management strategies—Static, Pruning, and Rolling—allow FedMoECap to adapt its behavior to a spectrum of operational requirements, from strict fixed-budget communication to maximum expert coverage within a per-round resource cap.

1.5 Contributions

The contributions of this thesis can be summarized as follows:

- **Activation Freeze Criterion.** We propose a novel expert selection criterion that pools all experts from all layers of a MoE network into a single global set and selects the top $X\%$ by activation rate, ignoring per-layer boundaries. This allows the framework to freeze entire layers when warranted and maintain varying amounts of trainable experts across layers, providing more efficient resource allocation than per-layer threshold methods such as ESFT [14].
- **Selective LoRA for MoE Fine-Tuning.** We apply Low-Rank Adaptation (LoRA) [9] selectively to only the trainable experts identified by a manually specified per-client freeze rate. This minimizes the number of trainable parameters and the associated communication overhead during federated aggregation.
- **Three Adaptive Convergence Management Strategies.** We introduce a convergence criterion based on relative parameter updates and develop three complementary strategies to act on it:
  - *FedMoECap-S (Static):* Expert selection is performed once at the first round and held fixed, ideal for strict and unchanging bandwidth constraints.
  - *FedMoECap-P (Pruning):* Experts that are close to convergence during training are progressively frozen, continuously reducing the active parameter set, communication cost, training time, and energy consumed.
  - *FedMoECap-R (Rolling):* Converged experts are swapped out for previously frozen, untrained experts, maximizing the total number of experts trained within a fixed resource budget.

- **Federated Heterogeneity Handling.** We design a mechanism where each client can manually specify an independent freeze rate based on its local hardware capacity. This allows heterogeneous clients to participate in the same training round while each respecting its own hardware constraints.
- **Hybrid Aggregation to Prevent Catastrophic Forgetting.** For the Rolling variant, we propose a hybrid aggregation rule that utilizes a soft-weighting approach. This rule explicitly preserves the weights of converged clients while integrating updates from active clients, preventing the expert from forgetting knowledge accumulated on clients that have already reached convergence.
- **Comprehensive Experimental Evaluation.** We evaluate FedMoECap on the **OLMoE-1B-7B** model [11] across two hardware tiers: an **NVIDIA Jetson AGX Orin** edge device and an **NVIDIA A100/DGX** server. Our results demonstrate that FedMoECap reduces communication overhead by up to **95%** and total energy consumption by **50.8%** compared to training every expert in the LLM, while maintaining competitive accuracy on PIQA, BoolQ, CommonsenseQA and Winogrande benchmarks.

1.6 Thesis Structure

The thesis is organized into 7 chapters as follows:

Chapter 2 provides the theoretical foundation required to understand this work. We introduce the Transformer architecture and the Mixture-of-Experts (MoE) paradigm, explain the Federated Learning setting, and discuss parameter-efficient fine-tuning techniques like LoRA and ESFT.

Chapter 3 surveys the related literature, reviewing existing work on federated fine-tuning of large models, sparse model training, and resource-aware distributed learning.

Chapter 4 bridges the gap identified in the related work by presenting the sequence of empirical observations that shaped the design of FedMoECap. We show that expert routing is sparse, task-dependent, and stable under a frozen router, that expert importance is a global rather than per-layer property, that hardware heterogeneity must be treated as a first-class design parameter, and that expert utility diminishes during training in a non-uniform way. Each observation motivates a concrete component of the framework described in the following chapter.

Chapter 5 presents the **FedMoECap** framework in full detail. We describe the Activation Freeze Criterion, the application of Selective LoRA, and the manual per-client freeze rate mechanism for handling heterogeneity. We also detail the three adaptive strategies (Static, Pruning, Rolling) and the hybrid aggregation rule designed for the Rolling variant.

Chapter 6 presents our experimental evaluation. We describe the setup using the OLMoE-1B-7B model on NVIDIA Jetson AGX Orin and DGX hardware. We report results on model accuracy and system efficiency, followed by a design and hyperparameter analysis examining the impact of the freeze rate, convergence thresholds and patience.

Chapter 7 summarizes the findings of this thesis, discusses the limitations of the proposed approach, and outlines promising directions for future work.

Chapter 2

Theoretical Background

2.1 The Transformer Architecture

The Transformer architecture, introduced by Vaswani et al. [15], has become the cornerstone of modern Natural Language Processing (NLP). By shifting away from recurrent and convolutional structures, the Transformer relies entirely on attention mechanisms to capture global dependencies in input sequences [15].

2.1.1 Self-Attention Mechanism

The fundamental building block of the Transformer is the Scaled Dot-Product Attention. Given an input sequence represented as embeddings, the model computes three distinct matrices: Query (Q), Key (K), and Value (V). The attention output is calculated as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1.1}$$

where d_k is the dimension of the keys [15]. This mechanism allows each token in the sequence to attend to all other tokens, effectively capturing long-range context [15].

2.1.2 Multi-Head Attention

Rather than computing a single attention function, the Transformer applies attention h times in parallel across different learned linear projections of the input [15]. This is known as Multi-Head Attention (MHA), and it allows the model to jointly attend to information from different representation subspaces at different positions. Formally, MHA is defined as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\, W^O \tag{2.1.2}$$

where each head is computed as:

$$\text{head}_i = \text{Attention}(QW_i^Q,\, KW_i^K,\, VW_i^V) \tag{2.1.3}$$

and W_i^Q, W_i^K, W_i^V, and W^O are learnable projection matrices [15]. By attending in multiple subspaces simultaneously, the model captures a richer set of contextual relationships than any single attention head could alone.

2.1.3 The Transformer Block

A full Transformer block consists of two primary sublayers, each wrapped with a residual connection and Layer Normalization [15]. The first sublayer is Multi-Head Attention, and the second is a position-wise Feed-Forward Network (FFN). For an input x, the computations are:

$$x' = \text{LayerNorm}(x + \text{MultiHead}(x)) \tag{2.1.4}$$

$$\text{output} = \text{LayerNorm}(x' + \text{FFN}(x')) \tag{2.1.5}$$

The FFN is typically a two-layer MLP with a non-linear activation:

$$\text{FFN}(x) = \sigma(xW_1 + b_1)W_2 + b_2 \tag{2.1.6}$$

The FFN sublayer is particularly significant for this thesis, as the Mixture-of-Experts paradigm replaces it with a set of parallel expert networks, as described in the following section.

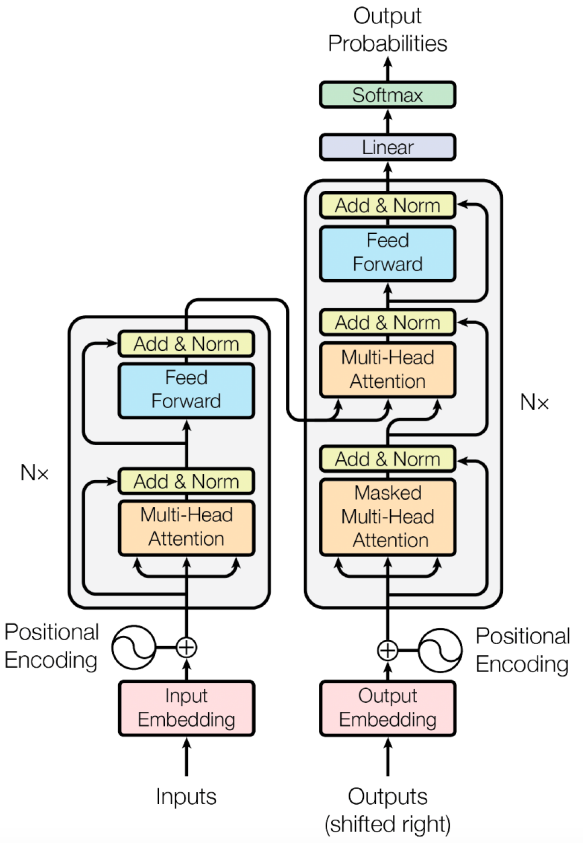


Figure 2.1.1: Transformer Architecture of Encoder-Decoder models [15]. Decoder-only models such as GPT, Gemini, and Claude use only the right (decoder) stack.

2.2 Mixture-of-Experts (MoE)

To scale model capacity without linearly increasing computational cost, sparse architectures such as the Mixture-of-Experts (MoE) have been proposed [10]. In an MoE layer, the standard dense Feed-Forward Network (FFN) is replaced by a set of N independent experts $\{E_1, E_2, \ldots, E_N\}$ [10, 11]. Only a small subset of these experts is activated for each input token, which decouples total model capacity from per-token compute cost.

2.2.1 Top-K Routing Mechanism

The gating network, or router $G(x)$, is responsible for selecting a sparse subset of experts for each token. The most common approach is Top-K routing, where only the k experts with the highest gating scores are activated:

$$G(x) = \text{Softmax}(\text{TopK}(H(x) + \epsilon)) \tag{2.2.1}$$

where $H(x)$ is typically a linear projection $H(x) = x \cdot W_g$ and ϵ is tunable noise added for load balancing [10]. The output of the MoE layer is then the weighted sum of the selected experts' outputs:

$$y = \sum_{i \in \mathcal{K}} G(x)_i \cdot E_i(x) \tag{2.2.2}$$

where $\mathcal{K}$ is the set of top-k expert indices. This allows models like **OLMoE-1B-7B** to possess 7 billion total parameters while only engaging approximately 1 billion parameters per token during a forward pass [11].

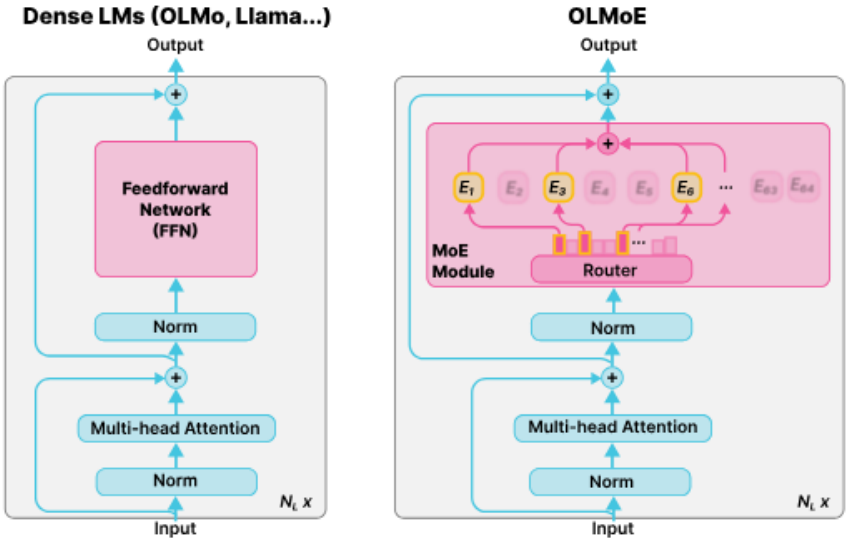


Figure 2.2.1: Mixture-of-Experts layer compared to a standard dense Feed-Forward layer. In the MoE layer, a router selects only a sparse subset of experts per token, keeping inference cost low while scaling total model capacity [11].

2.3 Parameter-Efficient Fine-Tuning (PEFT)

As model sizes grow, full fine-tuning of all parameters becomes computationally prohibitive, particularly on resource-constrained devices [2]. PEFT methods address this by updating only a small fraction of the model's weights while keeping the majority of pre-trained parameters frozen [9, 2]. This drastically reduces both memory requirements and communication costs in distributed settings.

2.3.1 Low-Rank Adaptation (LoRA)

LoRA is a widely adopted PEFT technique that hypothesizes that weight updates reside in a low-dimensional subspace [9]. Instead of updating the full weight matrix $W \in \mathbb{R}^{d \times k}$, LoRA represents the update as a product of two low-rank matrices:

$$W = W_0 + \Delta W = W_0 + BA \tag{2.3.1}$$

where W_0 is the frozen pre-trained weight matrix, and $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ are trainable matrices of rank $r \ll \min(d, k)$ [9]. During training, W_0 is kept frozen and receives no gradient updates, while

A and B are optimized. The number of trainable parameters is reduced from $d \times k$ to $r(d+k)$, which is orders of magnitude smaller for typical values of r (e.g., $r \in \{4, 8, 16\}$). In this thesis, LoRA is applied selectively to only the experts identified as trainable by the Activation Freeze Criterion described in Chapter 5.

2.4 Federated Learning (FL)

Federated Learning is a decentralized learning paradigm where training data remains on local clients [5]. Only model updates (gradients or weights) are shared with a central server to preserve data privacy [6, 5]. This makes FL particularly suitable for scenarios where data is sensitive, regulated, or simply too large to transmit to a central datacenter.

2.4.1 The Federated Averaging (FedAvg) Algorithm

The standard algorithm for FL is **FedAvg** [5]. In each communication round t, the server distributes the global model w_t to a set of clients. Each client k performs local Stochastic Gradient Descent (SGD) to obtain local weights w_{t+1}^k. The server then aggregates these updates as a weighted average:

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k \tag{2.4.1}$$

where n_k is the number of samples on client k and n is the total number of samples across all clients [5]. More robust aggregation algorithms have been developed since FedAvg, such as FedProx [8], which adds a proximal term to the local objective to limit client drift on non-IID data, and FedIT [6], which adapts federated training specifically for instruction tuning of language models. In this thesis, FedAvg is used as the aggregation baseline to keep the implementation focused on the expert selection and freezing mechanisms that are the core contribution of this work.

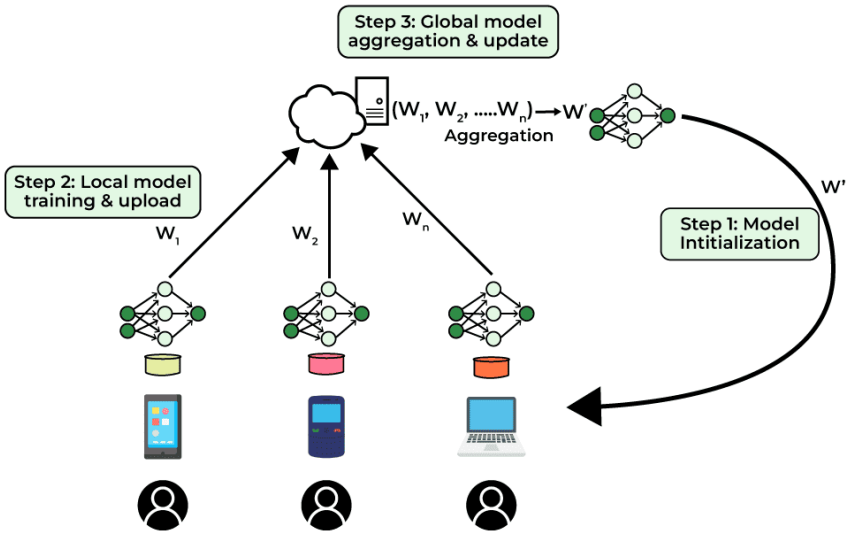


Figure 2.4.1: Basic Federated Learning workflow. Clients train locally on private data and upload only model updates to a central server, which aggregates them into a new global model before broadcasting it back. Image source: [16].

2.4.2 Challenges in Federated Learning

Despite its promise, Federated Learning faces several significant challenges that are directly relevant to the design of FedMoECap.

Non-IID Data Distributions. In practice, data across FL clients is rarely independently and identically distributed (IID). Each client's local dataset reflects its own usage patterns, domain, or user behavior, leading to statistical heterogeneity across the federation. When clients train on non-IID data, local model updates can diverge significantly from one another, causing the aggregated global model to converge more slowly or to a suboptimal solution [5, 8]. This is especially relevant in MoE models, where different data distributions may activate entirely different sets of experts on each client.

System Heterogeneity. FL clients vary widely in their hardware capabilities, including available VRAM, CPU speed, battery life, and network bandwidth. This system heterogeneity means that a single training configuration cannot satisfy all clients simultaneously. Stragglers — clients that are slow to complete a round — can bottleneck the entire training process [7]. For large MoE models, this challenge is severe: a client with limited VRAM may be unable to load or fine-tune the full set of expert parameters. FedMoECap directly addresses this by allowing each client to independently specify a freeze rate that limits its trainable parameter budget based on local hardware constraints.

Communication Overhead. Each FL communication round requires transmitting the model's trainable parameters between clients and the server. For large LLMs and MoE models, this communication cost is prohibitive, especially for clients on wireless or low-bandwidth connections [5, 7]. Reducing the number of parameters communicated per round is therefore a primary design goal of this thesis.

2.5 Model Quantization

Quantization is a model compression technique that reduces the numerical precision of a model's weights and activations, typically from 32-bit or 16-bit floating point (FP32 and FP16 respectively) to lower-precision formats such as 8-bit integers (INT8) or 4-bit integers (INT4) [17]. This reduces the memory footprint of the model and can accelerate inference on hardware that supports low-precision arithmetic.

In the context of large language models on resource-constrained edge devices, quantization is often a prerequisite for the model to fit within available VRAM at all. For example, the OLMoE-1B-7B model with 7 billion total parameters would require approximately 14 GB of VRAM in FP16, but can be reduced to under 4 GB with 4-bit quantization.

In this thesis, we employ **bitsandbytes** 8-bit quantization using the INT8 data type, introduced by Dettmers et al. [17]. Our codebase also supports 4-bit quantization using the NormalFloat 4 (NF4) datatype, introduced by Dettmers et al. [18] with QLoRA. NF4 is an information-theoretically optimal quantization format for normally distributed data, making it well-suited for the weight distributions of pre-trained LLMs. This allows us to load the base OLMoE-1B-7B model on the NVIDIA Jetson AGX Orin edge device, making fine-tuning feasible within the device's memory budget. Quantization is applied to the frozen base weights only; the LoRA adapter matrices are kept in full precision throughout training [18].

Chapter 3

Related Work

This chapter surveys the body of work most relevant to FedMoECap, organized into six thematic threads. We first examine existing approaches to federated fine-tuning of large language models, then discuss how parameter-efficient fine-tuning methods have been adapted for communication-constrained federated settings. We subsequently review expert-aware training and fine-tuning methods developed for Mixture-of-Experts architectures in centralized settings. We then discuss work on resource-aware and heterogeneous federated learning, followed by quantization techniques specific to MoE architectures. Finally, we survey approaches that directly combine Mixture-of-Experts with the federated learning paradigm. We conclude by identifying the gap that motivates this thesis.

