

# **Efficient training of LLMs and Federated Learning**

Dissertation submitted in partial fulfilment of the requirements  
for the award of the degree of

**Master of Technology**

by

**Moyank Giri**

(Roll No. 12310830)

Under the Supervision of

**Dr. Gagan Raj Gupta**

**Dr. Ness B Shroff**



Data Science and Artificial Intelligence

Department of Computer Science and Engineering

**INDIAN INSTITUTE OF TECHNOLOGY BHILAI**

**Bhilai - 491001, India**

June, 2025

## **DEDICATION**

Dedicated To

My Mother

Mrs. Monalisa Giri

and

My Father

Mr. Sukanta Kumar Giri

and

My Sister

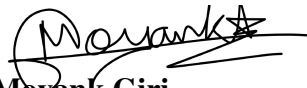
Ms. Suman Giri

## DECLARATION BY THE STUDENT

This is to certify that the thesis titled **Efficient Training of LLMs and Federated Learning** has been authored by me. It presents the research conducted by me under the supervision of **Dr. Gagan Raj Gupta and Dr. Ness B Shroff**. To the best of my knowledge, it is an original work, both in terms of research content and narrative, and has not been submitted elsewhere, in full or in parts, for the award of a degree or diploma. Further, due credit has been attributed to the relevant state-of-the-art collaborations (if any) with appropriate citations and acknowledgments, in line with established norms and practices.

Place: Indian Institute of Technology Bhilai

Date: 3rd July 2025



**Moyank Giri**

Master of Technology

Data Science and Artificial Intelligence

Indian Institute of Technology Bhilai

## CERTIFICATE AGAINST PLAGIARISM

This is to certify that the thesis entitled **Efficient training of LLMs and Federated Learning**, submitted by **Moyank Giri**, a bonafide record of the research work, done under my supervision is submitted to Indian Institute of Technology Bhilai on **June 25, 2025** to meet the requirements for the award of **Master of Technology** degree.

The contents of the thesis have been verified through similarity check software and I am convinced that the thesis has no component of plagiarism.



Place: Indian Institute of Technology Bhilai  
Date: 3rd July 2025

**Dr. Gagan Raj Gupta**  
Associate Professor  
Department of Computer Science and  
Engineering  
Indian Institute of Technology Bhilai

I have personally verified the report and find no significant similarity in the thesis.

Place: Indian Institute of Technology Bhilai  
Date:

**Dr. Anand M Baswade**  
Convener, DPGC  
Department of Computer Science and  
Engineering  
Indian Institute of Technology Bhilai

## THESIS CERTIFICATE

This is to certify that the thesis entitled **Efficient training of LLMs and Federated Learning**, submitted by **Moyank Giri** to Indian Institute of Technology Bhilai, for the award of the degree of **Master of Technology** is a bonafide record of research work carried out by the student under our supervision. The contents of the thesis, in full or in parts, have not been separately submitted to any Institute or University for the award of a degree or diploma.



**Dr. Gagan Raj Gupta**

Associate Professor

Department of Computer Science and Engineering

Indian Institute of Technology Bhilai

**Dr. Ness B Shroff**

Chaired Professor of ECE and CSE

Institute Director, NSF AI-Institute for Future Edge Networks and Distributed Intelligence (AI-EDGE)

Place: Indian Institute of Technology Bhilai

Date: 3rd July 2025

Place: Ohio State University, USA

Date: 5th July 2025

## ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to the Almighty for granting me the opportunity to be part of this esteemed institution and for blessing me with the support and guidance of remarkable individuals who have made my journey through IIT-Bhilai truly memorable.

My experience at IIT-Bhilai during my M.Tech has been truly remarkable. I've had the chance to challenge myself in various ways, and I feel incredibly lucky to have had an exceptional support network that has helped me navigate these past two years.

First and foremost, I extend my heartfelt appreciation to my supervisors, **Dr. Gagan Raj Gupta** and **Dr. Ness B Shroff**, whose mentorship has been invaluable. His profound insights, continuous support, and unwavering belief in my abilities have been instrumental in shaping my academic and personal growth. I am deeply grateful for his guidance throughout my thesis work.

I also extend my thanks to the members of the research seminar panel, **Dr. Anand M. Baswade**, **Dr. Rajesh Kumar Mundotiya** and **Dr. Kedarisetty Siddhardha** for their valuable feedback and encouragement that motivated me to strive for excellence in completing my thesis.

My journey would not have been as fulfilling without the support of my colleagues, friends, and mentors at IIT-Bhilai. I am thankful to my senior, Manisha, for their guidance and insights. I appreciate the camaraderie and support of my colleagues, Manish, Harshit, Tanmoy, and Deep, whose friendship has made my time at IIT-Bhilai enriching.

Last but certainly not least, I am indebted to my family – my parents, brothers, and sisters – for their unwavering support, love, and encouragement. Their belief in me has been my pillar of strength, motivating me to overcome challenges and pursue my academic aspirations with dedication and passion.

I am grateful to every individual who has played a part in shaping my thesis and academic journey at IIT-Bhilai.

**Moyank Giri**

# ABSTRACT

Efficient training of deep learning models and Large Language Models (LLMs) is critical for scalable and privacy-preserving artificial intelligence, yet existing methods struggle with high computational costs, excessive communication overhead, and poor adaptation to domain-specific challenges. This thesis addresses these limitations through two novel contributions, each targeting a unique aspect of efficient model training.

First, an **Efficient Hybrid Split Federated Learning Framework** is proposed, which optimizes decentralized learning by balancing global generalization and local personalization. This is achieved through a gradient norm-based dynamic selection strategy, coupled with activation caching to minimize communication overhead. Extensive experiments on diverse datasets, including ISIC2019, CIFAR10, FMNIST, VLCS, PACS, and OfficeHome, demonstrate an average of 96× reduction in computation costs and a 4.5× reduction in communication traffic, while maintaining strong test-time performance across heterogeneous data distributions.

Second, a **Dynamic Fine-tuning Strategy for LLMs** is developed, addressing the high computational demands of fine-tuning large models. By employing selective layer training using gradient-based criteria, combined with parameter-efficient techniques like LoRA and caching mechanisms, the approach achieves a 10% reduction in fine-tuning time and significantly lowers GPU memory utilization, making LLM fine-tuning more resource-efficient.