3.1 Federated Fine-Tuning of Large Language Models

The problem of training neural networks in a federated setting was formalized by McMahan et al. [5], who introduced the **FedAvg** algorithm. FedAvg alternates between distributing a global model to a subset of clients, performing local stochastic gradient descent, and aggregating the resulting weight updates as a weighted average. The authors demonstrated that this approach can reduce communication rounds by one to two orders of magnitude compared to centralized synchronous SGD, making it the canonical baseline for federated learning research.

As federated learning was applied to increasingly heterogeneous settings, the limitations of FedAvg under non-IID data distributions became apparent. Li et al. [8] proposed **FedProx**, which augments the local training objective with a proximal term that penalizes deviations from the global model. This bounded client drift significantly improves convergence stability when local datasets are statistically heterogeneous, a condition that is the norm rather than the exception in practical deployments. FedProx also introduces γ-inexact solution so that clients with different hardware capabilities can minimize the local loss function in different degree, according to their computational capabilities.

More recently, the community has turned its attention to adapting federated learning specifically for the fine-tuning of large language models (LLMs). Zhang et al. [6] introduced **FedIT**, a framework that applies federated training to instruction tuning. FedIT demonstrated that aggregating instruction-following gradients from diverse client datasets, each reflecting different tasks and linguistic styles, can improve model quality beyond what is achievable with a single client's data. This line of work is directly relevant to our setting, as FedMoECap targets the same instruction-tuning regime but extends it to MoE architectures under resource constraints.

A critical limitation shared by these approaches is that they treat the model as a monolithic entity. When applied to MoE architectures, FedAvg, FedProx, and FedIT must transmit and aggregate the full set of expert parameters at every round, incurring communication costs that scale linearly with the number of experts. FedMoECap explicitly addresses this by selectively training and communicating only the most relevant subset of experts.

3.2 Parameter-Efficient Fine-Tuning in Federated Settings

The prohibitive cost of transmitting full model updates in federated learning has motivated the integration of parameter-efficient fine-tuning (PEFT) techniques into federated pipelines [2]. Among these, Low-Rank Adaptation (LoRA), introduced by Hu et al. [9], has become the dominant approach. LoRA freezes the pre-trained weight matrices and injects trainable low-rank decomposition matrices $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ into each layer. Since the rank r is typically much smaller than both d and k, the number of trainable parameters is reduced by several orders of magnitude. In federated settings, this directly translates into a proportional reduction of the communication payload per round.

Dettmers et al. extended LoRA with **QLoRA** [18], combining 4-bit NormalFloat quantization of the frozen base model weights with full-precision LoRA adapters. QLoRA demonstrated that a 65B parameter model can be fine-tuned on a single 48 GB GPU without sacrificing accuracy relative to full fine-tuning. This quantization strategy is directly adopted in FedMoECap, where the frozen base weights of OLMoE-1B-7B are quantized to 4-bit or 8-bit precision to enable edge devices with limited VRAM to participate in training. The preceding work on 8-bit matrix multiplication by Dettmers et al. [17] provides the underlying INT8 quantization kernel used in our pipeline.

An important problem when combining LoRA with federated learning is **system heterogeneity**: different clients may be able to afford different LoRA ranks based on their hardware constraints. Cho et al. [7] addressed this directly with **Heterogeneous LoRA (HetLoRA)**, which allows clients to independently select their LoRA rank and proposes a server-side aggregation scheme that reconciles adapters of different dimensions. Their work establishes a key precedent for the kind of per-client heterogeneity handling we implement in FedMoECap. However, HetLoRA is designed for dense foundation models and does not exploit the sparse expert structure of MoE architectures. FedMoECap provides a complementary and orthogonal axis of heterogeneity: rather than varying the rank of LoRA adapters, it varies the number of experts to which LoRA is applied, allowing each client to independently specify a freeze rate that constrains its total trainable parameter budget.

An alternative compression strategy is knowledge distillation [19], where a smaller student model is trained to emulate the behaviour of a larger teacher. Wu et al. [20] applied this idea to the federated setting by having the server produce a compressed LLM aligned to the full model, which clients then fine-tune via a lightweight adapter. While this approach substantially reduces the computational burden on clients, it requires the server to hold a representative dataset for the alignment step, an assumption that may be difficult to satisfy in privacy-sensitive deployments.

3.3 Expert-Aware Fine-Tuning of Mixture-of-Experts Models

The concept of selectively engaging a subset of model components during fine-tuning has a natural home in the MoE paradigm. The sparsely-gated MoE layer introduced by Shazeer et al. [10] already encodes a form of dynamic selection: at inference time, a learned gating network routes each token to only the top-k of the available experts. This routing signal is a rich source of information about which experts are functionally relevant to a given task or dataset.

Wang et al. [14] directly leveraged this signal for fine-tuning in their **Expert-Specialized Fine-Tuning (ESFT)** method. ESFT computes a relevance score for each expert based on its activation frequency or gate score on the fine-tuning dataset, then trains only the top-ranked experts so that the sum of the relevance score in each layer surpass a specific predefined threshold. This approach demonstrated that ESFT while training only 5%-15% of experts achieve comparable and sometimes better accuracy compared to full fine-tuning on many tasks, while significantly reducing the computational cost and number of trainable parameters. ESFT is the most directly related work to FedMoECap and serves as the key conceptual predecessor of our Activation Freeze Criterion.

However, ESFT was designed for centralized fine-tuning and has two properties that make it ill-suited for the federated setting. First, it applies a per-layer selection policy, meaning it independently selects a fixed fraction of experts within each MoE layer. This ignores the global distribution of expert

importance across layers, which can be highly non-uniform: some layers may have several highly activated experts, while others may have nearly all of their experts activated roughly equally. Second, ESFT requires centralized access to the full dataset to compute expert relevance scores, which is unavailable in the federated setting.

FedMoECap departs from ESFT on both counts. Our Activation Freeze Criterion operates globally across all layers, pooling all experts into a single ranked list and selecting the top $X\%$ regardless of which layer they belong to. This allows entire layers to be frozen when warranted and concentrates the training budget on the most task-relevant experts across the network as a whole. Furthermore, each client in FedMoECap computes its own selection based on its local data distribution, naturally adapting to the statistical heterogeneity inherent in federated settings. Crucially, this profiling step is performed only once at the very first communication round, on a small randomly sampled subset of the client's local dataset. Because FedMoECap keeps the router frozen throughout training, the routing behavior — and therefore the expert activation frequencies — does not change across rounds, making the selection computed at round one valid for the entire training run and keeping the profiling overhead negligible.

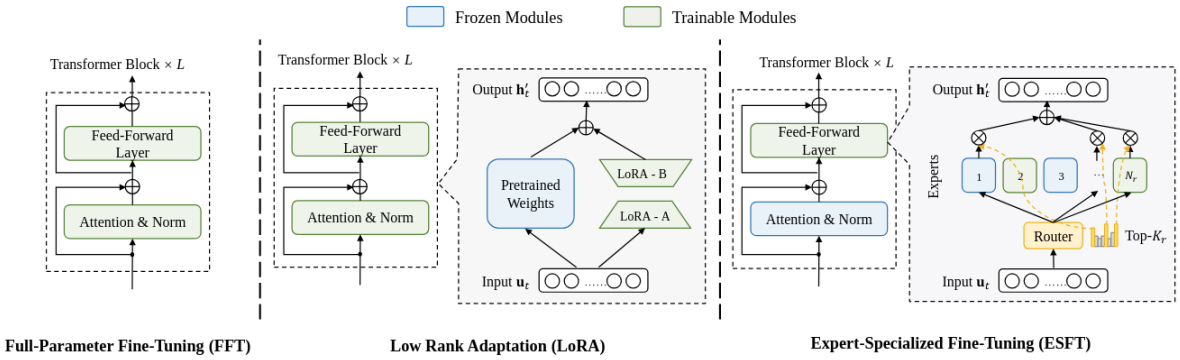


Figure 3.3.1: Comparison of fine-tuning paradigms. Full-Parameter Fine-Tuning (FFT) trains all weights, Low-Rank Adaptation (LoRA) injects trainable low-rank matrices into frozen layers, and Expert-Specialized Fine-Tuning (ESFT) selectively trains only the most relevant experts per layer. FedMoECap extends this line of work to the federated setting.

3.4 Resource-Aware and Heterogeneous Federated Learning

A defining challenge of practical federated learning deployments is the wide variation in client hardware capabilities. Early federated learning algorithms such as FedAvg assumed that all clients could hold and train the complete model, an assumption that becomes untenable as model sizes scale into the billions of parameters. This has motivated a body of work on resource-aware federated training, where different clients are permitted to participate with different computational and communication budgets.

The FedProx framework [8] remains a standard baseline for heterogeneous federated learning, as its proximal regularization effectively dampens the client drift that arises when clients train for different numbers of local steps or on datasets of varying statistical character. In FedMoECap, we build on this tradition by allowing each client to independently control its trainable expert set through a freeze rate parameter. Clients with limited VRAM or bandwidth can set a higher freeze rate, training fewer experts and transmitting a smaller update, while more capable clients can set a lower freeze rate to contribute a richer gradient signal. This design allows the federation to accommodate devices of widely varying capability within the same training round without requiring any client to be excluded. The NVIDIA Jetson AGX Orin edge device used in our experiments represents a concrete example of such a resource-constrained participant, operating alongside a much more powerful DGX server in the same federated training run.

3.5 Mixture of Experts and Quantization

Kim et al. [21] demonstrated that applying weight-only quantization to expert parameters in MoE architectures yields reliable model performance while substantially reducing memory consumption, without requiring any additional training. Notably, they showed that expert layers are significantly more robust to low-bit quantization than conventional feed-forward network layers, with 2-bit expert weights achieving better performance than a dense model trained on the same data while reducing model size by nearly 80%. This robustness of expert weights to quantization provides additional motivation for the quantization-friendly design of FedMoECap, where expert weights are frozen and only their compact LoRA adapters are trained and transmitted.

3.6 Mixture of Experts and Federated Learning

With the rapid spread of the MoE architecture many works try to utilize its unique architectural features in Federated Learning scenarios.

Tran et al. [22] instead of using different LoRA ranks in heterogeneous clients propose a resource-adaptive approach called A^3SMoE where different clients can activate different amount of experts in the topK routing mechanism of the gate in the Mixture of Experts architecture [10] according to their available resources. We instead do not try to change the top-K routing of the gates and leave clients to have as many trainable parameters as they desire according to their computational and accuracy needs.

Mei et al. [12] proposed **FedMoE**, a personalized federated learning framework that constructs a client-specific sub-MoE model for each participant. FedMoE operates in two stages: in the first stage, a preliminary round of fine-tuning collects expert activation probabilities on each client with sufficient memory to finetune the whole model, which are sent to the server; the server then performs a prediction of expert activation probabilities of clients with insufficient memory based on clients that have the same tasks as them and then performs a heuristic search to identify an optimal expert subset per client and distributes the corresponding submodels. In the second stage, each client fine-tunes only its assigned submodel and uploads the trained expert weights back to the server, which integrates them into the global model through a modular aggregation strategy that updates only the experts that were actively used. FedMoE demonstrated that routing-aware submodel construction reduces both memory footprint and communication volume while achieving competitive accuracy across heterogeneous tasks. However, its expert search applies a per-layer activation threshold, ignoring the global distribution of importance across the network. The two-stage initialization also introduces significant upfront overhead, requiring a dedicated preliminary fine-tuning phase before main training begins.

Fang et al. [13] proposed **HFedMoE**, the most closely related concurrent work to FedMoECap. HFedMoE also trains and transmits only a subset of experts per client, but introduces a more principled importance criterion grounded in the information bottleneck (IB) principle. For each training batch, it computes a per-expert importance score that combines a cumulative component, reflecting how consistently an expert is activated across all samples in the batch, with a specific component, capturing the peak routing score on individual samples to preserve experts that are critical for specific difficult inputs. A resource-aware selection module then re-schedules which experts participate in the backward pass to fit within each client's computing budget. After local training, each client uploads only the experts whose accumulated usage exceeds a threshold τ, and the server aggregates them selectively while applying importance-weighted averaging to the gating network parameters to mitigate routing divergence.

Compared to both FedMoE and HFedMoE, FedMoECap differs in three important ways. First, FedMoECap's profiling is performed only once at the first federated round, whereas HFedMoE re-identifies expert importance and refreshes its selection during every training round and for every mini-batch. Second, FedMoECap pools all experts into a single globally ranked list; in contrast, HFedMoE prioritizes layer-wise coverage by ensuring at least one expert is selected per layer before allocating the

remaining budget globally. Third, HFedMoE doesn't have a mechanism to manage experts that are close to convergence.

3.7 Summary and Positioning

Table 3.7.1 summarizes the landscape of related work and positions FedMoECap along the key dimensions of federated learning, PEFT, expert-awareness, and heterogeneity support.

Method	Federated	Expert-Aware	Lightweight Profiling	Heterogeneity	Conv. Mgmt.
FedAvg [5]	✓	×	×	×	×
FedProx [8]	✓	×	×	✓	×
FedIT [6]	✓	×	×	×	×
LoRA [9]	×	×	×	×	×
QLoRA [18]	×	×	×	×	×
HetLoRA [7]	✓	×	×	✓	×
ESFT [14]	×	✓	✓	×	×
A^3SMoE [22]	✓	✓	×	✓	×
FedMoE [12]	✓	✓	(✓)	✓	×
HFedMoE [13]	✓	✓	(✓)	✓	×
FedMoECap (Ours)	✓	✓	✓	✓	✓

Table 3.7.1: Comparison of related approaches along key design dimensions. ✓ indicates the property is fully satisfied; (✓) indicates profiling is performed but incurs non-negligible overhead (FedMoE requires a costly preliminary fine-tuning stage; HFedMoE re-profiles at every mini-batch); × indicates the property is not addressed.

The survey above reveals a clear and consistent gap in the literature. Federated fine-tuning methods such as FedAvg, FedProx, and FedIT treat MoE models as if they were dense networks, forfeiting the opportunity to exploit expert sparsity for communication and compute efficiency. Expert-aware methods such as ESFT assume centralized access to the full dataset and a single, uniform hardware environment. The most recent federated MoE approaches — FedMoE, HFedMoE, and A^3SMoE — represent important steps toward closing this gap, but each leaves key dimensions unaddressed: FedMoE applies a costly two-stage initialization and a per-layer selection criterion; HFedMoE introduces a more principled importance metric but recomputes it at every mini-batch; A^3SMoE adapts the number of activated experts per client but does not freeze experts based on a global importance criterion and does not manage convergence over time. FedMoECap is designed to occupy precisely this remaining intersection: a frozen-router, one-time global profiling identifies the most important experts, selective LoRA is applied only to those experts to maximize communication reduction, per-client freeze rates handle hardware heterogeneity, and three adaptive convergence management strategies (Static, Pruning, and Rolling) allow the framework to evolve intelligently over the course of federated training.

Chapter 4

Motivation

The preceding chapters have established the landscape of the problem. Chapter 2 introduced the technical foundations: the MoE architecture, the federated learning paradigm, and the family of parameter-efficient fine-tuning methods. Chapter 3 surveyed the state of the art and identified a consistent gap: existing methods either treat MoE models as monolithic dense networks, forfeiting their structural sparsity, or exploit expert routing in centralized settings that are incompatible with the privacy and resource constraints of federated deployments, or rely on expensive per-round profiling mechanisms for expert selection.

This chapter bridges that gap by presenting the motivating observations that shaped the design of FedMoECap. Rather than arriving at the framework through a top-down specification, we present the sequence of concrete observations — about expert routing behavior, hardware constraints, and the dynamics of federated training — that led us to each of the five design decisions described in Chapter 5.

4.1 Observation 1: Expert Routing Is Sparse, Skewed, and Task-Dependent

The foundational premise of FedMoECap is that not all experts matter equally for a given fine-tuning task. This claim is not merely intuitive; it is a structural consequence of how MoE models are trained. During pre-training, the load-balancing auxiliary loss encourages the router to distribute tokens roughly evenly across experts within each layer. However, once the model is applied to a specific downstream task, the routing distribution shifts: the subset of experts whose pre-trained representations are most aligned with the task domain receives a disproportionately large share of the token traffic.

This phenomenon has been documented empirically by Wang et al. [14], who showed that, on any given fine-tuning dataset, a small fraction of experts accounts for the vast majority of token assignments. The remaining experts are activated infrequently enough that their gradient signal during fine-tuning is negligible. Training them consumes compute and communication budget without contributing meaningfully to task adaptation.

Figure 4.1.1 makes this concrete. It shows the difference in per-expert routing probability between a Dolly-15K split and a Rosetta Code split on the same OLMoE-1B-7B model. The large-magnitude red and blue cells confirm that the two tasks activate fundamentally different expert subsets across nearly every layer: experts that are heavily used for natural language instruction following are largely dormant for code, and vice versa. Training all experts uniformly would waste the majority of the compute budget on parameters that carry no task-relevant gradient signal for either client.

The practical implication is direct: if we can identify the task-relevant experts before training begins, we can restrict gradient computation, VRAM allocation, and communication to that subset alone. This observation motivates the **Activation Freeze Criterion**: a one-time profiling pass over a small

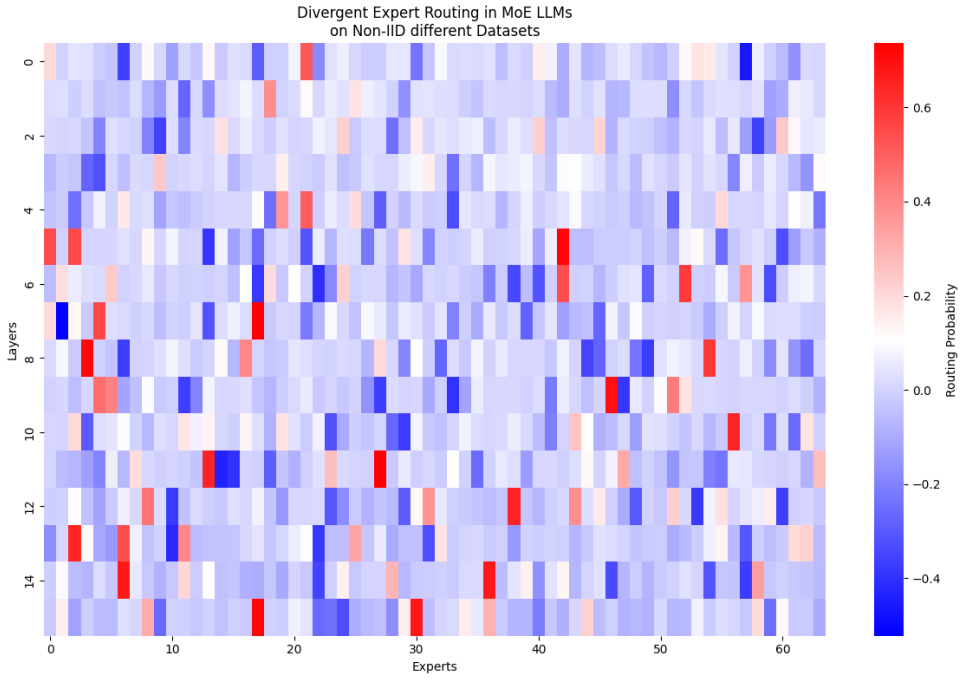


Figure 4.1.1: Difference in expert routing probability between a Dolly-15K split and a Rosetta Code split on OLMoE-1B-7B. Red cells indicate experts more active on Rosetta Code; blue cells indicate experts more active on Dolly15K. The large-magnitude divergence across layers confirms that expert routing is strongly task-dependent.

sample of the local dataset, measuring the routing frequency of each expert, followed by the selection of the top-ranked experts as the only trainable parameters.

4.2 Observation 2: Expert Importance Is a Global Property, Not a Per-Layer One

Having established that expert routing is skewed, a natural question is *how* to rank and select experts. ESFT [14] and FedMoE [12] adopt a per-layer approach. ESFT [14] for example, answers this with a per-layer activation threshold: within each layer, experts are ranked by their activation rate and the top experts are selected until their cumulative activation rate surpasses a fixed threshold (e.g., 0.3). This ensures that the selected experts in each layer account for at least a target share of the layer's total token traffic.

The problem with this approach is that different layers do not distribute their token traffic in the same way. In a layer where activation is concentrated — where one or two experts capture most of the traffic — the threshold is reached quickly, and only those few high-activation experts are selected. In a layer where activation is more dispersed across many experts, the cumulative rate grows slowly, so progressively more experts must be added to reach the same threshold, including experts with individually low activation rates.

This creates a cross-layer inconsistency. An expert in a dispersed layer may be selected despite having a low individual activation rate, simply because its layer had not yet reached the threshold. At the same time, an expert in a concentrated layer with a higher individual activation rate may remain frozen, because its layer surpassed the threshold before reaching it. The result is that the selected set is not aligned with the global routing signal: the trainable experts are not necessarily the most activated ones in the model, but rather the ones that happen to fall within their layer's threshold

budget. This is illustrated in Figures 5.2.2 and 5.2.3, and is discussed further in Section 5.2.

This observation motivates a **global expert ranking**: all $L \times N$ experts across all layers are pooled into a single list, ordered by activation rate, and the top fraction is selected regardless of which layer they belong to. Under this scheme, a layer with low overall activation may contribute no trainable experts, while a highly activated layer may contribute several. The training budget is concentrated where the routing statistics show it is needed most, rather than being shaped by how activation happens to be distributed within each layer.

4.3 Observation 3: The Pre-Trained Router Is a Reliable and Stable Signal

A key design choice that distinguishes FedMoECap from HFedMoE [13] is whether to recompute expert importance during training or to rely on a single measurement taken before training begins. HFedMoE recomputes its importance scores at every mini-batch, treating expert relevance as a dynamic quantity that must be continuously tracked.

This choice carries a significant overhead. Per-batch importance estimation adds computational cost to every local training step, and the additional communication required to synchronize routing statistics across clients consumes bandwidth that could otherwise carry model updates.

The alternative design, which FedMoECap adopts, rests on the observation that the pre-trained router is frozen throughout fine-tuning, like the strategy proposed by [14]. Because the gating network is not updated, the routing behavior it exhibits on a given dataset is determined entirely by its pre-trained weights and the input distribution. The activation statistics collected in round one are therefore a fixed property of the (model, dataset) pair, not a quantity that changes from round to round.

This means that a single profiling pass is sufficient. The activation rates measured before the first gradient update remain valid for the entire duration of training. Freezing the router also eliminates a second source of instability: non-IID routing divergence. When different clients update their routers on different local data distributions, the gating networks diverge, causing inconsistent expert assignments across the federation and destabilizing aggregation [12, 13]. By keeping the router frozen, FedMoECap sidesteps this problem entirely.

Figure 4.3.1 illustrates the severity of this divergence even within a single dataset. It shows the difference in routing probability between two non-IID splits of the Dolly-15K dataset assigned to different clients. Despite both clients working on the same underlying task, the skewed local distributions produce meaningfully different routing patterns across layers — a divergence that would grow larger if clients were allowed to update their routers independently. Freezing the router at its pre-trained state eliminates this source of instability entirely, ensuring that all clients operate under a consistent, shared routing function throughout training.

4.4 Observation 4: Hardware Heterogeneity Demands a First-Class Parameter

Federated learning is distinguished from distributed training by the nature of its participants. In datacenter distributed training, all workers are assumed to be homogeneous: they share the same hardware, the same memory capacity, and the same compute budget. In a federated setting, the participating devices may span several orders of magnitude in capability — from embedded edge modules such as the NVIDIA Jetson AGX Orin, to institution-grade servers with tens of gigabytes of GPU memory.

Standard federated fine-tuning protocols have no mechanism to accommodate this diversity. FedAvg sends the same model to every client and expects the same update in return. A client that cannot fit the required computation in its available VRAM must either be excluded from the round or force a configuration that ignores its constraints. Both outcomes are undesirable: the first reduces the effective

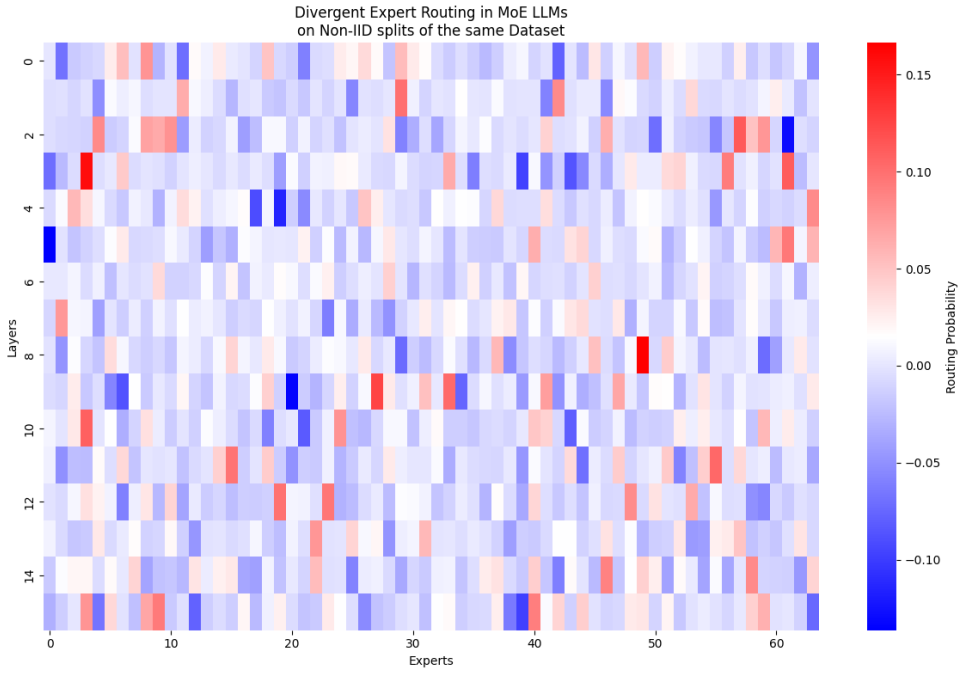


Figure 4.3.1: Difference in expert routing probability between two non-IID splits of the Dolly-15K dataset assigned to different federated clients, on OLMoE-1B-7B. Even within the same dataset, the skewed local distributions produce divergent routing patterns across layers, motivating the decision to freeze the router throughout training.

federation size and introduces selection bias; the second leads to out-of-memory failures or unstable training on under-resourced devices.

HetLoRA [7] addresses this for dense models by allowing clients to use different LoRA ranks. In the MoE setting, however, rank heterogeneity addresses only one dimension of the resource cost. The more significant cost driver is the number of experts loaded into VRAM for gradient computation. Even a low-rank adapter applied to a large number of experts can exceed the memory budget of an edge device.

This observation motivates the introduction of the **freeze rate** f_k as a per-client parameter. By controlling the fraction of experts each client freezes, FedMoECap directly controls the VRAM footprint, computation time, and communication payload independently per client. A device with limited resources sets a high freeze rate and trains few experts; a server-class device sets a low freeze rate and trains more. Both participate in the same round, contributing updates of different scope but aggregated consistently through the same FedAvg rule. Hardware heterogeneity is thus treated as a first-class design input rather than an operational edge case.

4.5 Observation 5: Expert Utility Diminishes During Training

The four observations above together justify the static configuration of FedMoECap-S: select the top experts once, apply LoRA to them, and train for all rounds. This is the simplest and most bandwidth-efficient operating mode, and as the experiments confirm, it achieves near-parity with full federated fine-tuning.

However, it leaves one source of inefficiency unaddressed. As training progresses, individual experts converge: the magnitude of their parameter updates shrinks monotonically as the adapter weights approach a local optimum on the federated objective. Moreover, Figure 4.5.1 shows that this convergence

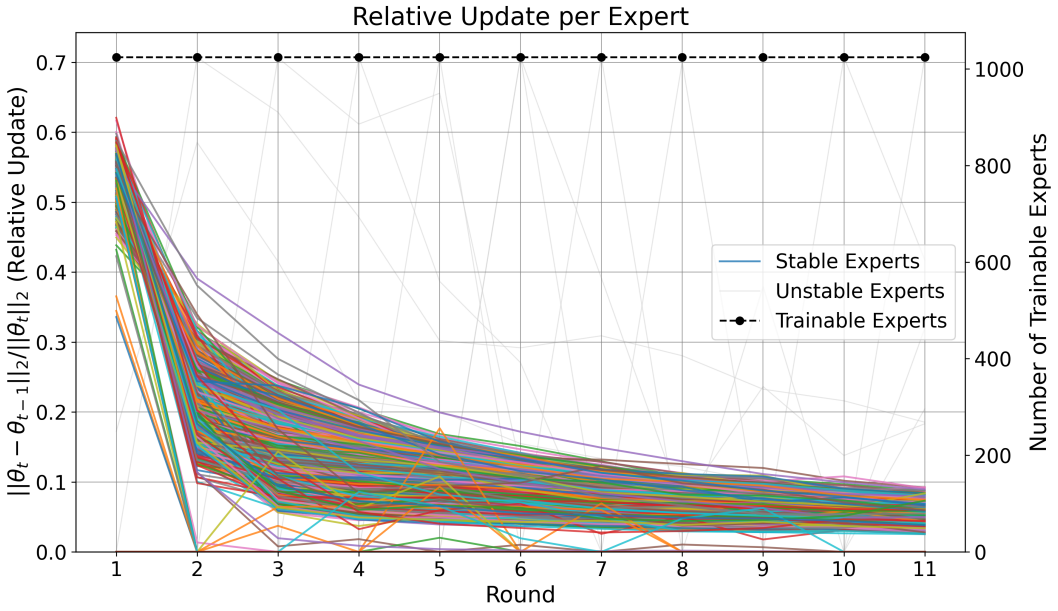


Figure 4.5.1: Relative parameter update magnitude per expert across federated rounds. Each colored line tracks one expert; The black dashed line (right axis) shows the number of still-trainable experts per round. Experts converge at different rates, motivating a dynamic pruning strategy that retires each expert individually as its updates fall below a threshold.

is not uniform: different experts cross the threshold at different rounds, leaving a progressively smaller active set as training proceeds. A converged expert continues to consume VRAM and communication bandwidth in every subsequent round, despite contributing negligible learning signal.

This observation motivates two additional strategies. **FedMoECap-P** monitors the per-expert update magnitude after each aggregation round and permanently freezes experts whose updates fall below a patience-gated threshold. The training workload thus shrinks over time, and the resource savings accumulate with each passing round. **FedMoECap-R** takes a different view of the same phenomenon: rather than discarding converged experts, it replaces them with previously frozen experts from the global ranking pool, allowing a larger total fraction of the expert population to receive some fine-tuning signal within a fixed per-round resource budget.

The two strategies represent distinct positions in the design space of convergence management, suited to different deployment objectives. Formalizing the convergence criterion and the mechanisms that act on it is a central contribution of the Methodology chapter.

4.6 Summary of Design Principles

The five observations above collectively define the design space that FedMoECap is built to occupy. We can distill them into four guiding principles that each correspond to a concrete component of the framework:

1. **Route before you train.** Expert routing statistics are a reliable, cheap, and stable proxy for task relevance. Measuring them once before any gradient update is sufficient, and doing so avoids per-round profiling overhead.

2. **Think globally, not layer by layer.** Task-relevant expertise is unevenly distributed across layers. A global expert ranking concentrates the training budget where it matters most, regardless of layer position.

3. **Make heterogeneity a parameter, not a problem.** Hardware diversity is an intrinsic property of real federated deployments. Encoding each client's resource constraint as a freeze rate allows all devices to participate without modification to the aggregation protocol.
4. **Stop training what has already learned.** Expert convergence is a signal, not noise. Detecting it and responding to it — either by pruning or rolling — converts a static resource budget into a dynamic one that improves as training progresses.

Chapter 5 translates these principles into the formal components of FedMoECap: the Activation Freeze Criterion, Selective LoRA, per-client freeze rates, the convergence criterion, and the three adaptive strategies built on top of it.

Chapter 5

Methodology

This chapter presents **FedMoECap**, the federated fine-tuning framework proposed in this thesis. We begin by formally stating the problem and the design objectives that motivated each component of the framework. We then describe the Activation Freeze Criterion used to identify which experts are worth training, followed by the application of Selective LoRA to those experts. We detail the mechanism for handling hardware heterogeneity across clients, then introduce the convergence criterion and the three adaptive strategies built on top of it. Finally, we describe the hybrid aggregation rule designed for the Rolling variant.

5.1 Problem Formulation and Design Objectives

Consider a federation of K clients, each holding a private local dataset $\mathcal{D}_k$ that cannot be shared with other participants. The clients collaborate to fine-tune a pre-trained Mixture-of-Experts language model $\mathcal{M}$ on their combined data, coordinated by a central server that performs aggregation but has no access to any raw data. The model $\mathcal{M}$ consists of L transformer blocks, each containing an MoE feed-forward sub-layer with N experts per layer, for a total of $L \times N$ experts across the network. A learned router at each layer routes each input token to the top-K_r most relevant experts, keeping the remaining $N - K_r$ experts idle for that token.

Standard federated fine-tuning protocols such as FedAvg require every client to hold and train the full set of model parameters. For a model with billions of expert parameters, this is impractical on resource-constrained edge devices. Furthermore, transmitting full model updates every round incurs prohibitive communication overhead. FedMoECap is designed around three primary objectives derived from these constraints:

1. **Efficiency:** Reduce the number of trainable parameters per round, thereby lowering VRAM consumption, computation time, energy usage, and communication bandwidth.

2. **Expert-Awareness:** Exploit the inherent sparsity of the MoE architecture by preferentially training the experts that are most relevant to the local task distribution on each client, rather than treating the model as a monolithic dense network.

3. **Heterogeneity:** Accommodate clients with widely varying hardware budgets within the same training round, allowing each device to independently control its trainable parameter budget without being excluded from participation.

FedMoECap addresses all three objectives simultaneously through the components described in the remainder of this chapter.

5.2 The Activation Freeze Criterion

The first step in each client's participation is to determine which experts are worth training given the client's local data and hardware budget. We propose a novel selection mechanism called the **Activation Freeze Criterion**.

5.2.1 Expert Activation Profiling

At the very first communication round, before any gradient updates are applied, each client performs a lightweight profiling pass over a small, randomly sampled subset of its local dataset $\mathcal{D}_k$. During this forward pass, the router at each MoE layer routes tokens to experts in the standard top-K_r fashion, and we record how often each expert is selected. Formally, let $c_{l,e}$ denote the total number of tokens routed to expert e in layer l across all profiling samples. The **activation rate** of expert (l, e) is then defined as:

$$p_{l,e} = \frac{c_{l,e}}{\sum_{e'=1}^{N} c_{l,e'}} \tag{5.2.1}$$

This profiling step is performed only once throughout the entire training run. Because FedMoECap keeps the router frozen throughout training, the routing behavior does not change between rounds, meaning the activation rates measured at round one remain valid for all subsequent rounds. This design choice is critical: it avoids the per-round or per-mini-batch re-profiling overhead incurred by methods such as HFedMoE [13], making the profiling cost negligible in relation to total training time.

A natural concern with one-time profiling is whether a small random subset of the local data produces a *consistent* expert selection — that is, whether a different random draw of the same size would yield a substantially different set of trainable experts. Figure 5.2.1 addresses this directly. For each of three datasets (Meta-MathQA, Rosetta Code, and Alpaca) and ten subset sizes ranging from 5 to 1000 samples, we drew three independent random subsets, ran the profiling pass on each, and computed the intersection ratio of the resulting expert selections. Two findings stand out. First, the selection is already highly stable at modest subset sizes: the average intersection ratio across all three datasets exceeds 0.85 at 50 samples and reaches above 0.93 at 100 samples, meaning that two independent profiling draws of 100 samples will select the same experts more than 93% of the time on average. Second, the wall-clock time of the profiling pass is virtually identical to that of ESFT across all subset sizes, confirming that the one-time cost of the Activation Freeze Criterion introduces no meaningful overhead relative to alternative selection methods.