Together, these contributions provide a scalable, resource-efficient, and domain-adaptive solution for training and deploying deep learning models and LLMs, pushing the boundaries of efficient AI.

**Keywords:** Federated Learning, Split Learning, Large Language Models, Efficient Fine-tuning, Communication Reduction, Computation Reduction.

# Contents

<b>DEDICATION</b>	<b>i</b>
<b>DECLARATION BY THE STUDENT</b>	<b>ii</b>
<b>CERTIFICATE AGAINST PLAGIARISM</b>	<b>iii</b>
<b>THESIS CERTIFICATE</b>	<b>iv</b>
<b>ACKNOWLEDGEMENTS</b>	<b>v</b>
<b>Abstract</b>	<b>vi</b>
<b>List of Tables</b>	<b>x</b>
<b>List of Figures</b>	<b>xi</b>
<b>Abbreviations</b>	<b>xii</b>
<b>Symbols</b>	<b>xiii</b>
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Problem Statement	1
1.2 Need for Efficient and Distributed Training Techniques	2
1.3 Challenges in Deep Learning Model Training	2
1.3.1 High Computational and Memory Costs	3
1.3.2 Data Fragmentation and Distribution Shift	3
1.3.3 Communication Constraints and Straggler Effects	3
1.3.4 Adaptability and Personalization Requirements	3
1.3.5 Deployment Mismatch and Evaluation Gaps	3
1.4 Distributed and Efficient Training Strategies	4
1.4.1 Unified Distributed Training for Vision Models	4
1.4.2 Efficient Fine-Tuning of Large Language Models	4
1.5 Problem Statement	5
1.6 Objectives	5
1.7 Scope and Limitations	5
1.8 Thesis Organization	6
1.9 Summary	6
<b>2 Literature Review</b>	<b>7</b>
2.1 Introduction	7
2.2 Federated Learning and Its Variants	7
2.3 Data Heterogeneity in Federated Learning	8



2.4	Personalized Federated Learning	8
2.5	Test-Time Adaptation and OOD Generalization	9
2.6	OOD Detection Techniques	9
2.7	Efficient Finetuning Techniques for LLMs	10
2.8	Summary	11
<b>3</b>	<b>Methodology</b>	<b>13</b>
3.1	Introduction	13
3.2	Adaptive Split Federated Learning (ASFL)	13
3.2.1	System Model Architecture	13
3.2.2	Loss Computation	15
3.2.3	Training Phases	15
3.2.3.1	Generalization Phase:	15
3.2.3.2	Personalization Phase:	16
3.2.3.3	GradientNorm OOD Threshold Calculation:	16
3.2.4	Key-Value Store	16
3.2.5	Inference	17
3.3	Efficient Fine-Tuning for Large Language Models	17
3.3.1	Encoder-Decoder Architectures (e.g., RoBERTa, T5)	17
3.3.1.1	System Design	17
3.3.1.2	Key Components	17
3.3.1.3	Training Pipeline	18
3.3.2	Decoder-Only Architectures (e.g., LLaMA, GPT)	19
3.3.2.1	Key Components	20
3.3.2.2	Taylor Approximation: Layer-Wise Summary	21
3.3.3	Application to Attention Projection Layers (Q, K, V, O)	21
3.3.4	Application to the MLP Layer with SwiGLU Activation	22
3.4	Summary	23
<b>4</b>	<b>Experimental Setup</b>	<b>24</b>
4.1	Introduction	24
4.2	Experimental Goals and Research Questions	24
4.3	Research Questions	24
4.3.1	Adaptive Split Federated Learning	24
4.3.2	Dynamic Layer Selection in LLMs	25
4.4	Hardware and Software Environment	25
4.5	Datasets and models	25
4.5.1	Computer Vision Datasets	25
4.5.2	Computer Vision Models	26
4.5.3	LLM Datasets	27
4.5.4	LLM Models	27
4.6	Experimental Design and Protocol	27
4.6.1	Adaptive Split Federated Learning	27
4.6.1.1	Evaluation Settings	27
4.6.1.2	Protocol	28
4.6.2	Dynamic Layer Selection in Large Language Models	28
4.6.2.1	LLM Fine-tuning Settings	28
4.6.2.2	Protocol	28

4.7	Evaluation Metrics	29
4.7.1	Model Performance: Accuracy	29
4.7.2	Efficiency Measurements	29
4.8	Summary	29
<b>5</b>	<b>Results and Discussion</b>	<b>30</b>
5.1	Introduction	30
5.2	Results for Federated Split Learning	30
5.2.1	ASFL-Q1: Performance Comparisons	31
5.2.1.1	Setting-1: IID Test Scenario	31
5.2.1.2	Setting-2: OOD Test Scenario	31
5.2.1.3	Setting-3: Mixed Test Scenario	31
5.2.1.4	Overall Performance Comparison	33
5.2.2	ASFL-Q2: Client-wise performance gains in OOD Scenarios	33
5.2.3	ASFL-Q3: Benefits of Contrastive Loss Strategy	34
5.2.4	ASFL-Q4: Efficiency Gains	35
5.2.4.1	Communication Efficiency	35
5.2.4.2	Computational Efficiency	35
5.3	Results for Efficient Fine-tuning of LLMs	36
5.3.1	LLM-RQ1: Performance Comparison	36
5.3.2	LLM-RQ2: Skip Connections Optimization:	40
5.4	Summary	41
<b>6</b>	<b>Conclusion and Future work</b>	<b>43</b>
6.1	Introduction	43
6.2	Summary of Findings	43
6.3	Contributions of the Thesis	43
6.4	Practical Implications	43
6.5	Scope for Future Work	44
6.6	Summary	44
	<b>Bibliography</b>	<b>49</b>
	<b>List of Publications</b>	<b>50</b>

## LIST OF TABLES

3.1	Dataset augmentations	14
5.1	Client test accuracies in Setting-1 (IID test)	31
5.2	Client test accuracies in Setting-2 (OOD Test )	32
5.3	Client test accuracy in Setting-3a (20%-OOD-80%-IID)	32
5.4	Client test accuracy in Setting-3b (50%-OOD-50%-IID)	32
5.5	Client test accuracy in Setting-3c (80%-OOD-20%-IID)	33
5.6	Accuracy of ASFL-G and ASFL-P with/without contrastive loss. Runs without contrastive loss are denoted by *.	34
5.7	Number of Times Increase in Communication Overhead Compared to Our Method (ASFL)	35
5.8	Communication overhead (GB) of various algorithms on benchmark datasets till convergence	36
5.9	Floating-point operations at the client in various Algorithms till convergence	36
5.10	Number of Times Increase in Computation Overhead Compared to Our Method (ASFL)	37