A further advantage of the Activation Freeze Criterion over ESFT becomes apparent specifically in the federated setting. As shown in Figure 5.2.1, ESFT's expert selection is substantially more sensitive to the specific samples used for profiling: at 50 samples its average intersection ratio is approximately 0.61, compared to 0.87 for the Activation Freeze Criterion. A low intersection ratio means that two clients with similarly distributed data but different random subsets may select substantially different experts to train, introducing noise into the aggregation that is unrelated to any genuine difference in their local task distributions. The higher stability of the Activation Freeze Criterion ensures that expert selection across clients reflects real differences in data distribution rather than sampling artefacts — a property that is irrelevant in centralized training but becomes a concrete correctness concern in the federated setting.

5.2.2 Global Expert Ranking and Selection

Unlike ESFT [14], which applies a per-layer selection policy and independently selects a fraction of experts within each MoE layer, FedMoECap operates on a single **global expert pool**. All $L \times N$ experts from all layers are pooled together and ranked by their activation rate $p_{l,e}$ in descending order. Client k then selects the top $(1 - f_k) \times L \times N$ experts to be trainable, where $f_k \in [0, 1)$ is the client's **freeze rate**.

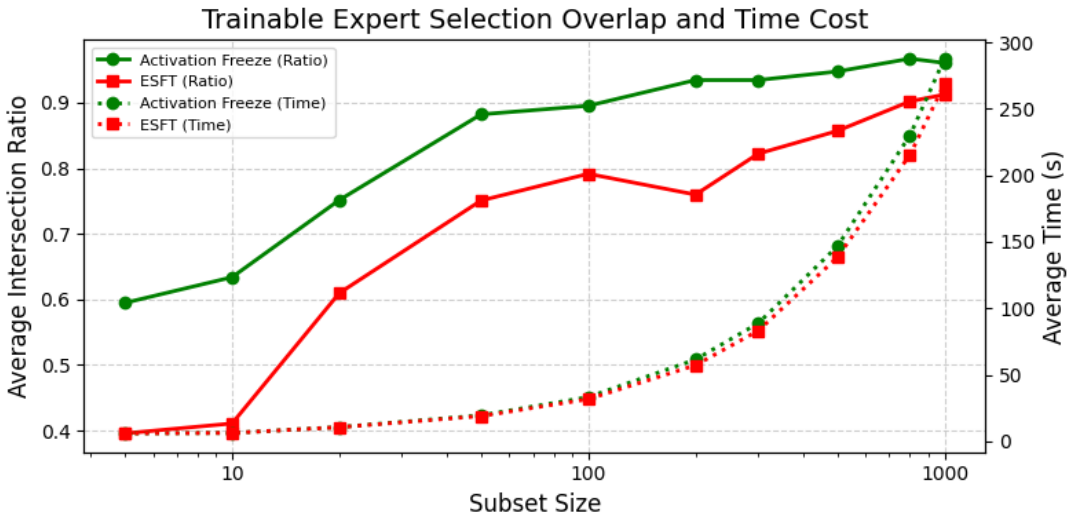


Figure 5.2.1: Stability and time cost of the Activation Freeze Criterion profiling pass as a function of subset size, averaged over Meta-MathQA, Rosetta Code, and Alpaca datasets. Solid lines (left axis) show the average intersection ratio of expert selections produced by three independent random subsets of each size: a ratio of 1.0 means all three draws selected identical experts. Dashed lines (right axis) show the average wall-clock time of the profiling pass. The Activation Freeze Criterion (green) achieves intersection ratios above 0.85 at just 50 samples and above 0.93 at 100 samples, while its time cost remains comparable to ESFT (red) at every subset size.

The structural impact of this global ranking is illustrated in Figures 5.2.2 and 5.2.3. In these heatmaps, each cell (e, l) represents an expert, where darker shading indicates a higher activation rate $p_{l,e}$ for experts selected as trainable. The histograms on the top and right margins provide a summary of selection density: the top histogram shows the number of selected experts per layer, while the right histogram shows how many layers each expert index is active in.

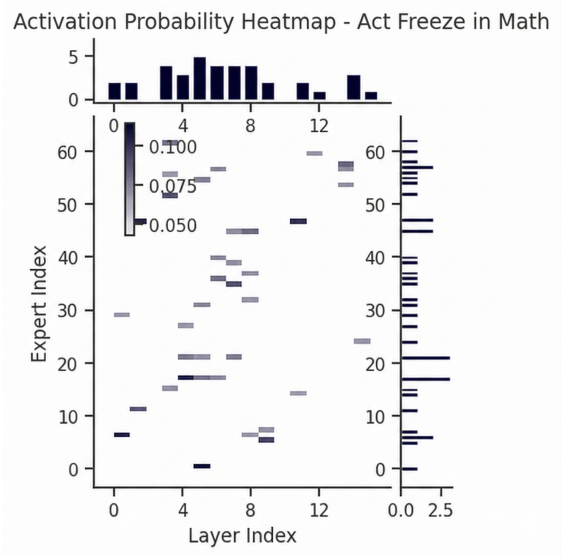
As seen in the comparison, the global pooling allows FedMoECap to concentrate the training budget on the most task-relevant experts regardless of their position, even allowing entire layers to be frozen if their experts are less active. In contrast, per-layer methods like ESFT are forced to distribute a fixed quota of trainable parameters to every layer, including those that may be irrelevant to the fine-tuning task.

5.2.3 Frozen Router

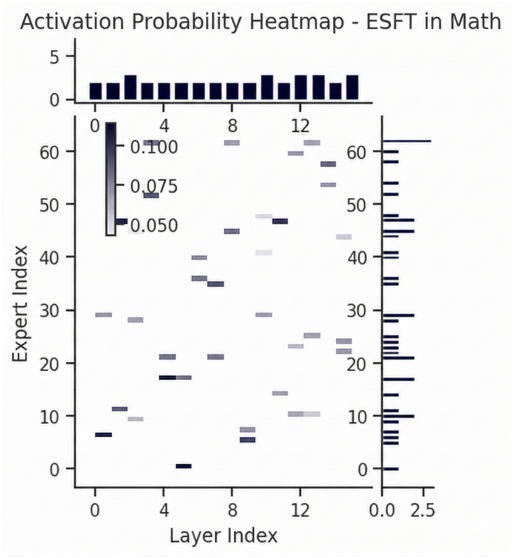
A fundamental design choice of FedMoECap, shared with ESFT [14], is to keep the MoE router frozen throughout the entire fine-tuning process. This decision is motivated by three considerations. First, freezing the router ensures that the routing behavior — and therefore the expert activation rates — remains constant, making the one-time profiling valid for all rounds. Second, the router is a relatively small component of the model, and its pre-trained routing decisions already encode a meaningful decomposition of the input space across experts; modifying it during fine-tuning can destabilize this structure. Third, freezing the router avoids the load-balancing issues that arise when routers are jointly optimized during distributed training, where different clients with different data distributions could push the router toward conflicting routing policies.

5.3 Selective LoRA for MoE Fine-Tuning

Having identified the set of trainable experts via the Activation Freeze Criterion, FedMoECap applies **Low-Rank Adaptation (LoRA)** [9] selectively to the weight matrices of those experts only. For



(a) Activation Freeze Criterion (Global)



(b) ESFT (Per-layer)

Figure 5.2.2: Routing probability heatmaps comparing the expert selection patterns of the Activation Freeze Criterion and ESFT, derived from mathematical routing probabilities. Selection density and distribution are discussed in Section 5.2.

each trainable expert e in layer l, we inject a low-rank adapter into its feed-forward weight matrices:

$$W_{l,e} = W^0_{l,e} + \Delta W_{l,e} = W^0_{l,e} + B_{l,e}A_{l,e} \tag{5.3.1}$$

where $W^0_{l,e} \in \mathbb{R}^{d\times k}$ is the frozen pre-trained weight matrix, and $B_{l,e} \in \mathbb{R}^{d\times r}$, $A_{l,e} \in \mathbb{R}^{r\times k}$ are the trainable low-rank matrices of rank $r \ll \min(d, k)$. The frozen experts receive no LoRA adapters and contribute no trainable parameters.

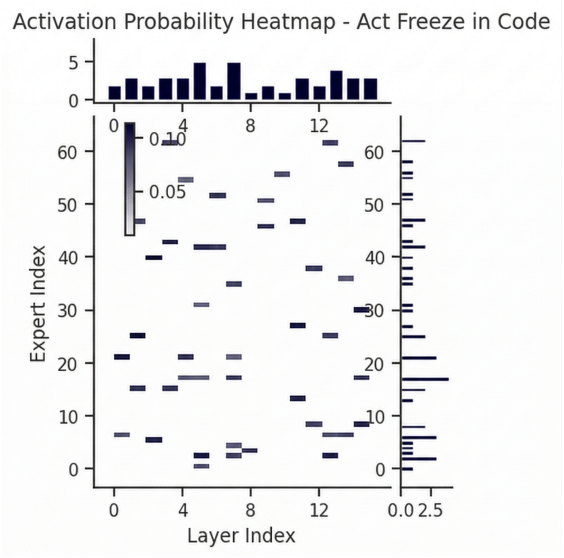
This selective application achieves a multiplicative reduction in the number of trainable parameters. In MoE models, expert weights constitute the vast majority of the total parameter count, so reductions in the number of trained experts translate almost directly into reductions in total trainable parameters. For example, in the OLMoE-1B-7B model approximately 93% of all parameters reside in expert weight matrices. Consequently, a freeze rate of $f_k = 0.75$ combined with LoRA rank $r = 8$ yields a 75.89% reduction in trainable parameters compared to applying LoRA to all experts — an almost one-to-one correspondence with the freeze rate.

During federated aggregation, only the LoRA adapter matrices $\{B_{l,e}, A_{l,e}\}$ of the trainable experts are transmitted to the server and averaged. The frozen base weights $W_{l,e}$ and the weights of frozen experts are never transmitted, substantially reducing per-round communication cost.

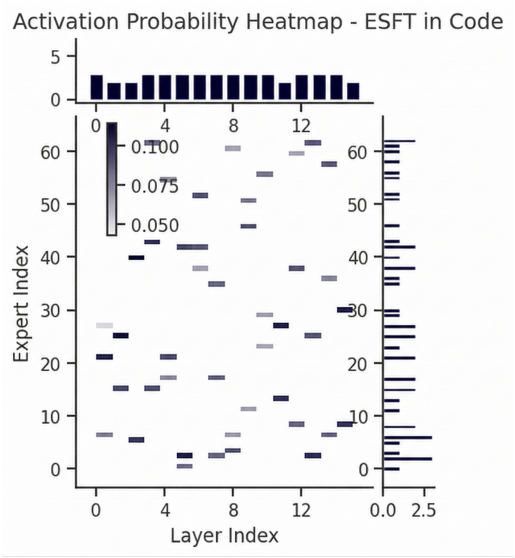
5.4 Per-Client Freeze Rate and Heterogeneity Handling

A central challenge in federated learning with large models is **system heterogeneity**: clients differ widely in the amount of VRAM, compute, and network bandwidth available to them. A mechanism that forces all clients to train the same number of experts will either exclude resource-constrained devices entirely, or bottleneck the entire round on the slowest participant.

FedMoECap addresses this through the **per-client freeze rate** f_k. Before training begins, each client independently sets $f_k \in [0, 1)$ based on its own hardware assessment. A client with limited VRAM sets a high freeze rate (e.g., $f_k = 0.95$), training only 5% of the total experts and transmitting a very small update. A more capable client sets a low freeze rate (e.g., $f_k = 0.5$), contributing updates from



(a) Activation Freeze (Implementation)



(b) ESFT (Implementation)

Figure 5.2.3: Heatmaps computed from actual token routing counts recorded during the profiling pass, confirming the practical reproduction of the mathematical selection patterns.

a larger fraction of experts. All clients participate in the same round simultaneously, each respecting its own budget, without requiring any client to be excluded or any synchronization overhead related to heterogeneity negotiation.

Because the Activation Freeze Criterion operates on the client's own locally computed activation rates, clients with different data distributions will naturally select different expert subsets, even if they share the same freeze rate. This per-client selection aligns the trainable expert set with both the hardware constraints and the statistical characteristics of each client's local data, providing a natural form of personalization within the federated framework.

5.5 Adaptive Convergence Management

The three variants of FedMoECap differ in how they respond to experts that have substantially converged during training. This section first defines the convergence criterion common to all variants, then describes the behavior of each variant in detail.

5.5.1 Convergence Criterion

An expert is monitored for convergence at the end of each federated round by measuring the **relative parameter update** of its LoRA adapter. At the coarsest level, this is defined as:

$$\delta_{l,e}^{(t)} = \frac{\left\| \theta_{l,e}^{(t)} - \theta_{l,e}^{(t-1)} \right\|_2}{\left\| \theta_{l,e}^{(t)} \right\|_2 + \varepsilon} \tag{5.5.1}$$

where $\theta_{l,e}^{(t)}$ denotes the concatenated LoRA parameters $[B_{l,e}, A_{l,e}]$ of expert (l, e) after round t, and $\varepsilon > 0$ is a small constant for numerical stability.

In practice, each expert contains multiple internal sub-layers (e.g., the gate, up, and down projections of a standard MoE feed-forward block), each of which carries its own pair of LoRA matrices. To account for this structure, we compute $\delta_{l,e}^{(t)}$ as the average relative update across all LoRA matrix pairs

within the expert:

$$\delta_{l,e}^{(t)} = \frac{1}{2\,|\mathcal{G}|} \sum_{g \in \mathcal{G}} \left(\delta_{l,e,g,A}^{(t)\,2} + \delta_{l,e,g,B}^{(t)\,2} \right) \tag{5.5.2}$$

where $\mathcal{G}$ is the set of internal sub-layers within the expert, $\delta_{l,e,g,A}^{(t)}$ and $\delta_{l,e,g,B}^{(t)}$ are the relative updates of the A and B LoRA matrices of sub-layer g, respectively, and $|\mathcal{G}|$ is the number of such sub-layers.

To avoid spurious convergence detections due to noisy single-round fluctuations, we employ a **patience mechanism**: an expert is declared converged only if $\delta_{l,e}^{(t)} < \tau$ for P consecutive rounds, where τ is the convergence threshold and P is the patience parameter. Both τ and P are global hyperparameters shared across all clients.

5.5.2 FedMoECap-S: Static Selection

The **Static** variant (**FedMoECap-S**) is the simplest of the three. Expert selection is performed once at the first communication round using the Activation Freeze Criterion, and the resulting trainable set is held fixed for the entire duration of training. Converged experts continue to receive and transmit LoRA updates every round; the convergence criterion plays no operational role in this variant.

FedMoECap-S is designed for scenarios where the client's communication and compute budget is fixed and known in advance and must be strictly respected throughout training. Its predictability makes it easy to deploy and reason about, and it provides a clean baseline for evaluating the contribution of the adaptive strategies. The main limitation of this variant is that resources continue to be spent on experts that have already converged, representing an inefficiency that the Pruning and Rolling variants address.

5.5.3 FedMoECap-P: Convergence-Based Pruning

The **Pruning** variant (**FedMoECap-P**) introduces adaptivity by progressively reducing the active expert set as training proceeds. After each aggregation round, the server evaluates the convergence criterion for every currently trainable expert. Any expert that satisfies the patience-gated threshold is permanently frozen: its LoRA adapters are fixed and excluded from all subsequent training rounds.

The effect is a monotonically shrinking trainable parameter set. As more experts converge, communication cost, VRAM usage, and training time all decrease with each round. By the final rounds, only the most volatile or task-relevant experts remain actively training, concentrating the remaining budget precisely where learning is still occurring.

FedMoECap-P is most suitable for scenarios where minimizing resource consumption during training or minimizing total training time and energy consumed is more important than maximizing the total number of experts fine-tuned or achieving the highest possible final accuracy. It does not attempt to train experts that were initially frozen by the Activation Freeze Criterion, focusing entirely on the efficient utilization of the initially selected set.

5.5.4 FedMoECap-R: Rolling Expert Swapping

The **Rolling** variant (**FedMoECap-R**) takes a different approach to convergence management: rather than permanently dropping converged experts, it *replaces* them with previously frozen experts from the Activation Freeze Criterion pool. When an expert (l, e) is declared converged, it is frozen and a new expert (l', e') — the next highest-ranked expert not yet trained — is activated by injecting fresh LoRA adapters initialized using Kaiming initialization [23].

This rolling mechanism allows FedMoECap-R to eventually fine-tune a larger fraction of the total expert population than the initial freeze rate would suggest, all while keeping the total number of simultaneously trainable experts — and therefore the per-round resource footprint — approximately constant. FedMoECap-R is most appropriate when the goal is to maximize the total number of experts

that receive some fine-tuning signal within a fixed resource budget, at the cost of increased complexity in the aggregation step.

Soft Convergence and Expert State Transitions

In FedMoECap-R, convergence is divided into two states to give the system additional flexibility. An expert that first crosses the patience threshold enters **Soft Convergence**: it stops receiving gradient updates on the clients where it has converged, but its global weights continue to be updated by clients that have not yet converged on it. Once the cumulative dataset fraction of clients that have declared the expert converged exceeds a hard convergence threshold, the expert enters **Hard Convergence** and is permanently retired from training, freeing its slot for a new untrained expert. Each client's contribution toward this threshold is weighted proportionally to its dataset size, $\frac{|\mathcal{D}_k|}{|\mathcal{D}_{\text{total}}|}$, so that clients with more data carry more influence over the retirement decision. This two-phase design prevents the premature retirement of experts that may still be undergoing meaningful adaptation on some clients due to the statistical heterogeneity of their local data distributions.

5.6 Hybrid Aggregation for the Rolling Variant

The Rolling variant introduces an aggregation challenge that is absent from the Static and Pruning variants. When an expert is in Soft Convergence, a subset of clients $\mathcal{S}_{\text{conv}}$ have frozen it locally and contribute no update, while the remaining clients $\mathcal{S}_{\text{active}}$ continue training it. If standard FedAvg aggregation is applied — averaging only the updates from active clients — the resulting global weights will drift away from the knowledge accumulated on clients in $\mathcal{S}_{\text{conv}}$, causing those clients to effectively *forget* what they learned. This is a form of catastrophic forgetting introduced at the aggregation step.