## LIST OF FIGURES

1.1	Challenges in Deep Learning Model training	2
3.1	System Model Architecture for Robust Split Learning, highlighting the modules and communication and storage mechanisms at different phases	14
3.2	Dynamic Layer Selection for Encoder-Decoder LLM Architectures	19
3.3	Dynamic Layer Selection for Encoder-Decoder LLM Architectures	20
5.1	Hybridization benefit in clients under OOD scenarios for CIFAR-10 Dataset.	34
5.2	Validation loss over time on the QQP dataset across different LoRA strategies. “Layer Selection + Reduction” (orange) achieves a strong balance between convergence speed and final loss, converging at approximately 3211.68 seconds. This is close to full-layer training (red), which converges fastest at 2870.48 seconds. “Per Layer Selection” (blue) follows at 3585.39 seconds, while “Reverse Selection” (green) lags significantly at 5528.22 seconds.	37
5.3	Evaluation loss over epochs on the SST-2 dataset across different LoRA strategies. All methods reach similar final loss, with full-layer training (red) converging fastest, followed closely by “Layer Selection + Reduction” (orange). “Per Layer Selection” (blue) and “Reverse Selection” (green) converge more slowly, though differences diminish after epoch 10.	38
5.4	Validation accuracy over epochs on the SST-2 dataset across different LoRA strategies. All methods reach similar final accuracy, with full-layer training (red) converging fastest, followed closely by “Layer Selection + Reduction” (green). “Per Layer Selection” (blue) and “Reverse Selection” (orange) converge more slowly	39
5.5	Evaluation loss of Qwen 2.5 using the proposed layer selection strategy. The model converges reliably, with steep early loss reduction and smooth stabilization across epochs, despite selective fine-tuning.	40
5.6	Training time comparison with skip connections (blue curve) and without skip connections (red curve) on the implemented algorithm for Encoder-Decoder architecture on QQP dataset. The skip-optimized model achieves up to 12% speedup without sacrificing performance.	41

## Abbreviations

<b>DL</b>	<b>Deep Learning</b>
<b>LLM</b>	<b>Large Language Models</b>
<b>SL</b>	<b>Split Learning</b>
<b>ASFL</b>	<b>Adaptive Split Federated Learning</b>
<b>FL</b>	<b>Federated Learning</b>
<b>IID</b>	<b>Independent and Identically Distributed</b>
<b>InD</b>	<b>In-Distribution</b>
<b>OOD</b>	<b>Out Of Distribution</b>
<b>NLP</b>	<b>Natural Language Processing</b>
<b>CNN</b>	<b>Convolutional Neural Network</b>
<b>DLT</b>	<b>Distributed Learning Techniques</b>
<b>LoRA</b>	<b>Low-Rank Adaptation</b>
<b>BERT</b>	<b>Bidirectional Encoder Representations from Transformers</b>
<b>GPT</b>	<b>Generative Pre-trained Transformer</b>
<b>GPU</b>	<b>Graphics Processing Unit</b>
<b>TPU</b>	<b>Tensor Processing Unit</b>
<b>DLS</b>	<b>Dynamic Layer Selection</b>
<b>RoBERTa</b>	<b>Robustly Optimized BERT Approach</b>
<b>SwiGLU</b>	<b>Switch Gated Linear Unit</b>
<b>SiLU</b>	<b>Sigmoid-weighted Linear Unit</b>
<b>TL</b>	<b>Transfer Learning</b>

## List of Symbols

$B_m$	The $m^{th}$ batch of input samples
$N$	Number of samples in a batch
$x_{nm}$	The $n^{th}$ sample in batch $B_m$
$\text{Score}(B_m)$	Aggregated score for batch $B_m$
$\text{score}(x)$	Gradient-based score for a single input $x$
$D_{KL}(u \parallel q)$	KL divergence between uniform distribution $u$ and model prediction $q$
$\mathbf{w}$	Model parameters (weights)
$\ \cdot\ _1$	L1 norm of a vector
$\mathbf{T}$	Threshold score used to classify ID/OOD, set via GradNorm
$\text{Discriminator}(B_m)$	Function to classify a batch as ID or OOD
ID	In-Distribution classification label
OOD	Out-of-Distribution classification label
$W$	Original weight matrix in a neural network layer
$\Delta W$	Update to weight matrix using LoRA
$A, B$	Low-rank adaptation matrices in LoRA
$d, r, k$	Dimensions of $A$ and $B$ with $r \ll \min(d, k)$
$f(W, x)$	Output of a layer with weights $W$ and input $x$
$f(W + \Delta W, x)$	Output after applying LoRA perturbation
$J_{\text{SwiGLU}}(z)$	Jacobian matrix of the SwiGLU activation function at $z$
$z_1, z_2$	Components of input $z$ to the SwiGLU function
$\text{SiLU}(z)$	Sigmoid-weighted Linear Unit activation function
$\text{SiLU}'(z)$	Derivative of the SiLU function
$\Delta z_1, \Delta z_2$	Changes in $z_1, z_2$ respectively
$\Delta h$	Approximate change in activation after perturbation
$xAB$	Projected perturbation from input $x$ through LoRA matrices $A$ and $B$
<code>combined_loss</code>	Final training loss combining classification and contrastive losses
<code>ce_loss</code>	Cross-entropy loss for classification
<code>sc_loss</code>	Supervised contrastive loss between augmented views
$\lambda_{\text{weight}}$	Weighting coefficient for contrastive loss term
$X_a, X_b$	Augmented views of raw input at the client side

# CHAPTER 1

## INTRODUCTION

### 1.1 Problem Statement

Over the past decade, deep learning has demonstrated transformative success across domains such as computer vision, natural language processing (NLP), and medical diagnostics. From convolutional neural networks (CNNs) in image classification to transformer-based large language models (LLMs) in language understanding, the field has evolved toward increasingly large and complex models trained on massive datasets. However, this growth has brought substantial challenges: these models demand immense computational resources, centralized access to diverse data, and complex infrastructure for both training and deployment.

Despite their remarkable capabilities, the training and fine-tuning of such models remain prohibitively expensive and inaccessible for most real-world applications. In scenarios like personalized healthcare analytics, industrial IoT, financial document analysis, or edge-based surveillance systems, computational budgets are limited, data is fragmented across locations, and communication may be unreliable or costly. Centralized training—where all data is aggregated in a single server—breaks down under these constraints.

Moreover, real-world data is often non-IID (non-identically and independently distributed), meaning that data collected by different users, institutions, or devices varies significantly in distribution and scale. Models trained under centralized and homogeneous assumptions may generalize poorly when deployed across diverse edge environments or user populations.

This thesis is motivated by a fundamental challenge: **how can we train and adapt trainable architectures—including convolutional and transformer-based models—efficiently across distributed, resource-constrained environments in a real-world data setting?** This question is especially pertinent when dealing with two dominant classes of models:

- **Vision-based models**, used in tasks such as object recognition, anomaly detection, and medical image segmentation.
- **Large Language Models (LLMs)**, used for text classification, question answering, summarization, and dialogue generation.

While these models differ in modality and structure, they share a common pain point—**training and fine-tuning them in practical settings is computationally intensive, data-hungry, and often infeasible without specialized solutions**. This thesis proposes a unified exploration of distributed and efficient training techniques that are capable of scaling both vision and language models while respecting resource constraints and real-world limitations.

## 1.2 Need for Efficient and Distributed Training Techniques

The need for training efficiency is not merely an optimization objective—it is foundational to enabling real-world deployment of intelligent systems. Efficient training frameworks allow models to:

- Scale across distributed devices without overwhelming infrastructure.
- Adapt to local or personalized data without retraining from scratch.
- Operate under communication, energy, and latency constraints.
- Generalize to data distribution shifts and non-stationary environments.

Two core technical directions have emerged to address this need:

1. **Distributed Learning Techniques (DLTs):** Federated Learning (FL) and Split Learning (SL) allow training models collaboratively across clients or devices without requiring centralized access to raw data.
2. **Efficient Fine-Tuning Strategies for LLMs:** Emerging methods like LoRA, adapters, and dynamic layer selection allow large-scale language models to be fine-tuned with minimal resource overhead.

This thesis bridges these domains by developing scalable, resource-aware training frameworks that apply to both vision and language tasks, without assuming access to centralized infrastructure or unlimited compute. By addressing the growing gap between state-of-the-art models and deployable solutions, the proposed work aims to democratize access to deep learning capabilities.

## 1.3 Challenges in Deep Learning Model Training

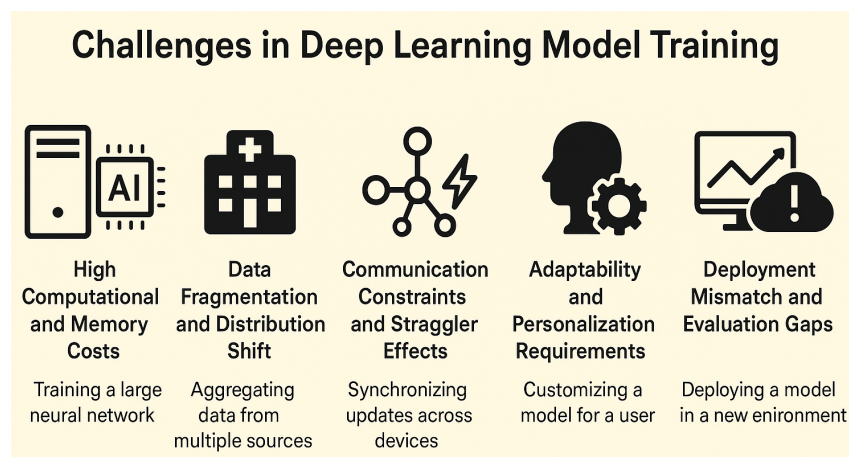


Figure 1.1: Challenges in Deep Learning Model training



Although deep learning systems are now ubiquitous, their training workflows face critical bottlenecks in practical deployments. These challenges span computation, communication, personalization, and generalization:

### **1.3.1 High Computational and Memory Costs**

Modern architectures, particularly transformer-based models like BERT or GPT variants, consist of hundreds of millions to billions of parameters. Fine-tuning or training such models from scratch requires high-end GPUs or TPUs, extensive memory, and long runtimes, which are not accessible in typical on-device, institutional, or real-time scenarios.

### **1.3.2 Data Fragmentation and Distribution Shift**

In most real-world systems, data is decentralized—collected by hospitals, mobile users, sensors, or enterprise systems. Centralizing such data is often infeasible due to governance restrictions, data ownership, and logistical constraints. Furthermore, local data distributions may vary significantly, making traditional IID-based training paradigms insufficient for robust generalization.

### **1.3.3 Communication Constraints and Straggler Effects**

Training in distributed settings requires frequent synchronization of gradients, weights, or activations. In networks with limited bandwidth, client variability, or heterogeneous hardware, this synchronization becomes a bottleneck. Clients may drop out, slow down the system, or overload communication channels, leading to inefficient training dynamics.

### **1.3.4 Adaptability and Personalization Requirements**

Many applications require models to adapt to user-specific or domain-specific contexts—e.g., customizing a language model to legal or clinical jargon. However, full fine-tuning of large models for each client is computationally and memory-wise impractical. Efficient personalization methods are needed to retain general model behavior while enabling adaptation.

### **1.3.5 Deployment Mismatch and Evaluation Gaps**

Training is typically performed in high-resource environments, while deployment occurs in low-resource, real-time, or distributed settings. Without aligned evaluation protocols and deployment-aware training strategies, performance degradation is common, especially in the presence of domain shift or novel tasks.

## 1.4 Distributed and Efficient Training Strategies

To address these challenges, this thesis develops and evaluates two major solutions under a unified vision of efficient deep learning training:

### 1.4.1 Unified Distributed Training for Vision Models

Federated Learning (FL) and Split Learning (SL) represent two major paradigms for collaborative model training without centralizing data. These are treated in this thesis as part of a broader class of **Distributed Learning Techniques (DLTs)**:

- **FL:** Clients train local models and transmit weight updates to a central aggregator. Data remains private, but communication overhead and model divergence are concerns.
- **SL:** Clients compute partial forward passes and transmit intermediate activations to a server that completes the computation. It reduces client workload but increases communication and cut-layer sensitivity.