To prevent this, we introduce the **Hybrid Aggregation Rule**. For each expert e in Soft Convergence at round t, the updated global weights are computed as:

$$w_{t+1}^{e} = \underbrace{\sum_{i \in \mathcal{S}_{\text{conv}}} \frac{|\mathcal{D}_i|}{|\mathcal{D}_{\text{total}}|} w_t^e}_{\text{Preserve (Old Knowledge)}} + \underbrace{\sum_{j \in \mathcal{S}_{\text{active}}} \frac{|\mathcal{D}_j|}{|\mathcal{D}_{\text{total}}|} w_{t+1,j}^e}_{\text{Learn (New Knowledge)}} \tag{5.6.1}$$

where $|\mathcal{D}_{\text{total}}| = \sum_{k=1}^{K} |\mathcal{D}_k|$ is the total number of samples across all clients, w_t^e is the current global weight of expert e at round t, and $w_{t+1,j}^e$ is the locally updated weight on active client j.

The first term anchors the global weights to the converged state, weighted proportionally by the dataset sizes of converged clients. Intuitively, it acts as a data-weighted regularizer that reduces the effective learning rate of the expert in proportion to how much of the federation has already converged on it. The second term integrates new learning from active clients, also weighted by dataset size. When all clients are active (i.e., before any convergence occurs), Equation 5.6.1 reduces exactly to standard FedAvg, ensuring backward compatibility. When all clients have converged, the rule reduces to the identity, correctly preserving the expert's final weights without further modification. Once an expert reaches Hard Convergence, it is retired from training entirely and excluded from all subsequent aggregation steps.

5.7 The Full FedMoECap Training Protocol

Figure 5.7.1 provides a unified view of the complete FedMoECap training protocol, illustrating how the client-side and server-side components interact across a typical federated round. The diagram distinguishes the three variants — FedMoECap-S, FedMoECap-P, and FedMoECap-R — through colour-coded arrows, and highlights that Step 1 (expert profiling and selection) is executed only once at the very first round, after which routing statistics remain valid for the lifetime of training due to the frozen-router design.

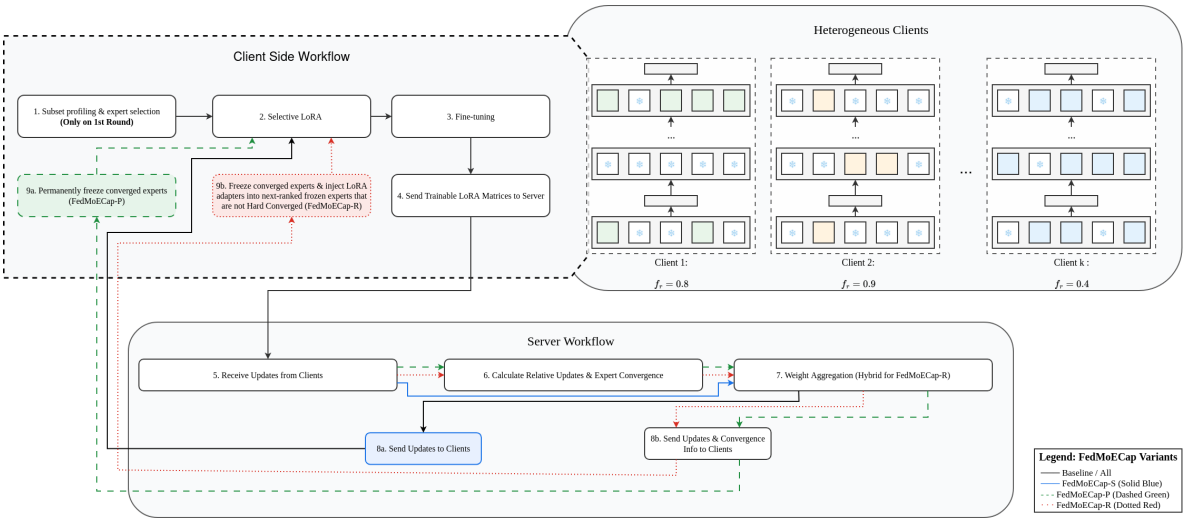


Figure 5.7.1: Overview of the FedMoECap training protocol. The **Client Side Workflow** (top-left) shows the per-round steps executed on each participating device, while the **Server Workflow** (bottom) shows the aggregation and convergence management steps. Step 1 (expert profiling and selection) is performed only once at the first communication round. Coloured arrows distinguish the three variants: black arrows denote steps shared by all variants, solid blue arrows denote the FedMoECap-S path (steps 7 → 8a → 2), dashed green arrows denote the FedMoECap-P path (steps 7 → 8b → 9a → 2), and dotted red arrows denote the FedMoECap-R path (steps 7 → 8b → 9b → 2). The heterogeneous client panel (top-right) illustrates that different clients may operate with different freeze rates f_r, reflected in the varying fraction of coloured (trainable) experts across client models.

At a high level, the protocol proceeds as follows. In round 1, each client performs a one-time profiling pass to compute expert activation rates, ranks all experts globally, applies its freeze rate f_k to select the trainable expert set, and injects LoRA adapters into those experts. In subsequent rounds, each client trains its trainable experts locally, transmits only the LoRA adapter deltas to the server, and the server aggregates them using either standard FedAvg (for FedMoECap-S and FedMoECap-P) or the Hybrid Aggregation Rule (for FedMoECap-R). After aggregation, the server evaluates the convergence criterion for each expert and, depending on the variant, either does nothing (Static), permanently freezes converged experts (Pruning), or initiates the rolling swap mechanism (Rolling).

5.8 Summary

FedMoECap combines five interacting components into a coherent framework for efficient federated fine-tuning of MoE models. The Activation Freeze Criterion provides a principled, one-time global expert ranking that is both task-aware and data-efficient. Selective LoRA minimizes the trainable parameter count on each client, directly reducing VRAM usage and communication cost. Per-client freeze rates translate hardware heterogeneity into a first-class parameter of the framework, allowing all devices to contribute to training within their individual constraints. The three adaptive strategies — Static, Pruning, and Rolling — offer practitioners a spectrum of trade-offs between simplicity, ongoing resource reduction, and total expert coverage. Finally, the Hybrid Aggregation Rule prevents catastrophic forgetting in the Rolling variant by explicitly preserving the contributions of clients that have reached convergence. Together, these components position FedMoECap as a framework designed from first principles for the unique demands of federated MoE fine-tuning.

Chapter 6

Experimental Evaluation

This chapter presents a comprehensive evaluation of the FedMoECap framework across multiple tasks, hardware platforms, and configuration variants. We begin by describing the experimental setup, including the model, datasets, hardware infrastructure, and default hyperparameters. We then report results on model accuracy, comparing the three FedMoECap variants against a **FedAvg-LoRA (All Experts)** baseline, which applies LoRA adapters to every expert in every round, and the pre-trained base model. Next, we analyse the system efficiency gains achieved by each variant on both edge and server hardware. Finally, we conduct a design and hyperparameter analysis to examine key design choices and to isolate the effect of three key hyperparameters: the expert freeze rate, the convergence threshold τ, and the patience P.

6.1 Experimental Setup

6.1.1 Model

All experiments use **OLMoE-1B-7B-0125** [11], a sparse Mixture-of-Experts language model developed by the Allen Institute for AI. The model has approximately 1 billion active parameters per forward pass while hosting a total of 7 billion parameters spread across its expert pool, making it an ideal testbed for evaluating the resource savings achievable through selective expert training. To enable deployment on memory-constrained edge devices, all experiments apply **INT8 quantization** to the model weights, halving the memory footprint relative to the default 16-bit precision.

6.1.2 Hardware Infrastructure

Experiments are conducted on two hardware tiers to assess FedMoECap across the full spectrum of realistic federated deployments.

The **edge tier** is represented by the **NVIDIA Jetson AGX Orin**, a compact system-on-module targeted at embedded AI workloads. The device features a unified memory architecture with limited VRAM headroom, making it a representative proxy for bandwidth- and energy-constrained federated clients such as mobile phones, industrial IoT nodes, or autonomous robots.

The **server tier** is represented by an **NVIDIA A100 (40 GB)** GPU hosted on an NVIDIA DGX system. This platform is used to simulate a powerful federated participant or an institution-grade client where training throughput and total training time are the primary concerns.

Reporting results on both tiers allows us to assess whether the efficiency gains of FedMoECap generalise across hardware generations.

6.1.3 Datasets and Evaluation

We evaluate model quality on four commonsense reasoning benchmarks commonly used in the LLM fine-tuning literature.

PIQA (Physical Intuition Question Answering) [24] is a binary-choice benchmark that tests a model's understanding of everyday physical commonsense, such as how objects interact with the physical world.

CommonsenseQA [25] is a five-way multiple-choice benchmark that requires models to reason about common concepts and their relationships, drawing on implicit world knowledge not explicitly stated in the question.

BoolQ [26] is a yes/no question-answering dataset constructed from naturally occurring questions paired with short passages. It tests the model's ability to draw binary inferences from context.

Winogrande [27] is a large-scale commonsense reasoning benchmark built as a harder and more diverse successor to the original Winograd Schema Challenge. Each example presents a sentence with a blank to be filled by one of two candidate words, requiring the model to resolve pronoun or referent ambiguity through implicit world knowledge.

Accuracy is the primary metric for all four benchmarks. To measure accuracy we use lm-eval [28] and its default configuration files for each of the aforementioned datasets.

To form non-IID client partitions, we embed every example in the training split using the **all-mpnet-base-v2** sentence encoder [29], which is built on top of MPNet [30]. The resulting embeddings are projected to a lower-dimensional space via Principal Component Analysis (PCA) and then clustered using k-means++ [31], a seeding-aware variant of k-means that provides stronger initialisation guarantees. The optimal number of clusters is determined via the Elbow Method. We then use a Dirichlet distribution with $\alpha = 0.3$ to split the clusters across clients, yielding skewed and heterogeneous local data distributions for PIQA, CommonsenseQA, and BoolQ. Winogrande is evaluated under an IID setup, where training examples are distributed uniformly across clients.

6.1.4 Federated Setup and Hyperparameters

We simulate a federated environment with multiple clients sharing a common communication round schedule of **12 to 15 rounds**. The default hyperparameters used across all experiments are summarised below.

The **freeze rate** is set to 0.90, meaning that 90% of experts (by activation frequency) are frozen per client per round, and the remaining 10% are selected for fine-tuning with LoRA adapters. The **convergence threshold** τ is set to 0.15, representing a relative parameter update of 15% below which an expert is considered to have converged. The **patience** is set to 2, meaning an expert must satisfy the convergence criterion for two consecutive rounds before it is acted upon by the Pruning or Rolling strategies. These defaults were determined through preliminary experiments and are varied individually in the design and hyperparameter analysis.

6.1.5 Evaluation Metrics

We evaluate system efficiency using four metrics, each measured and aggregated consistently across all experiments.

Allocated VRAM is recorded per client at the end of each training round using `torch.cuda.memory_allocated()`, which reports the amount of GPU memory actively occupied by tensors, excluding memory reserved by the PyTorch caching allocator but not yet in use. The reported value is the mean across all clients, averaged over all rounds.

Communication bandwidth is measured as the total number of bytes exchanged between the server and the clients over the entire training run. It is computed as the cumulative sum across all rounds of the bidirectional traffic, combining the data sent from the server to clients (model weights broadcast)

and the data sent from clients to the server (local model updates). All values are reported in megabytes (MB).

Time per batch is derived from the per-round training time logged for each client. Each client's round time is divided by its local number of mini-batches, normalising for differences in local dataset size across clients. The reported value is the mean across all clients, averaged over all rounds.

Energy consumption is measured exclusively on the NVIDIA Jetson AGX Orin using `tegrastats`, NVIDIA's built-in system monitoring utility for Jetson devices. `tegrastats` samples the total board power draw in milliwatts at regular intervals, covering all system components including the GPU, CPU, and memory subsystem. Total energy in watt-hours (Wh) is obtained by integrating the reported power over the full duration of the training run. This metric is not reported for the NVIDIA A100, as server-grade data centre hardware operates in a shared infrastructure environment where per-job board-level power isolation is not directly available. Instead, for the NVIDIA A100 we report the average **training time per client**, measured in seconds.

6.2 Model Performance

Table 6.2.1 reports the accuracy of each method on the PIQA, CommonsenseQA, BoolQ, and Winogrande benchmarks. We compare the three FedMoECap variants — Static (S), Pruning (P), and Rolling (R) — against two reference points: the **Base Model** (the frozen, pre-trained OLMoE-1B-7B without any fine-tuning) and **FedAvg-LoRA (All Experts)**, which applies LoRA adapters to every expert in every round.

Method	**PIQA (%)**	**CommQA (%)**	**BoolQ (%)**	**Winogrande (%)**
Base Model	78.73	52.33	69.97	68.82
FedAvg-LoRA (All Experts)	80.85	70.43	84.28	79.48
FedMoECap-S	80.00	67.24	82.11	76.16
FedMoECap-P	79.92	62.41	77.43	72.22
FedMoECap-R	–	65.60	80.24	74.66

Table 6.2.1: Accuracy comparison on PIQA, CommonsenseQA (CommQA), BoolQ, and Winogrande. The symbol '–' indicates that the experiment for that configuration was not yet completed at the time of writing.

Several observations can be drawn from these results. First, all FedMoECap variants substantially outperform the Base Model across every benchmark where results are available, confirming that the selective fine-tuning signal provided by the federated procedure is genuinely beneficial even when only 10% of experts are updated. On CommonsenseQA, for instance, FedMoECap-S improves over the Base Model by nearly 15 percentage points, from 52.33% to 67.24%. Similarly large gains are observed on BoolQ (69.97% to 82.11%) and Winogrande (68.82% to 76.16%), confirming that expert-selective fine-tuning generalises well across both binary and multi-choice reasoning formats.

Second, **FedMoECap-S** achieves the highest accuracy among the three variants and closes most of the gap to the FedAvg-LoRA (All Experts) baseline across all benchmarks. On PIQA, it reaches 80.00%, only 0.85 percentage points below the full FedAvg score of 80.85%, despite training only one tenth of the experts. On BoolQ and Winogrande the gaps are similarly small at 2.17 and 3.32 points respectively. The largest remaining gap appears on CommonsenseQA at 3.19 points, suggesting that this five-way reasoning task benefits more from broader expert coverage than the binary-choice benchmarks. Overall, these near-parity results demonstrate that the Activation Freeze Criterion is effective at identifying the small subset of experts most relevant to each task, so that training them alone is nearly as informative as training the entire expert pool.

Third, **FedMoECap-P** shows a more pronounced accuracy drop relative to FedMoECap-S, consistently across all four benchmarks. The gap is largest on CommonsenseQA (62.41% vs. 67.24%, a difference of 4.83 points) and smallest on PIQA (79.92% vs. 80.00%, just 0.08 points). On BoolQ

and Winogrande the gaps are 4.68 and 3.94 points respectively. This pattern reflects the progressive expert freezing inherent to the Pruning strategy: as experts converge and are permanently removed from training, the effective number of trainable experts decreases throughout the run, which can limit the model's ability to fully adapt to the task distribution. Nevertheless, the trade-off between this accuracy cost and the substantial system efficiency gains of FedMoECap-P is highly favourable, as the next section demonstrates.

Finally, **FedMoECap-R** consistently occupies an intermediate position between FedMoECap-S and FedMoECap-P across all benchmarks where results are available. On CommonsenseQA it scores 65.60%, on BoolQ 80.24%, and on Winogrande 74.66%, sitting 1.64, 1.87, and 1.50 points above FedMoECap-P respectively. This ordering reflects the Rolling strategy's ability to expose a larger fraction of the expert population to fine-tuning over the course of training by swapping converged experts for previously untrained ones, rather than letting converged experts remain permanently frozen as in FedMoECap-P.

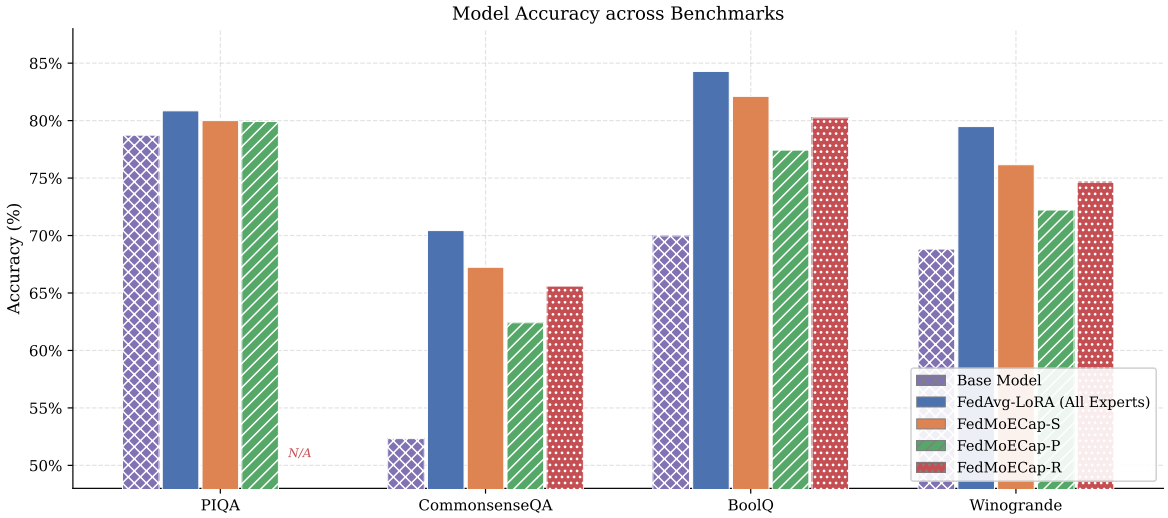


Figure 6.2.1: Accuracy of each method across all four benchmarks. FedMoECap-S closely approaches the FedAvg-LoRA (All Experts) ceiling, while FedMoECap-P and FedMoECap-R occupy progressively lower positions. The missing bar for FedMoECap-R on PIQA indicates a result not yet available at the time of writing.

6.2.1 Edge Deployment: NVIDIA Jetson AGX Orin

Table 6.2.2 reports resource consumption for the CommonsenseQA task on the Jetson AGX Orin. This setting is the most demanding from an efficiency standpoint, as the device operates under tight power and thermal budgets.