This thesis proposes the **Adaptive Split Federated Learning (ASFL)** framework:

- Dynamically splits models across client-server boundaries.
- Selectively transmits only essential gradients or activations to reduce communication load.
- Introduces test-time adaptation using gradient norms to improve generalization under distribution shift.
- Supports heterogeneous client capabilities through flexible layer placement.

### 1.4.2 Efficient Fine-Tuning of Large Language Models

LLMs are notoriously expensive to fine-tune due to their size. However, adapting them to domain-specific tasks (e.g., legal document classification or telecom dialogue modeling) is crucial for performance.

This thesis explores parameter-efficient fine-tuning methods:

- **LoRA:** Injects low-rank updates into existing layers.
- **Adapters:** Inserts small bottleneck modules between frozen layers.
- **Dynamic Layer Selection (DLS): (Proposed)**—selects and updates only the most relevant layers based on per-layer gradient importance. Reduces training overhead without sacrificing accuracy.

## 1.5 Problem Statement

Despite individual advances in distributed training and efficient model adaptation, a unified and generalizable framework that supports both modalities under realistic constraints is lacking. Existing solutions are often:

- Modality-specific and not transferable across domains.
- Inefficient under client heterogeneity or data non-IID conditions.
- Inflexible in adapting to personalization or on-device adaptation needs.

This thesis addresses these gaps by proposing unified frameworks that can operate across both vision and NLP tasks under constraints of communication, compute, and data variability.

## 1.6 Objectives

The main objectives of this research are as follows:

1. Design an adaptive, communication-efficient training framework—**ASFL**—that merges FL and SL for scalable distributed training.
2. Develop a **Dynamic Layer Selection (DLS)** mechanism for scalable and personalized fine-tuning of LLMs.
3. Analyze robustness and generalization of proposed methods under distribution shift and real-world heterogeneity.
4. Evaluate the effectiveness of both approaches on diverse benchmarks across vision (OfficeHome, PACS, ISIC2019) and language (SST-2, QQP) domains.

## 1.7 Scope and Limitations

### Scope:

- Focus on training efficiency and scalability across distributed learning and LLM fine-tuning.
- Simulation of realistic distributed settings for vision models using ASFL.
- Development and evaluation of DLS for transformer-based language models.
- Empirical validation using standardized datasets and benchmark tasks.

### Limitations:

- Real-world asynchronous client deployment is out of scope.
- Fine-tuning strategies are evaluated on specific LLM backbones (e.g., RoBERTa, BERT).
- Limited to proof-of-concept in simulation; hardware-aware optimization is not deeply explored.

## 1.8 Thesis Organization

This thesis is organized as follows:

- **Chapter 1: Introduction** – Introduces the research problem, motivation, challenges, objectives, and scope.
- **Chapter 2: Literature Review** – Reviews federated learning, personalization, OOD detection, and efficient fine-tuning of large language models (LLMs).
- **Chapter 3: Methodology** – Describes the proposed Adaptive Split Federated Learning (ASFL) and dynamic LLM fine-tuning methods, including system design and training strategies.
- **Chapter 4: Experimental Setup** – Outlines datasets, protocols, hardware, and evaluation metrics for both vision and language models.
- **Chapter 5: Results and Discussion** – Presents results for ASFL and LLM tuning, covering performance, efficiency, and OOD adaptation.
- **Chapter 6: Conclusion and Future Work** – Summarizes findings, contributions, and suggests directions for future research.

## 1.9 Summary

This chapter introduced the motivation and context for investigating scalable and efficient training methodologies for deep learning systems across both vision and NLP domains. It identified critical challenges in computation, communication, and generalization, and outlined a unified research agenda based on distributed training (ASFL) and adaptive LLM fine-tuning (DLS). The rest of this thesis details the proposed methods, their implementation, and their evaluation across a diverse set of realistic tasks and environments.

## CHAPTER 2

### Literature Review

#### 2.1 Introduction

The rapid proliferation of Large Language Models (LLMs) and the increasing need for decentralized training paradigms such as Federated Learning (FL) have posed new challenges in terms of scalability, computational efficiency, personalization, and generalization across diverse data distributions. Efficient finetuning and training mechanisms have become crucial to adapting these models to real-world constraints without compromising performance. This literature review presents a comprehensive survey of foundational methods and state-of-the-art approaches in FL, test-time adaptation, out-of-distribution (OOD) generalization, and efficient finetuning of LLMs. Additionally, this section highlights the growing importance of domain-specific benchmarks, such as those in the telecommunications sector, to evaluate and improve LLM effectiveness under realistic constraints.

#### 2.2 Federated Learning and Its Variants

Federated Learning (FL) enables collaborative training of machine learning models across distributed clients while maintaining data privacy. The seminal work [1] proposed **FedAvg**, a simple yet powerful algorithm that averages locally trained models to form a global model. Despite its effectiveness, FedAvg struggles with data heterogeneity and slow convergence.

To address these issues, **FedProx** [2] was introduced. It extends FedAvg by adding a proximal term to the local objective, ensuring that local updates remain close to the global model. This mitigates divergence in heterogeneous environments.

Another major variant is **Split Learning (SL)** [3], where the model is split between the client and server, with only the shallow layers residing on the client. This significantly reduces client-side computation and memory consumption. However, SL requires back-and-forth communication of activations and gradients, leading to increased latency.

These FL variants collectively aim to improve scalability, privacy, and efficiency, but they still face challenges in generalization under heterogeneous data settings and adapting to unseen test-time distributions.

## 2.3 Data Heterogeneity in Federated Learning

Data heterogeneity—differences in data distributions across clients—remains a major hurdle in FL. As discussed by [4], heterogeneity can arise in the form of:

- **Label distribution shift ( $P(y)$ ):** Clients possess different label frequencies.
- **Feature distribution shift ( $P(x)$ ):** Variations in feature domains across clients.

[5] conducted empirical evaluations on FL setups and showed that this heterogeneity leads to poor model generalization and instability during training. Their work emphasized that both inter-client and intra-client distribution shifts significantly affect global model performance, especially under test-time shifts and OOD data.

## 2.4 Personalized Federated Learning

To tackle client-specific needs, Personalized Federated Learning (PFL) has emerged as a promising solution. Several techniques have been developed to adapt global models to local client distributions.

**Transfer Learning / Fine-tuning:** [6, 7] fine-tune the global model or specific layers locally, offering personalization at the cost of increased computation.

**Meta-learning:** Model-Agnostic Meta-Learning (MAML) integrated into FL, as proposed by [8, 9, 10, 11, 12], provides fast adaptation to new tasks using minimal gradient steps. These works demonstrate strong personalization capabilities, particularly in few-shot settings.

**Layer Personalization:** [13, 14] propose excluding certain layers from global aggregation, allowing clients to maintain local-specific features. This approach reduces negative transfer among clients with divergent distributions.

**Hypernetwork-based Personalization:** [15] introduces a hypernetwork that generates personalized parameters, facilitating knowledge sharing while preserving local model characteristics.

**Adaptive Aggregation:** [16] clusters clients based on similarity measures to perform adaptive aggregation. This strategy results in more coherent and efficient knowledge transfer.

**Regularization-based Approaches:** [17, 18, 19] add regularization terms (e.g., L2 norm penalties) to control model drift and improve personalization, balancing global consistency and local fit.

Despite their success, over-personalization remains a risk. [20] demonstrated that overly specific local models tend to overfit and perform poorly on unseen test distributions, thus necessitating generalizable personalization strategies.

## 2.5 Test-Time Adaptation and OOD Generalization

Intra-client heterogeneity and OOD data further complicate FL deployment. Several test-time adaptation methods have been proposed to address this:

**Entropy Minimization:** [21] introduced Test-time Entropy Minimization (TENT), which adapts model parameters to minimize prediction entropy on unlabeled test data, improving OOD performance.

**Selective Layer Finetuning:** [22] suggested fine-tuning only specific layers at test time, showing that this targeted adaptation can yield results comparable or superior to full-model fine-tuning.

**Self-Supervised Test-Time Training:** [23, 24] employed self-supervised objectives (e.g., rotation prediction, jigsaw tasks) during test time to update model weights, thus improving robustness.

### Global OOD Generalization Techniques:

- [25] incorporated attention mechanisms to focus on semantically salient regions in images.
- [26] added covariate-shift regularization to feature extractors, improving robustness under distribution drift.
- [27] aligned client gradients to reduce variance in global predictions.

Contrastive learning has also played a pivotal role. [28, 29, 30, 31] showed its effectiveness in learning robust visual representations. Building on this, [5] introduced FEDICON, which employs supervised contrastive loss to enhance inter-client generalization in FL.

## 2.6 OOD Detection Techniques

OOD detection identifies when test data deviates significantly from training distribution. Key approaches include:

**Likelihood-Based Detection:** [32, 33, 34] approximate data likelihood using generative models such as normalizing flows and autoencoders. These methods, though theoretically sound, often struggle in high-dimensional spaces.

**Feature Representation-Based:** [35, 36, 37, 38, 39, 40] use uncertainty metrics (e.g., softmax confidence, Mahalanobis distance) derived from model outputs to identify OOD samples.

**Gradient-Based Approaches:** [41, 42, 43] leverage the norm or direction of model gradients. Specifically, [42] demonstrated that the gradient norm of the final classification layer provides a reliable signal for OOD detection. [34] extended this by incorporating Fisher Information Matrix approximations, while [44] used projections onto low-rank subspaces.

In our work, we adapt the gradient-based approach of [42] within the Split Learning framework. Given the personalized nature of client models, the classification layer gradient effectively captures deviation from the training distribution.

## 2.7 Efficient Finetuning Techniques for LLMs

Recent advancements in finetuning methodologies for Large Language Models (LLMs) have focused on addressing the substantial computational and memory demands associated with adapting these models to specific tasks. A significant portion of research in this area revolves around selective layer finetuning—identifying and updating only a strategically chosen subset of model parameters or layers to reduce overhead while maintaining competitive performance.

The work titled *"Less is More: Selective Layer Finetuning with SubTuning"* [45] introduces **SubTuning**, a strategy that selectively fine-tunes a subset of layers in large pre-trained neural networks while freezing the remaining layers. This method is positioned between linear probing (which updates only the classification head) and full-model finetuning (which updates all layers). SubTuning significantly reduces computational cost by finetuning only a fixed subset of layers tailored to the downstream task and dataset. However, the approach is limited by its focus on standard neural networks rather than LLMs and by its use of a fixed, rather than dynamic, subset of layers.

Building on this idea, *"Gradient-Mask Tuning Elevates the Upper Limits of LLM Performance"* [46] presents **Gradient-Mask Tuning (GMT)**, a novel approach tailored specifically for LLMs. GMT uses task-specific gradient information to rank parameters and masks those with lower gradient magnitudes, allowing for selective parameter updates. This fine-grained selection enhances computational efficiency and reduces redundancy in traditional finetuning while preserving performance. Despite its precision in parameter selection, GMT incurs additional computational overhead compared to supervised finetuning, potentially limiting its practicality in time-sensitive scenarios.

A further refinement in selective finetuning is proposed by *"On Surgical Fine-tuning for Language Encoders"* [47], which leverages the **Fisher Information Matrix (FIM)** to rank model layers based on their relevance to the downstream task. The method identifies and updates only layers with high FIM scores, thereby reducing training cost without significant loss in per-



formance. Empirical evaluations on benchmarks such as GLUE and SuperGLUE demonstrate that this method often matches or surpasses full-model finetuning, particularly for tasks requiring nuanced linguistic understanding. Nevertheless, performance degradation was observed in tasks like RTE and CB, and the approach still employed a constant subset of layers, without dynamic adjustment during training.

Memory efficiency is another critical consideration in LLM finetuning. "*SLIMFIT: Memory-Efficient Fine-Tuning of Transformer-based Models Using Training Dynamics*" [48] introduces a training-dynamics-based approach that addresses memory overhead by **dynamically freezing layers** with minimal contribution during training. The core mechanism, known as Inter-Layer Scheduling (ILS), tracks the magnitude of parameter updates across layers and iteratively freezes those with lower update magnitudes. In addition, SLIMFIT incorporates pruning and quantization techniques to further reduce static activation memory. Although this strategy lowers memory consumption and computational complexity, it lacks explicit runtime efficiency improvements and continues to rely on a fixed subset of layers without dynamic scheduling as training evolves.