Method	VRAM (% drop)	Bandwidth (MB)	Time (s/batch)	Energy (Wh)
FedAvg-LoRA (All Experts)	-	37972	16.04	123.15
FedMoECap-S	−9.1%	3801 (−90.0%)	13.05 (−18.6%)	104.01 (−15.5%)
FedMoECap-P	−9.2%	1603 (−95.8%)	13.00 (−19.0%)	60.56 (−50.8%)
FedMoECap-R	−8.8%	3835 (−89.9%)	13.22 (−17.6%)	109.05 (−11.4%)

Table 6.2.2: System efficiency on the NVIDIA Jetson AGX Orin (CommonsenseQA task). Percentages in parentheses indicate change relative to FedAvg-LoRA (All Experts).

The results reveal three distinct sources of savings. The most dramatic is **communication bandwidth**: FedMoECap-S and FedMoECap-R reduce it by approximately 90%, while FedMoECap-P reduces it by over 95%, bringing the total bandwidth from roughly 38 GB down to under 2 GB.

This reduction directly reflects the fact that only the LoRA adapters of the active 10% of experts are communicated to the server, rather than the full set of expert weights. For federated deployments over cellular or satellite links, this bandwidth reduction can be the difference between a viable and an infeasible system.

The second notable saving is in **energy consumption**. FedMoECap-R saves approximately 11.4% of energy relative to the baseline, FedMoECap-S saves 15.5%, while FedMoECap-P achieves the largest reduction at 50.8%, cutting total energy roughly in half. For battery-powered or thermally constrained edge devices, this improvement enables significantly longer or more frequent local training sessions without overheating the mobile processor. The outsized saving of FedMoECap-P is explained by its progressive expert freezing: as rounds progress and more experts converge, the active training workload continuously shrinks, reducing power draw throughout the run rather than just at a fixed level. As a by-product, this progressive reduction in active experts may also shorten the number of rounds required to reach full convergence, since by the final rounds all remaining trainable experts will have eventually converged.

Finally, all three variants reduce **VRAM usage** by roughly 9% and **batch processing time** by approximately 17–19%. While these savings are more modest in relative terms, they are meaningful on a device where every megabyte of available memory and every second of compute translate directly into feasibility.

6.2.2 Server Deployment: NVIDIA A100

Table 6.2.3 reports resource consumption for BoolQ on the NVIDIA A100. On this platform, VRAM is not a binding constraint, so the relevant metrics are bandwidth, per-batch latency, and total training time per client.

Method	VRAM (%)	Bandwidth (MB)	Time (s/batch)	Train Time/Client (s)
FedAvg-LoRA (All Experts)	-	37963	8.69	2923.42
FedMoECap-S	−7.4%	3853 (−89.8%)	7.24 (−16.7%)	2422.52 (−17.1%)
FedMoECap-P	−13.5%	1789 (−95.3%)	6.71 (−22.8%)	1528.05 (−47.2%)
FedMoECap-R	−7.3%	3917 (−89.7%)	7.23 (−16.8%)	2422.24 (−17.1%)

Table 6.2.3: System efficiency on the NVIDIA A100 (BoolQ task). Percentages in parentheses indicate change relative to FedAvg-LoRA (All Experts). Training time is averaged across clients.

The A100 results confirm and extend the findings from the Orin. The 90–95% bandwidth reduction is entirely preserved on server hardware, demonstrating that this saving is a structural property of the algorithm rather than an artefact of a specific hardware constraint.

The most notable finding on the A100 is the **training time reduction** achieved by FedMoECap-P: its average training time per client drops by 47.2%, from 2923.42 seconds to just 1528.05 seconds. This speedup stems from the Pruning strategy's progressive expert freezing mechanism: as rounds progress, fewer experts require gradient computation and parameter updates, continuously shrinking the effective workload. For the same reason, FedMoECap-P may also require fewer total communication rounds to reach convergence, since the remaining trainable experts will have converged well before the scheduled end of training. FedMoECap-S and FedMoECap-R both achieve a training time reduction of 17.1%. For FedMoECap-S this is attributable purely to the smaller fixed set of trainable parameters maintained throughout training, while for FedMoECap-R the Rolling mechanism keeps the number of simultaneously active experts approximately constant by swapping in new experts as others converge, which explains why its training time closely mirrors that of the Static variant.

All three variants also deliver a 16–23% improvement in per-batch latency relative to the baseline, consistent with the Orin results. Regarding **VRAM**, FedMoECap-P achieves the largest reduction at 13.5%, while FedMoECap-S and FedMoECap-R deliver comparable savings of 7.4% and 7.3% respectively. The larger VRAM saving of FedMoECap-P is again explained by its progressive freezing: as experts are pruned over time, fewer adapter weights are held in GPU memory, whereas FedMoECap-S

and FedMoECap-R maintain a fixed active parameter set throughout. Taken together, these results establish FedMoECap as a system-level improvement that is robust across the hardware diversity characteristic of real federated deployments.

6.3 Design and Hyperparameter Analysis

We conduct a design and hyperparameter analysis to examine the impact of key design choices and to understand the sensitivity of FedMoECap to three key hyperparameters: the **freeze rate**, the **convergence threshold** τ, and the **patience** P. Each analysis is evaluated on two platforms simultaneously: the **NVIDIA Jetson AGX Orin** running **CommonsenseQA** and the **NVIDIA A100** running **BoolQ**. Since the two setups differ in both hardware and task, results are reported separately for each platform and are not directly compared. FedMoECap-S is used for the freeze rate study, while FedMoECap-P and FedMoECap-R are used for the threshold and patience studies. All other hyperparameters are held at their default values throughout.

6.3.1 Impact of the Freeze Rate

The freeze rate is the most fundamental hyperparameter in FedMoECap, as it directly controls the fraction of experts that are frozen per client and therefore sets the upper bound on all resource savings. Tables 6.3.1 and 6.3.2 explore three values around the default of 0.90 on the Jetson and A100 respectively.

Freeze Rate	VRAM (%)	Bandwidth (MB)	Time (s/batch)	Energy (Wh)	Acc (%)
0.80 (More Training)	+1.2%	7645 (+101.1%)	13.33 (+2.1%)	106.88 (+2.8%)	69.37
0.90 (Default)	–	3800.97	13.05	104.01	67.24
0.95 (High Freeze)	−0.6%	1913.49 (−49.7%)	12.74 (−2.4%)	102.53 (−1.4%)	66.34

Table 6.3.1: Sensitivity to the expert freeze rate on FedMoECap-S (CommonsenseQA, NVIDIA Jetson AGX Orin). Percentages are relative to the default freeze rate of 0.90.

Freeze Rate	VRAM (%)	Bandwidth (MB)	Time (s/batch)	Train Time/Client (s)	Acc (%)
0.80 (More Training)	+1.0%	7761.78 (+101.4%)	7.53 (+3.9%)	2517.93 (+3.9%)	83.24
0.90 (Default)	–	3853.25	7.24	2422.52	82.11
0.95 (High Freeze)	−0.5%	1928.59 (−49.9%)	7.44 (+2.7%)	2478.02 (+2.3%)	81.90

Table 6.3.2: Sensitivity to the expert freeze rate on FedMoECap-S (BoolQ, NVIDIA A100). Percentages are relative to the default freeze rate of 0.90.

The results reveal a consistent pattern across both platforms: **bandwidth** is by far the most sensitive metric to changes in the freeze rate, while all other metrics remain relatively stable. Reducing the freeze rate from 0.90 to 0.80 — thus doubling the number of trainable experts — more than doubles the communication overhead on both platforms (+101.1% on the Jetson, +101.4% on the A100), while the accuracy gain is modest: +2.13 points on CommonsenseQA and +1.13 points on BoolQ. The energy cost on the Jetson increases by only 2.8%, suggesting that the additional expert updates contribute relatively little to the overall power draw compared to the fixed system overhead.

Increasing the freeze rate to 0.95 halves the bandwidth on both platforms (−49.7% and −49.9% respectively), but the effect on accuracy differs. On CommonsenseQA the drop is 0.90 points, while on BoolQ it is a negligible 0.21 points, suggesting that BoolQ is a less expert-sensitive task and tolerates more aggressive freezing with minimal accuracy cost. Notably, the higher freeze rate does not reduce training time on either platform, and on the A100 it increases slightly. This is likely a statistical artefact: at fr=0.90, the number of trainable parameters is already so small relative to the full model that a further halving of the active expert set produces negligible change in per-batch compute time.

These findings confirm that 0.90 is a well-calibrated default. Reducing the freeze rate below 0.90 more than doubles the bandwidth for only marginal accuracy gains, while increasing it to 0.95 halves

the bandwidth at a modest and task-dependent accuracy cost, making it a viable option for highly bandwidth-constrained deployments.

6.3.2 Impact of the Convergence Threshold on FedMoECap-P

The convergence threshold τ governs how aggressively the Pruning variant freezes experts during training. A lower τ imposes a stricter convergence criterion, requiring experts to demonstrate smaller relative parameter updates before they are frozen, thus allowing them to train for longer. Tables 6.3.3 and 6.3.4 report results for three values of τ on the Jetson and A100 respectively.

Threshold (τ)	**VRAM (%)**	**Bandwidth (MB)**	**Time (s/batch)**	**Energy (Wh)**	**Acc (%)**
0.10 (Stricter)	−2.9%	2028.19 (+26.6%)	12.66 (−2.6%)	95.84 (+58.3%)	65.03
0.15 (Default)	–	1602.67	13.00	60.56	62.41
0.20 (Looser)	+0.1%	1586.82 (−1.0%)	10.78 (−17.1%)	50.15 (−17.2%)	63.64

Table 6.3.3: Sensitivity to the convergence threshold τ on FedMoECap-P (CommonsenseQA, NVIDIA Jetson AGX Orin). Percentages are relative to the default threshold of 0.15.

Threshold (τ)	**VRAM (%)**	**Bandwidth (MB)**	**Time (s/batch)**	**Train Time/Client (s)**	**Acc (%)**
0.10 (Stricter)	−0.9%	2389.94 (+33.6%)	6.70 (−0.2%)	2048.96 (+34.1%)	80.43
0.15 (Default)	–	1789.42	6.71	1528.05	77.43
0.20 (Looser)	+7.1%	1610.00 (−10.0%)	6.01 (−10.5%)	1005.85 (−34.2%)	75.41

Table 6.3.4: Sensitivity to the convergence threshold τ on FedMoECap-P (BoolQ, NVIDIA A100). Percentages are relative to the default threshold of 0.15.

The results reveal a clear and consistent accuracy trade-off across both platforms. Tightening τ to 0.10 forces experts to satisfy a stricter convergence criterion before being frozen, meaning they receive more gradient updates overall. On the Jetson this yields a +2.62 point accuracy gain on CommonsenseQA, and on the A100 a +3.00 point gain on BoolQ, at the cost of substantially increased bandwidth (+26.6% and +33.6% respectively) and resource consumption (+58.3% energy on the Jetson, +34.1% total training time on the A100).

Relaxing τ to 0.20 produces the opposite effect on the A100: training time per client drops by 34.2% and bandwidth by 10.0%, but accuracy falls by 2.02 points on BoolQ. On the Jetson, the looser threshold delivers a 17.1% reduction in batch time and a 17.2% energy saving, while accuracy actually improves slightly by +1.23 points. This apparent discrepancy between platforms reflects the different nature of the two tasks: BoolQ's binary yes/no format is more sensitive to premature expert freezing, whereas CommonsenseQA's five-way structure allows more redundancy in the experts that contribute to the correct answer.

These results demonstrate that τ gives practitioners a meaningful lever to tune the accuracy–efficiency trade-off in FedMoECap-P. For deployments where accuracy is paramount, $\tau = 0.10$ is preferable. For highly resource-constrained edge scenarios where energy and latency are the binding constraints, $\tau = 0.20$ is a strong alternative, provided the target task is not overly sensitive to expert coverage.

6.3.3 Impact of the Convergence Threshold on FedMoECap-R

We repeat the threshold sensitivity analysis for FedMoECap-R, whose expert replacement mechanism responds to convergence signals differently from the Pruning strategy. Tables 6.3.5 and 6.3.6 summarise the results for the Jetson and A100 respectively.

In sharp contrast to FedMoECap-P, the Rolling variant exhibits near-zero sensitivity to the convergence threshold across all system metrics on both platforms. VRAM, bandwidth, batch time, training time, and energy are essentially identical across all three values of τ. This stability arises because the Rolling strategy maintains a constant number of simultaneously active experts regardless of how quickly individual experts converge: as soon as one expert is frozen, another is swapped in to take

Threshold (τ)	VRAM (%)	Bandwidth (MB)	Time (s/batch)	Energy (Wh)	Acc (%)
0.10 (Stricter)	−0.0%	3838.69 (+0.1%)	13.15 (−0.5%)	106.06 (−2.7%)	66.01
0.15 (Default)	–	3835.32	13.22	109.05	65.60
0.20 (Looser)	−0.0%	3839.00 (+0.1%)	13.22 (−0.0%)	108.60 (−0.4%)	65.19

Table 6.3.5: Sensitivity to the convergence threshold τ on FedMoECap-R (CommonsenseQA, NVIDIA Jetson AGX Orin). Percentages are relative to the default threshold of 0.15.

Threshold (τ)	VRAM (%)	Bandwidth (MB)	Time (s/batch)	Train Time/Client (s)	Acc (%)
0.10 (Stricter)	−0.1%	3892.10 (−0.6%)	7.40 (+2.3%)	2480.05 (+2.4%)	80.76
0.15 (Default)	–	3917.44	7.23	2422.24	80.24
0.20 (Looser)	+0.1%	3918.66 (+0.0%)	7.34 (+1.5%)	2454.85 (+1.3%)	79.17

Table 6.3.6: Sensitivity to the convergence threshold τ on FedMoECap-R (BoolQ, NVIDIA A100). Percentages are relative to the default threshold of 0.15.

its place. The total workload per round therefore remains approximately constant irrespective of how aggressively convergence is declared.

The only meaningful difference across thresholds appears in **accuracy**. On both platforms, the stricter threshold $\tau = 0.10$ yields the best accuracy (66.01% on CommonsenseQA, 80.76% on BoolQ), while the looser $\tau = 0.20$ produces the lowest (65.19% and 79.17% respectively). The accuracy spread across thresholds is 0.82 points on CommonsenseQA and 1.59 points on BoolQ, suggesting that allowing experts to train slightly longer before rotating them out leads to marginally better specialisation, but the effect is small in absolute terms.

Overall, the threshold sensitivity analysis reveals a clear asymmetry between the two dynamic strategies. FedMoECap-P's efficiency profile is highly sensitive to τ, offering practitioners a rich trade-off space between accuracy and resource consumption. FedMoECap-R, by contrast, is essentially threshold-agnostic in its system behaviour, making it a more predictable and robust choice when consistent resource usage across rounds is required.

6.3.4 Impact of the Patience on FedMoECap-P

The patience P controls how many consecutive rounds an expert must satisfy the convergence criterion before it is frozen. A higher patience therefore requires more sustained evidence of convergence before an expert is removed from training, effectively allowing experts to train for longer before being pruned. The convergence threshold is fixed at $\tau = 0.15$ for all experiments in this subsection. Tables 6.3.7 and 6.3.8 report results for patience values of 2, 3, and 4 on the Jetson and A100 respectively.

Patience (P)	VRAM (%)	Bandwidth (MB)	Time (s/batch)	Energy (Wh)	Acc (%)
2 (Default)	–	1602.67	13.00	60.56	62.41
3	−3.8%	1920.51 (+19.8%)	12.47 (−4.1%)	70.68 (+16.7%)	63.64
4	−2.4%	2237.80 (+39.6%)	12.64 (−2.8%)	73.99 (+22.2%)	64.29

Table 6.3.7: Sensitivity to the patience P on FedMoECap-P (CommonsenseQA, NVIDIA Jetson AGX Orin). Percentages are relative to the default patience of 2.

As patience increases, accuracy improves consistently on both platforms: from 62.41% to 64.29% on CommonsenseQA (+1.88 points) and from 77.43% to 79.88% on BoolQ (+2.45 points). This is expected — a stricter patience requirement forces experts to demonstrate sustained convergence before being frozen, ensuring they have been more thoroughly trained before their slot is released. The cost is a corresponding increase in communication bandwidth: roughly +20% per additional patience step on the Jetson (+19.8% at $P = 3$, +39.6% at $P = 4$) and roughly +17% per step on the A100 (+17.3% at $P = 3$, +34.6% at $P = 4$), as active experts transmit more rounds of LoRA updates before being frozen. Energy consumption on the Jetson also increases correspondingly, by +16.7% at $P = 3$ and

Patience (P)	VRAM (%)	Bandwidth (MB)	Time (s/batch)	Train Time/Client (s)	Acc (%)
2 (Default)	–	1789.42	6.71	1528.05	77.43
3	−19.6%	2099.01 (+17.3%)	5.38 (−19.9%)	1606.77 (+5.2%)	78.75
4	−17.2%	2407.71 (+34.6%)	5.53 (−17.6%)	1815.26 (+18.8%)	79.88

Table 6.3.8: Sensitivity to the patience P on FedMoECap-P (BoolQ, NVIDIA A100). Percentages are relative to the default patience of 2.

+22.2% at $P = 4$.

The apparent reductions in **VRAM** and **Time (s/batch)** with higher patience are a statistical artefact and should *not* be interpreted as genuine savings. As patience increases, experts take longer to be pruned, which means the model goes through more late rounds in which very few experts remain active. These low-activity rounds drag down the per-round averages for VRAM and batch time, even though the peak resource usage during the active training phase is unchanged. This effect is especially pronounced on the A100, where the apparent VRAM reduction reaches −19.6% at $P = 3$ and −17.2% at $P = 4$, far larger than the modest −3.8% and −2.4% observed on the Jetson. Figure 6.3.1 confirms this interpretation: the trainable expert count per round exhibits essentially the same profile across patience values, shifted later in time, rather than a fundamentally different trajectory.