Dynamic and adaptive layer selection becomes particularly important in distributed settings, such as Federated Learning (FL), where client-specific data heterogeneity and resource limitations add complexity to the finetuning process. The paper "*Exploring Selective Layer Fine-Tuning in Federated Learning*" [49] provides both theoretical and empirical analysis of layer selection in FL settings. It introduces a **client-specific dynamic layer selection strategy** that adapts to local data characteristics and constraints. This approach enhances model convergence and performance by considering gradient magnitudes and promoting cross-client consistency. However, the method does not implement a progressive mechanism to reduce the number of fine-tuned layers over time, which could further optimize resource utilization.

Finally, the study "*The Unreasonable Ineffectiveness of the Deeper Layers*" [50] offers a valuable perspective by empirically demonstrating that not all layers in large models contribute equally to downstream performance. The authors show that deeper layers often provide diminishing returns in task-specific finetuning. This insight strengthens the rationale for selectively finetuning only the most impactful layers, thus conserving computational resources while maintaining or even improving performance.

Collectively, these works highlight a growing consensus in the community: effective finetuning of LLMs can be achieved by selectively and adaptively updating only a subset of parameters or layers. Whether through gradient-based masking, information-theoretic criteria like the Fisher Matrix, or training-dynamics-informed freezing, the key lies in smart layer selection strategies that balance efficiency, scalability, and performance.

## 2.8 Summary

This literature review provides a comprehensive exploration of methods and advances in federated learning, personalization, OOD detection, test-time adaptation, and efficient finetuning of LLMs. In federated environments, data heterogeneity and privacy constraints necessitate adaptive training techniques. Advances in selective finetuning of LLMs—particularly those that reduce computational burden while maintaining accuracy—form the cornerstone for scalable deployment. Moreover, domain-specific benchmarks in sectors such as telecommunications

are essential to assess real-world performance. The convergence of these research directions underscores the importance of developing unified frameworks for efficient model training and deployment in decentralized, resource-constrained, and highly specialized environments.

## CHAPTER 3

### Methodology

#### 3.1 Introduction

This chapter presents the system design and methodology of two novel learning frameworks: (1) **Adaptive Split Federated Learning (ASFL)** for computer vision models (Like CNNs) and (2) **Efficient Fine-Tuning of Large Language Models (LLMs) using Dynamic Layer selection**. The first part introduces a multi-phase federated split learning system incorporating supervised contrastive learning, caching techniques, and adaptive OOD detection. The second presents an LLM-efficient training strategy leveraging gradient magnitudes, caching, and Taylor approximations.

#### 3.2 Adaptive Split Federated Learning (ASFL)

##### 3.2.1 System Model Architecture

The main components of the implemented architecture comprise of

- **Client:** Simulates an edge device, participating in the distributed training process, performing only local computations while retaining their private labels (i.e., data)
- **Offloading Server:** Responsible for the bulk of the computations, hosting the middle layers of the model.
- **Key-Value Store:** Responsible for caching activations of frozen layers, for all phases of training, thereby reducing communication and computation costs
- **Model Aggregation Server:** Responsible for aggregation of models between various clients and the server.

This architectural design enhances computational efficiency by dividing the pre-trained model into three components: Client Front, Center, and Client Back model.

- **Client Front:** This section encompasses the initial levels of the model and is located on the client device.
- **Server Center:** This component includes the intermediate layers and is located on the offloading server. It is responsible for performing computationally heavy activities.

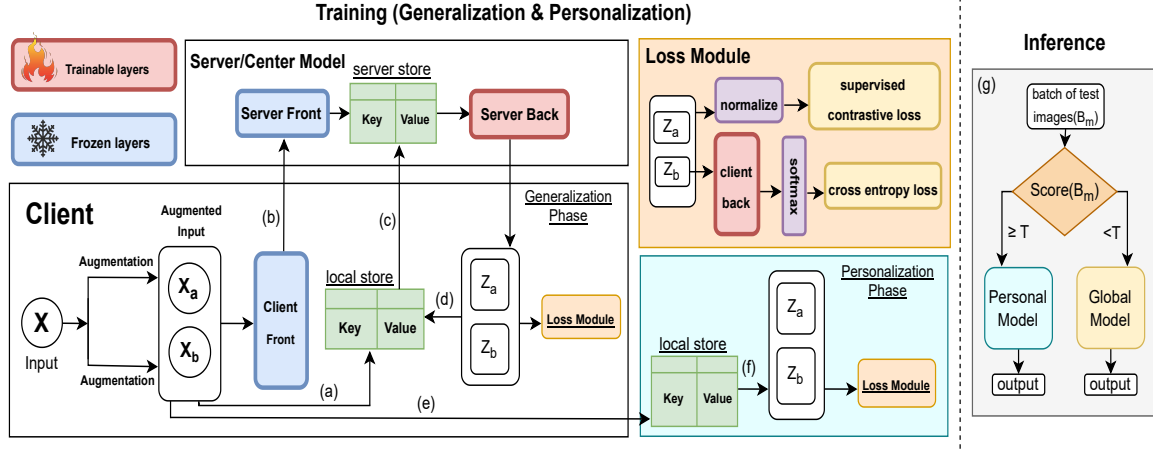


Figure 3.1: System Model Architecture for Robust Split Learning, highlighting the modules and communication and storage mechanisms at different phases

- **Client Back:** This portion encompasses the last few layers of the model and is located on the client’s device.

At the client side, raw inputs undergo augmentation to generate two distinct views ( $X_a$  and  $X_b$ ), each associated with a unique identification key. Varying augmentations are used for various datasets, depending upon the complexity of shifts each simulates, and the details on augmentations used are given in Table 3.1. These augmentations are done to support contrastive

Table 3.1: Dataset augmentations

Dataset	Augmentations
CIFAR10, FMNIST	RandomHorizontalFlip, Resize(224, 224), Normalize
VLCS, PACS, OfficeHome	Grayscale, RandomResizedCrop(224,224), RandomHorizontalFlip, ColorJitter, GaussianBlur, Normalize
ISIC2019	RandomScale, Rotate, RandomBrightnessContrast, Flip, Affine, Resize(224, 224), Normalize

training that makes model invariant to corruption by focusing on core class defining features. These views are first processed by a frozen client front model to extract feature representations, which are transmitted to the Center model (server). In the first epoch, all clients’ input data feature representations are forwarded and passed through a frozen server front model, and the resulting outputs are stored in a server-side key-value store. From the second epoch onward, only the keys are exchanged, significantly reducing communication overhead. The client maintains a local key-value store to retain the input keys.