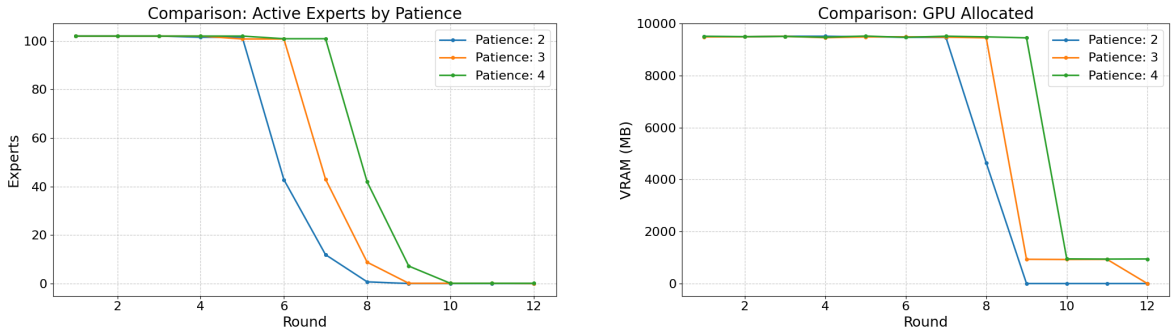


Figure 6.3.1: Left: trainable experts per round for different patience values on FedMoECap-P. Right: allocated VRAM per round. The expert curve is shifted later with higher patience, confirming that the apparent VRAM reduction is an averaging artefact.

It is also informative to compare FedMoECap-P with patience $P = 4$ against FedMoECap-S, since $P = 4$ achieved the highest accuracy among all FedMoECap-P patience configurations. Table 6.3.9 reports this comparison, with FedMoECap-S as the reference.

Metric	BoolQ (A100)	CommonsenseQA (Jetson)
Bandwidth (MB)	−37.5%	−41.1%
Train Time/Client (s)	−25.1%	−35.0%
Energy (Wh)	–	−28.9%
Acc (%)	−2.23	−2.95

Table 6.3.9: FedMoECap-P with patience $P = 4$ vs FedMoECap-S as reference, both with an initial freeze rate of 0.90. Values indicate the change relative to FedMoECap-S (BoolQ on NVIDIA A100 and CommonsenseQA on NVIDIA Jetson AGX Orin).

Even with the highest patience tested, FedMoECap-P reduces bandwidth by 37–41% and total training time by 25–35% compared to FedMoECap-S, at an accuracy cost of approximately 2–3 points. This confirms that FedMoECap-P with increased patience occupies a distinct and useful point on the efficiency–accuracy curve: substantially more efficient than the Static variant while remaining considerably more accurate than FedMoECap-P at the default patience.

6.3.5 Impact of the Patience on FedMoECap-R

Similarly to FedMoECap-P, patience in FedMoECap-R controls how many consecutive rounds an expert's relative parameter update must remain below the threshold τ before the expert is declared converged and swapped out for a previously untrained one. All experiments in this subsection use $\tau = 0.10$. Tables 6.3.10 and 6.3.11 report results for patience values of 2, 3, and 4 on the Jetson and A100 respectively.

Patience (P)	**VRAM (%)**	**Bandwidth (MB)**	**Time (s/batch)**	**Energy (Wh)**	**Acc (%)**
2 (Default)	–	3838.69	13.15	106.06	65.60
3	−0.3%	3825.79 (−0.3%)	13.04 (−0.9%)	103.77 (−2.2%)	66.42
4	−0.6%	3815.47 (−0.6%)	12.98 (−1.2%)	103.67 (−2.3%)	66.99

Table 6.3.10: Sensitivity to the patience P on FedMoECap-R (CommonsenseQA, NVIDIA Jetson AGX Orin). Percentages are relative to the default patience of 2.

Patience (P)	**VRAM (%)**	**Bandwidth (MB)**	**Time (s/batch)**	**Train Time/Client (s)**	**Acc (%)**
2 (Default)	–	3892.10	7.40	2480.05	80.76
3	+0.0%	3880.98 (−0.3%)	7.39 (−0.1%)	2467.27 (−0.5%)	81.53
4	−0.0%	3870.15 (−0.6%)	7.41 (+0.3%)	2476.73 (−0.1%)	81.44

Table 6.3.11: Sensitivity to the patience P on FedMoECap-R (BoolQ, NVIDIA A100). Percentages are relative to the default patience of 2.

The results reveal a pattern that is fundamentally different from what was observed for FedMoECap-P. Across both platforms and all patience values, **every efficiency metric remains essentially unchanged**. Bandwidth varies by at most −0.6%, energy by at most −2.3%, and training time by at most −0.5% relative to the default. These are not statistical artefacts of averaging, as was the case for FedMoECap-P: the Rolling mechanism genuinely keeps the number of simultaneously active experts constant throughout training by immediately replacing each converged expert with an untrained one. The per-round resource footprint is therefore structurally invariant to patience, regardless of how long experts are required to demonstrate convergence before being swapped.

Despite this efficiency stability, **accuracy improves with patience**. On the Jetson, accuracy increases consistently from 65.60% at $P = 2$ to 66.42% at $P = 3$ and 66.99% at $P = 4$, a cumulative gain of +1.39 points at no meaningful resource cost. On the A100, accuracy improves from 80.76% to 81.53% at $P = 3$, with a slight regression to 81.44% at $P = 4$. The non-monotonic behaviour on the A100 is best interpreted as a minor statistical fluctuation rather than a meaningful reversal of the trend, as the differences at $P = 3$ and $P = 4$ are within normal run-to-run variance.

The reason higher patience helps accuracy in FedMoECap-R is the same as in FedMoECap-P: requiring more sustained evidence of convergence before acting on an expert ensures it has been more thoroughly trained before its slot is reassigned. In the Rolling context this means each expert contributes more meaningfully to the global model before being retired, leading to better overall specialisation.

A deeper interpretation of this result concerns the quality–quantity trade-off inherent to the Rolling strategy. With lower patience, converged experts are swapped out sooner, allowing the training budget to be spread across a larger fraction of the expert pool. With higher patience, the same budget is concentrated on fewer experts but each is trained more thoroughly before being retired. The consistent accuracy improvement with patience suggests that **depth of training on the most task-relevant experts is more valuable than breadth of coverage across a larger set**. This is consistent with the motivation behind the Activation Freeze Criterion: the most highly routed experts are the most task-relevant, and ensuring they are trained to a high quality before being swapped out yields better model specialisation than prematurely rotating them in favour of experts that are intrinsically less activated for the task.

6.3.6 ESFT vs Activation Freeze Criterion

This experiment compares our proposed Activation Freeze Criterion against the expert selection method of ESFT [14], which selects experts on a per-layer basis by applying an activation threshold independently within each MoE layer. To ensure a fair comparison, we calibrate the ESFT threshold so that the total number of trainable experts is approximately equal to that selected by our method at a freeze rate of 0.90. Both configurations are evaluated using the FedMoECap-S variant, and since the number of trainable experts is approximately matched between the two methods, all efficiency metrics — bandwidth, VRAM, energy, and training time — are expected to be approximately equal. We therefore focus exclusively on final accuracy.

Expert Selection Criterion	**BoolQ (A100)**	**CommonsenseQA (Jetson)**
Activation Freeze (Ours)	82.11	67.24
ESFT [14]	81.62	66.75

Table 6.3.12: Accuracy comparison between the Activation Freeze Criterion and ESFT as the expert selection method within FedMoECap-S, with the number of trainable experts matched between the two criteria.

The Activation Freeze Criterion outperforms ESFT by +0.49 percentage points on both benchmarks, a modest but consistent margin across two different tasks and hardware platforms. We attribute this advantage to the structural difference between the two selection policies. ESFT enforces a per-layer quota, distributing a fixed number of trainable experts to every layer regardless of its relevance to the task. This forces it to allocate training budget to experts in layers that may contribute little to the target distribution. Our global pooling approach, by contrast, ranks all experts across all layers jointly and concentrates the training budget on the most activated experts network-wide, which can leave entire layers frozen if their experts are less task-relevant, as illustrated in Section 5.2.

Beyond accuracy, the Activation Freeze Criterion also offers a practical advantage in terms of interpretability and controllability. Because the freeze rate f_r directly and exactly determines what fraction of the total expert pool is trained, practitioners can reason about and set resource budgets in a transparent and hardware-agnostic way. ESFT's per-layer threshold, by contrast, requires tuning an indirect parameter whose effect on the total number of trainable experts depends on the routing distribution of the specific model and dataset, making it harder to predict and control the resulting resource footprint.

6.3.7 FedMoECap-P with Freeze Rate 0 vs FedAvg-LoRA vs FedMoECap-S

This experiment investigates an alternative operating mode of FedMoECap-P in which the initial freeze rate is set to $f_r = 0$, meaning that every expert is trainable from the very first round. Rather than relying on the Activation Freeze Criterion to pre-select a sparse subset of experts, this configuration allows the Pruning strategy to discover on its own which experts are worth retaining, by progressively freezing those whose relative parameter updates fall below the convergence threshold τ. We compare this variant against FedAvg-LoRA (All Experts) and FedMoECap-S with the default freeze rate of 0.90, which represents the opposite extreme: a fixed, activation-guided sparse selection held constant throughout training.

The most striking finding is that **FedMoECap-P $f_r = 0$ closely matches the accuracy of FedMoECap-S** on both platforms, despite never using the Activation Freeze Criterion. On CommonsenseQA it achieves 66.26%, only 0.98 points below FedMoECap-S (67.24%), and on BoolQ it reaches 81.87%, just 0.24 points below FedMoECap-S (82.11%). This suggests that the task-relevant fine-tuning signal is not uniquely concentrated in the activation-guided expert subset selected by FedMoECap-S, and that the Pruning strategy is capable of identifying an equally effective sparse training configuration through convergence-based elimination alone, without any prior knowledge of expert routing behaviour.

Method	VRAM (%)	Bandwidth (MB)	Time (s/batch)	Energy (Wh)	Acc (%)
FedAvg-LoRA (All Experts)	–	37971.81	16.04	123.15	70.43
FedMoECap-S fr=0.9	−9.1%	3800.97 (−90.0%)	13.05 (−18.6%)	104.01 (−15.5%)	67.24
FedMoECap-P fr=0.9	−9.2%	1603.00 (−95.8%)	13.00 (−19.0%)	60.56 (−50.8%)	62.41
FedMoECap-P fr=0	−3.4%	17266.22 (−54.5%)	13.22 (−17.6%)	117.69 (−4.4%)	66.26

Table 6.3.13: Comparison of FedMoECap-P with freeze rate 0 against FedAvg-LoRA and FedMoECap-S (CommonsenseQA, NVIDIA Jetson AGX Orin). Percentages are relative to FedAvg-LoRA (All Experts).

Method	VRAM (%)	Bandwidth (MB)	Time (s/batch)	Train Time/Client (s)	Acc (%)
FedAvg-LoRA (All Experts)	–	37963.00	8.69	2923.42	84.28
FedMoECap-S fr=0.9	−7.4%	3853.00 (−89.8%)	7.24 (−16.7%)	2422.52 (−17.1%)	82.11
FedMoECap-P fr=0.9	−13.5%	1789.00 (−95.3%)	6.71 (−22.8%)	1528.05 (−47.2%)	77.43
FedMoECap-P fr=0	−2.8%	18656.00 (−50.9%)	8.37 (−3.7%)	2800.94 (−4.2%)	81.87

Table 6.3.14: Comparison of FedMoECap-P with freeze rate 0 against FedAvg-LoRA and FedMoECap-S (BoolQ, NVIDIA A100). Percentages are relative to FedAvg-LoRA (All Experts).

However, this accuracy parity comes at a substantial resource cost. The initial rounds of FedMoECap-P fr=0 are identical to FedAvg-LoRA, as all experts are active before any pruning occurs. This is directly reflected in the **bandwidth** figures: 17266 MB on the Jetson (−54.5% relative to FedAvg) and 18656 MB on the A100 (−50.9%), compared to just 3801 MB and 1789 MB for FedMoECap-S and FedMoECap-P $f_r = 0.9$ respectively. **Energy consumption** on the Jetson tells the same story: FedMoECap-P $f_r = 0$ consumes 117.69 Wh, nearly identical to the FedAvg baseline of 123.15 Wh, whereas FedMoECap-P $f_r = 0.9$ consumes only 60.56 Wh.

These results reveal a clear trade-off between the two approaches to expert selection. The Activation Freeze Criterion of FedMoECap-S identifies the most relevant experts upfront using a cheap one-time profiling pass, and maintains their efficiency advantage throughout every training round. FedMoECap-P fr=0 instead pays the full resource cost of training all experts in the early rounds before pruning converges to a similarly sparse set, ultimately achieving comparable accuracy but at roughly 4–5× the bandwidth cost. For resource-constrained deployments, the Activation Freeze Criterion therefore remains the preferred approach, while FedMoECap-P $f_r = 0$ may be of interest in settings where routing-based profiling is unavailable or unreliable, and the extra upfront resource expenditure is acceptable.

6.4 Pareto Analysis

The preceding sections reported accuracy and efficiency metrics separately. Here we synthesise all experimental results into Pareto front plots that directly visualise the accuracy–efficiency trade-off for every configuration tested. A point lies on the Pareto frontier if no other configuration achieves higher accuracy at equal or lower resource cost. Points below the frontier are *dominated*: there exists at least one configuration that is both more accurate and more efficient. We plot two pairs of figures: accuracy versus total communication bandwidth (Figure 6.4.1), which is the primary bottleneck for federated deployments over constrained networks, and accuracy versus energy consumption on the Jetson and training time per client on the A100 (Figure 6.4.2), which capture the compute cost borne by each participating device.

Bandwidth frontier. Across both platforms, the Pareto frontier for communication bandwidth is defined exclusively by **FedMoECap-S**, **FedMoECap-P** and the FedAvg-LoRA baseline. All three freeze-rate configurations of FedMoECap-S (fr=0.80, fr=0.90, fr=0.95) lie on the frontier, forming a smooth curve that traces the best achievable accuracy at each bandwidth level. Moving from fr=0.80 to fr=0.95 halves the bandwidth at each step with only marginal accuracy loss, confirming that the freeze rate is a precise and well-behaved lever for navigating this frontier. The leftmost points on the Pareto

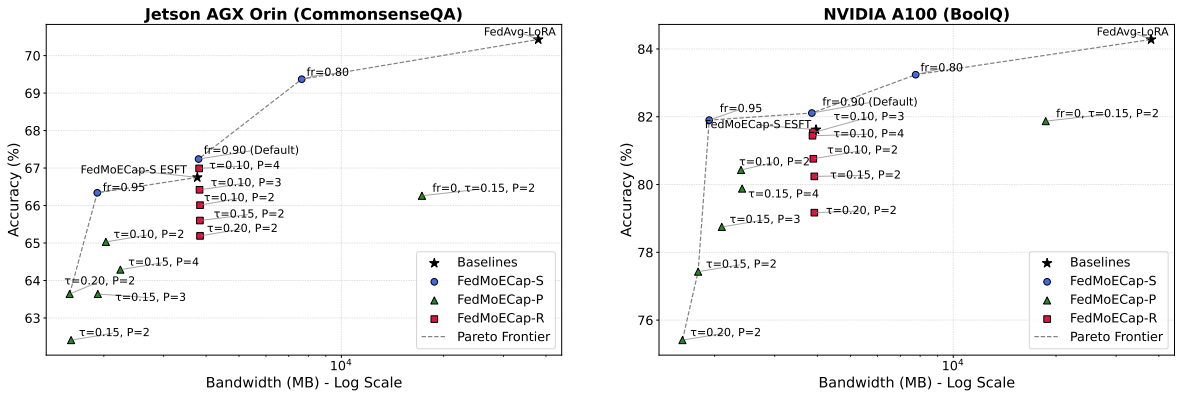


Figure 6.4.1: Pareto front: accuracy vs. total communication bandwidth on the Jetson AGX Orin (CommonsenseQA, left) and the NVIDIA A100 (BoolQ, right). The dashed line connects Pareto-optimal configurations. Points below the frontier are dominated.

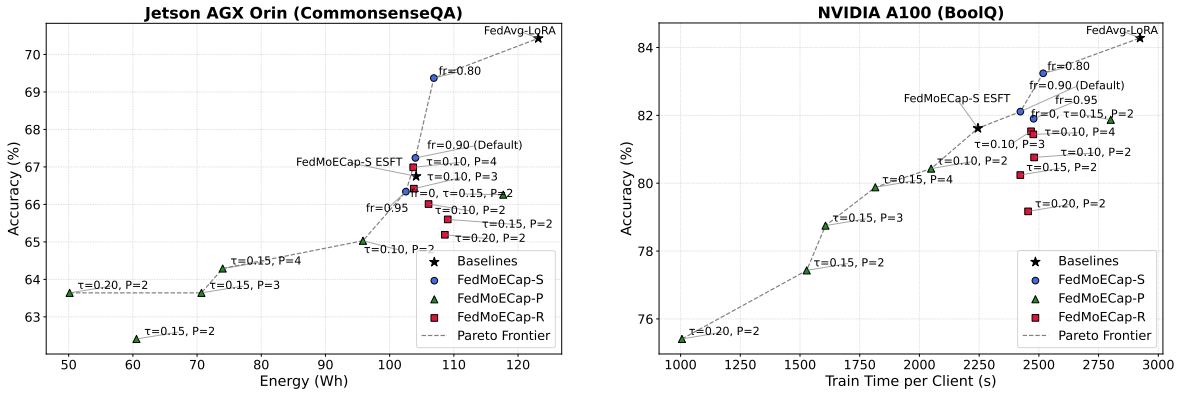


Figure 6.4.2: Pareto front: accuracy vs. total energy consumption on the Jetson AGX Orin (CommonsenseQA, left) and accuracy vs. training time per client on the NVIDIA A100 (BoolQ, right). The dashed line connects Pareto-optimal configurations.

Front are P variants with higher threshold and lower patience. All FedMoECap-R configurations are *dominated* on the bandwidth axis: for any given bandwidth budget, there exists a FedMoECap-S configuration that achieves equal or higher accuracy. This is a direct consequence of FedMoECap-S's design: it uses a fixed sparse expert set with the highest possible activation rate throughout training, that are trained thoroughly without premature swapping.

Energy and compute frontier. The picture changes substantially when the resource axis is energy or training time. Here, **FedMoECap-P** defines the Pareto frontier across the entire low-resource region of both plots. On the Jetson, the P variants with $\tau = 0.20$ and $\tau = 0.15$ at various patience levels occupy the optimal curve from 50 Wh up to approximately 95 Wh, at approximately 100 Wh the P variant with $\tau = 0.1$ with Patience 2 and then the FedMoECap-S takes over. On the A100, FedMoECap-P variants dominate the training-time axis up to approximately 2050 seconds, beyond which FedMoECap-S again provides the best trade-off. This finding reflects the progressive nature of the Pruning strategy: as experts converge and are frozen over the course of training, the active compute workload continuously shrinks, yielding energy and time savings that a static method like FedMoECap-S cannot match. Notably, FedMoECap-P with $\tau = 0.20$, $P = 2$ achieves the single most energy-efficient configuration in the entire experiment (50.15 Wh on the Jetson, 1005.85 seconds on the A100), at the cost of approximately 2 percentage points of accuracy relative to FedMoECap-S.

FedMoECap-R is dominated everywhere. In both figures and on both platforms, no FedMoECap-R configuration lies on the Pareto frontier. On the bandwidth axis, the R variants cluster in a tight vertical band at approximately 3800–3900 MB, well to the right of the FedMoECap-S frontier. On the energy and time axes, the R variants sit above 100 Wh and above 2400 seconds respectively, where FedMoECap-S already provides superior accuracy. This does not mean FedMoECap-R is without value: it achieves higher accuracy than FedMoECap-P at comparable bandwidth, and its threshold-agnostic behaviour makes it robust to hyperparameter miscalibration. However, for a practitioner explicitly optimising the accuracy–efficiency trade-off, either FedMoECap-S or FedMoECap-P will always provide a better operating point depending on whether bandwidth or compute is the binding constraint. In the next chapter we will discuss some ideas that may improve the performance of the R variant and are future work.

Hyperparameter sensitivity as frontier navigation. A key practical insight from these plots is that hyperparameter tuning does not uniformly improve all metrics simultaneously: it moves configurations along specific directions in the accuracy–efficiency plane. For FedMoECap-S, varying the freeze rate moves points horizontally along the bandwidth frontier. For FedMoECap-P, varying τ moves points diagonally along the energy frontier, while varying patience moves points upward (better accuracy) at a moderate horizontal cost. For FedMoECap-R, all hyperparameter changes produce nearly vertical movement — accuracy changes with negligible efficiency impact — which is both a strength and a limitation of the Rolling strategy.

Chapter 7

Conclusion

7.1 Summary of Contributions

This thesis presented **FedMoECap**, a federated fine-tuning framework specifically designed to exploit the structural sparsity of Mixture-of-Experts large language models. The central motivation was a gap in the existing literature: while PEFT methods such as LoRA have made fine-tuning tractable for dense models, and federated learning has extended this paradigm to privacy-sensitive distributed settings, the combination of MoE architectures with federated PEFT remained largely unexplored. Prior federated approaches treated MoE models as monolithic dense networks, missing the opportunity to exploit expert routing information for resource allocation.

FedMoECap addresses this gap through a set of tightly integrated components. The **Activation Freeze Criterion** introduces a global expert ranking based on token-level routing statistics collected in a single lightweight profiling pass. By pooling all $L \times N$ experts into one global pool and ranking them across layer boundaries, the criterion allows the framework to allocate the entire training budget to the most task-relevant experts, and to freeze entire layers when none of their experts rank highly enough. This is a principled improvement over per-layer methods such as ESFT [14], which are forced to distribute a fixed training quota to every layer regardless of its relevance to the fine-tuning task.

Selective LoRA then applies Low-Rank Adaptation exclusively to the trainable experts identified by the Activation Freeze Criterion, minimising both the VRAM footprint and the communication overhead of each federated round. Per-client freeze rates allow hardware-heterogeneous participants to independently calibrate their trainable parameter budget, making FedMoECap a first-class solution for the device diversity that characterises real-world federated deployments.

Building on this foundation, FedMoECap offers three adaptive convergence management strategies. **FedMoECap-S** (Static) fixes the expert selection at the first round and maintains it throughout training, providing the cleanest and most bandwidth-efficient operating point. **FedMoECap-P** (Pruning) progressively freezes experts as their parameter updates fall below a patience-gated convergence threshold, continuously reducing the active workload and delivering the largest energy and training-time savings. **FedMoECap-R** (Rolling) replaces converged experts with previously frozen ones, maximising the total number of experts that receive a fine-tuning signal within a fixed per-round resource budget. To prevent catastrophic forgetting in the Rolling variant, a **Hybrid Aggregation Rule** preserves the contributions of converged clients at the aggregation step, anchoring the global weights to accumulated knowledge while continuing to integrate updates from active clients.

7.2 Key Findings

The experimental evaluation on OLMoE-1B-7B across four commonsense reasoning benchmarks and two hardware tiers produced several findings that together characterise the practical value of FedMoECap.

Accuracy is preserved with a fraction of experts. All three FedMoECap variants substantially outperform the frozen base model on every benchmark, demonstrating that selective fine-tuning of only 10% of the expert pool is sufficient to produce meaningful task adaptation. FedMoECap-S, the most accurate of the three variants, achieves near-parity with the FedAvg-LoRA (All Experts) baseline across all benchmarks: the gap is at most 3.19 percentage points on the most demanding task (CommonsenseQA), and as small as 0.85 points on PIQA. These results confirm that the Activation Freeze Criterion effectively identifies the small subset of experts that are most informative for the task, so that training them alone closely approximates the benefit of training all experts.

Communication overhead is reduced by up to 95%. The most dramatic efficiency gain is in communication bandwidth. FedMoECap-S and FedMoECap-R reduce total bandwidth by approximately 90% relative to FedAvg-LoRA on both hardware platforms (from approximately 38 GB down to around 3.8 GB), while FedMoECap-P achieves a reduction exceeding 95% (down to approximately 1.6–1.8 GB). This improvement is a direct structural consequence of transmitting only the LoRA adapter deltas of the active 10% of experts each round, and it generalises across hardware generations without modification. For federated deployments over bandwidth-limited links such as cellular or satellite networks, a reduction of this magnitude can be the difference between a viable and an infeasible system.

Energy and training time are halved by the Pruning strategy. FedMoECap-P delivers the largest compute savings of any variant. On the NVIDIA Jetson AGX Orin, it reduces total energy consumption by 50.8% relative to FedAvg-LoRA, from 123.15 Wh to 60.56 Wh. On the NVIDIA A100, it reduces average training time per client by 47.2%, from 2923.42 to 1528.05 seconds. These savings stem from the progressive nature of the Pruning mechanism: as experts converge over successive rounds, the active training workload monotonically shrinks, so that the system operates increasingly efficiently as training progresses.

Each variant occupies a distinct and complementary niche. The Pareto analysis synthesises the accuracy–efficiency trade-off across all configurations and reveals a clear specialisation among the three variants. FedMoECap-S dominates the communication bandwidth frontier: for any bandwidth budget, no other configuration achieves higher accuracy at lower communication cost. FedMoECap-P dominates the energy and training-time frontier in the low-resource region, where its progressive workload reduction yields savings that a static method cannot match. FedMoECap-R does not lie on either Pareto frontier, but occupies an intermediate accuracy position between S and P at roughly the same bandwidth cost as FedMoECap-S. Its primary advantage lies in maximising the total number of experts that receive any fine-tuning signal, which may be beneficial in settings where broad expert coverage is prioritised over strict resource minimisation.

Freeze rate is a precise and well-behaved control lever. The design and hyperparameter analysis confirms that varying the freeze rate between 0.80 and 0.95 moves configurations smoothly along the bandwidth Pareto frontier, allowing practitioners to navigate the accuracy–efficiency trade-off with a single intuitive parameter. The convergence threshold τ and patience P provide complementary control over the Pruning and Rolling strategies, with τ primarily governing energy savings and P primarily governing accuracy preservation.

7.3 Limitations

Despite the promising results, FedMoECap has several limitations that should be acknowledged.

The Activation Freeze Criterion relies on expert routing statistics obtained from the pre-trained router, which is frozen throughout training. While this design avoids per-round re-profiling overhead, it also means that the expert selection cannot adapt to distributional shifts that may occur during fine-tuning. In highly dynamic or continual learning settings where the task distribution changes significantly over the course of training, a static selection may become suboptimal.

The evaluation is conducted on a single MoE model family (OLMoE-1B-7B) and four commonsense reasoning benchmarks. Although these provide a broad and diverse test bed, the generalisability of the findings to other MoE architectures — such as those with different routing mechanisms or expert granularities — and to other task types such as generation or summarisation remains to be verified.

The Rolling variant, while conceptually appealing, did not achieve Pareto-optimal performance in any of the evaluated configurations. The hybrid aggregation rule effectively prevents catastrophic forgetting, but the sequential exposure of different experts to the fine-tuning signal across rounds introduces heterogeneity in the degree to which individual experts are trained, which may limit the final model quality. Furthermore, PIQA results for FedMoECap-R were not yet available at the time of writing, leaving one benchmark slot in the accuracy comparison incomplete.

Finally, all experiments simulate a federated environment on shared hardware rather than deploying to a physically distributed system. Factors such as network latency variance, client availability, and stragglers are not modelled in the current evaluation.

7.4 Future Work

Several directions extend naturally from the current framework and could address the limitations identified above.

Improving the Rolling variant. The FedMoECap-R strategy is the most complex of the three variants but also the most promising in terms of maximising expert coverage. Future work could investigate alternative expert replacement schedules, such as selecting newly activated experts based on their complementarity to already-trained ones rather than purely by activation rank. Additionally, maintaining a fixed core of always-trainable high-activation experts while restricting the rolling mechanism to a smaller swappable subset may mitigate premature convergence and improve final accuracy. Weighting the contribution of clients in the hybrid aggregation rule by a measure of their local contribution, rather than purely by dataset size, may also improve the fidelity of the preserved knowledge. Concretely, one could replace the raw dataset size $|\mathcal{D}_i|$ with an *Effective Sample Size* $N_{l,e}^i = |\mathcal{D}_i| \cdot r_{l,e}^i$, where $r_{l,e}^i$ is the routing probability of expert e in layer l on client i, reflecting how much each client actually relies on that expert. Furthermore, introducing a scalar hyperparameter $\lambda \in [0, 1]$ into the Hybrid Aggregation Rule would give practitioners explicit control over the balance between preserving old knowledge and integrating new updates:

$$w_{t+1}^e = \lambda \sum_{i \in \mathcal{S}_{\text{conv}}} \frac{N_{l,e}^i}{N_{l,e}^{\text{total}}} w_t^e + (1 - \lambda) \sum_{j \in \mathcal{S}_{\text{active}}} \frac{N_{l,e}^j}{N_{l,e}^{\text{total}}} w_{t+1,j}^e \tag{7.4.1}$$

where $N_{l,e}^{\text{total}} = \sum_{k=1}^{K} N_{l,e}^k$. Setting $\lambda = 0.5$ recovers a balanced anchor, while tuning λ closer to 1 prioritises stability for experts that have already converged on a large fraction of the federation.

Cross-architecture generalisation. Evaluating FedMoECap on architectures beyond OLMoE-1B-7B, such as Mixtral [32] or DeepSeek-MoE [33], would test whether the gains reported here are specific to the OLMoE routing mechanism or reflect a general property of MoE fine-tuning. Architectures that

use different top-K_r routing strategies, shared expert pools, or auxiliary load-balancing losses present distinct profiling challenges that merit dedicated investigation.

Freeze rate assignment based on client hardware The current framework requires each client to specify its freeze rate as a static configuration parameter. An adaptive mechanism that infers a client's hardware capacity from metrics such as observed round latency, hardware speed and dataset size, and automatically adjusts its freeze rate accordingly, would simplify deployment and improve participation rates in heterogeneous federations where device capabilities are not known in advance.

Evaluation on generative tasks and real deployments. The current evaluation focuses exclusively on discriminative commonsense reasoning benchmarks. Extending the evaluation to generation tasks such as summarisation, question answering with long-form output, or instruction following would provide a more complete picture of FedMoECap's applicability. Similarly, a deployment study on a physically distributed federation of edge devices would expose the impact of real-world factors such as network latency variance, client dropouts, and stragglers on the overall training dynamics and convergence behaviour.

7.5 Closing Remarks

The confluence of large language models, sparse mixture-of-experts architectures, and federated learning defines one of the most challenging and practically important frontiers in modern machine learning systems research. Each of these paradigms individually imposes steep resource demands; their combination compounds the difficulty. FedMoECap demonstrates that this challenge is not intractable: by treating expert routing structure as a first-class design input rather than an implementation detail, it is possible to reduce communication overhead by up to 95% and energy consumption by over 50% while preserving the vast majority of the accuracy gain achievable by full federated fine-tuning.

The three variants of FedMoECap — Static, Pruning, and Rolling — provide a coherent family of solutions that cover the dominant operating regimes encountered in practice: strict bandwidth budgets, energy-constrained edge deployments, and maximum expert coverage scenarios. Together, they represent a principled step toward making federated fine-tuning of large MoE models a practical reality on the heterogeneous, resource-constrained devices that increasingly define the edge of modern AI infrastructure.

Bibliography

[1] Xin Luna Dong et al. "Towards Next-Generation Intelligent Assistants Leveraging LLM Techniques". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, Aug. 2023, pp. 5792–5793. ISBN: 9798400701030. DOI: 10.1145/3580305.3599572.

[2] Zeyu Han et al. "Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey". In: (Sept. 2024). URL: http://arxiv.org/abs/2403.14608.

[3] Nguyen Truong et al. "Privacy preservation in federated learning: An insightful survey from the GDPR perspective". In: *Computers & Security* 110 (2021), p. 102402. DOI: 10.1016/j.cose.2021.102402.

[4] Olivia Choudhury et al. "Anonymizing Data for Privacy-Preserving Federated Learning". In: (2020). URL: https://arxiv.org/abs/2002.09096.

[5] H. Brendan McMahan et al. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. Tech. rep. arXiv, 2017. URL: https://arxiv.org/abs/1602.05629.

[6] Jianyi Zhang et al. "TOWARDS BUILDING THE FEDERATEDGPT: FEDERATED INSTRUCTION TUNING". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. Institute of Electrical and Electronics Engineers Inc., 2024, pp. 6915–6919. DOI: 10.1109/ICASSP48485.2024.10447454.

[7] Yae Jee Cho et al. *Heterogeneous LoRA for Federated Fine-tuning of On-Device Foundation Models*. Tech. rep. ArXiv, 2024. URL: https://arxiv.org/abs/2401.06432.

[8] Tian Li et al. "Federated Optimization in Heterogeneous Networks". In: (2020). URL: https://arxiv.org/abs/1812.06127.

[9] Edward J. Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: (Oct. 2021). URL: http://arxiv.org/abs/2106.09685.

[10] Noam Shazeer et al. "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer". In: (Jan. 2017). URL: http://arxiv.org/abs/1701.06538.

[11] Niklas Muennighoff et al. "OLMoE: Open Mixture-of-Experts Language Models". In: (Mar. 2025). URL: http://arxiv.org/abs/2409.02060.

[12] Hanzi Mei et al. "FedMoE: Personalized Federated Learning via Heterogeneous Mixture of Experts". In: (2024). URL: https://arxiv.org/abs/2408.11304.

[13] Zihan Fang et al. "HFedMoE: Resource-aware Heterogeneous Federated Learning with Mixture-of-Experts". In: (2026). URL: https://arxiv.org/abs/2601.00583.

[14] Zihan Wang et al. "Let the Expert Stick to His Last: Expert-Specialized Fine-Tuning for Sparse Architectural Large Language Models". In: (July 2024). URL: http://arxiv.org/abs/2407.01906.

[15] Ashish Vaswani et al. *Attention Is All You Need*. Tech. rep. 2023.

[16] GeeksForGeeks. *Collaborative Learning – Federated Learning*. https://www.geeksforgeeks.org/machine-learning/collaborative-learning-federated-learning/. Accessed: 2026. 2024.

[17] Tim Dettmers et al. "LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale". In: (2022). URL: http://arxiv.org/abs/2208.07339.

[18] Tim Dettmers et al. "QLoRA: Efficient Finetuning of Quantized LLMs". In: (2023). URL: http://arxiv.org/abs/2305.14314.

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the Knowledge in a Neural Network". In: (Mar. 2015). URL: http://arxiv.org/abs/1503.02531.

[20] Feijie Wu et al. "FedBiOT: LLM Local Fine-tuning in Federated Learning without Full Model". In: (Aug. 2024), pp. 3345–3355. ISSN: 2154817X. DOI: 10.1145/3637528.3671897.

[21] Young Jin Kim, Raffy Fahim, and Hany Hassan Awadalla. "Mixture of Quantized Experts (MoQE): Complementary Effect of Low-bit Quantization and Robustness". In: (Oct. 2023). URL: http://arxiv.org/abs/2310.02410.

[22] Van-Tuan Tran, Khiem Le, and Quoc-Viet Pham. "REVISITING SPARSE MIXTURE OF EXPERTS FOR RESOURCE-ADAPTIVE FEDERATED FINE-TUNING FOUNDATION MODELS". In: ().

[23] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: (Feb. 2015). URL: http://arxiv.org/abs/1502.01852.

[24] Yonatan Bisk et al. "PIQA: Reasoning about Physical Commonsense in Natural Language". In: (Nov. 2019). URL: http://arxiv.org/abs/1911.11641.

[25] Alon Talmor et al. "CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge". In: (Mar. 2019). URL: http://arxiv.org/abs/1811.00937.

[26] Christopher Clark et al. "BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions". In: (May 2019). URL: http://arxiv.org/abs/1905.10044.

[27] Keisuke Sakaguchi et al. "WinoGrande: An Adversarial Winograd Schema Challenge at Scale". In: (Nov. 2019). URL: http://arxiv.org/abs/1907.10641.

[28] Leo Gao et al. *A framework for few-shot language model evaluation*. 2023. URL: https://github.com/EleutherAI/lm-evaluation-harness.

[29] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. URL: https://arxiv.org/abs/1908.10084.

[30] Kaitao Song et al. "Mpnet: Masked and permuted pre-training for language understanding". In: *arXiv preprint arXiv:2004.09297* (2020).

[31] David Arthur and Sergei Vassilvitskii. "k-means++: The Advantages of Careful Seeding". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics.

[32] Albert Q. Jiang et al. "Mixtral of Experts". In: *CoRR* abs/2401.04088 (2024). arXiv: 2401.04088. URL: https://arxiv.org/abs/2401.04088.

[33] Damai Dai et al. "DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models". In: *CoRR* abs/2401.06066 (2024). arXiv: 2401.06066. URL: https://arxiv.org/abs/2401.06066.