The server processes stored values from its store through the server-back layers and sends them back to the Client Back model, which refines them for final classification. After the global model converges, generalization phase ends and the activations sent to the client-back model are stored as values in the local key-value store, facilitating an efficient personalization phase without requiring recomputation. The model operates in three key phases:

- **Generalization Phase** – Provides a global model optimized for all client distributions. We name this common model as **ASFL-G**
- **Personalization Phase** – Adapts the global model to specific client distributions for improved individual performance. This model is named as **ASFL-P**, which is unique to each client
- **OOD Detection Phase** – uses a gradient-based thresholding mechanism to differentiate between IID and OOD samples at runtime, allowing the system to dynamically select the appropriate model (global or personal) for inference. This adaptive inferencing approach is named as **ASFL-H/Hybrid**.

Loss computation is performed locally at the client. The Loss Module applies Supervised Contrastive Loss and Cross-Entropy Loss, incorporating softmax normalization for effective classification.

### 3.2.2 Loss Computation

The loss module and its components are illustrated in Figure 3.1. To achieve robust training, the loss function is defined as a combination of *Supervised Contrastive Loss* and *Cross Entropy Loss*. This combined loss function is expressed as:

$$\text{combined\_loss} = \text{ce\_loss} + \lambda_{\text{weight}} \cdot \text{sc\_loss}, \quad (3.1)$$

where  $\lambda_{\text{weight}}$  is a hyperparameter that balances the contribution of the supervised contrastive loss to the overall loss. The normalized activations of the two augmented views are passed through Supervised Contrastive loss, while only one unnormalized view is used for Cross Entropy loss. Combining these two losses serves a dual purpose: the contrastive loss enhances the quality of learned feature representations by improving the separability of embeddings, particularly for OOD samples, while the cross-entropy loss ensures high classification accuracy on in-distribution samples. Experimental results show a reduction in the ASFL-G and ASFL-P model performance on average after removing contrastive loss.

### 3.2.3 Training Phases

The training process is divided into three distinct phases: *Generalization*, *Personalization* and *GradientNorm OOD Threshold Calculation*. Each phase has a specific goal, ensuring optimal model performance across diverse scenarios.

#### 3.2.3.1 Generalization Phase:

The goal of the Generalization Phase is to train a global model that performs well across the entire data distribution. In this phase, each client independently trains its model using local data. After completing one pass over their data, the central layers are aggregated at the offloading server, while the client-specific back layers are averaged at the model aggregation server. This

iterative process continues until global convergence is achieved, at which point the training transitions to the Personalization Phase.

### **3.2.3.2 *Personalization Phase:***

The Personalization Phase aims to adapt the model to client-specific data by updating only the client back layers using the saved activations in local key-value store, without aggregation. This localized training allows for fine-tuning the model to individual needs while maintaining privacy. Once personalization is complete, the training progresses to the Thresholding Phase.

### **3.2.3.3 *GradientNorm OOD Threshold Calculation:***

In the final phase, we use a gradient-based thresholding mechanism to identify ID and OOD samples. This method relies on the gradient information from the personalized model. It compares the model's output with a uniform reference distribution. By measuring the gradient norm of this divergence, the approach sets a threshold adaptively. It does not change the model architecture or require extra training.

A key advantage of this technique is its adaptability. Instead of using a fixed threshold, the threshold is derived from the model's own gradient behavior. This makes it naturally tuned to client-specific data.

## **3.2.4 Key-Value Store**

ASFL utilizes pre-trained models and transfer learning to extract features, thereby accelerating convergence. During the *Generalization Phase*, the server-side key-value store is responsible for managing activation values and their associated unique keys per input. Since samples dynamically join batches in batch training, both the key and the corresponding activation linked with each sample are transmitted only during the initial epoch.

These activations are passed through the server-side *Center Front Model*. After processing, the resulting activations, along with their keys, are stored in the server-side key-value store. From the second epoch onward, this design eliminates the need to transmit the activations again; instead, only the unique keys per sample are exchanged with the server. This approach significantly reduces redundant computational work and communication overhead.

During the *Personalization Phase*, fine-tuning is confined to the client's *Back Model*. At this stage, caching the activations of each sample - originally computed by the server-side center model - into the client's local key-value store becomes possible. This mechanism removes the requirement for further communication between the server and the client, thus minimizing computational load and data exchange. Ultimately, this leads to enhanced overall system performance.

### 3.2.5 Inference

Batch-wise scores are computed:

$$\text{Score}(B_m) = \frac{1}{N} \sum_{n=1}^N \text{score}(x_{nm})$$

Where score is given as,

$$\text{score}(x) = \left\| \frac{\partial D_{KL}(u \parallel q)}{\partial \mathbf{w}} \right\|_1. \quad (3.2)$$

where,  $D_{KL}$  is the KL divergence between a uniform distribution and the softmax output

Then:

$$\text{Discriminator}(B_m) = \begin{cases} \text{ID} & \text{if } \text{Score}(B_m) \geq \mathbf{T} \\ \text{OOD} & \text{otherwise} \end{cases} \quad (3.3)$$

Where,  $\mathbf{T}$  is the threshold set by using the GradNorm. The system chooses ASFL-G or ASFL-P for inference accordingly.

## 3.3 Efficient Fine-Tuning for Large Language Models

In this project, we introduce efficient fine-tuning strategies for large language models (LLMs), with approaches tailored separately to encoder-decoder and decoder-only architectures. These methods leverage gradient-based layer freezing, LoRA-based parameter updates, and advanced computational optimizations including skip connections, activation caching and Taylor approximations.

### 3.3.1 Encoder-Decoder Architectures (e.g., RoBERTa, T5)

#### 3.3.1.1 System Design

Encoder-decoder models are widely used in tasks like classification and summarization. These architectures consist of a transformer-based encoder that processes the input sequence and a decoder that generates the output.

Our method focuses on dynamically freezing layers in both encoder and decoder based on their gradient magnitude during fine-tuning. This selective training reduces the compute footprint while maintaining model performance.

#### 3.3.1.2 Key Components

The key components of the implemented strategy include:

- **LoRA Integration:** Fine-tuning is performed using Low-Rank Adaptation (LoRA) modules, which introduce low-rank trainable parameters ( $A, B$ ) into selected layers:

$$W' = W + \Delta W = W + AB$$

where  $A \in \mathbb{R}^{d \times r}, B \in \mathbb{R}^{r \times k}$  and  $r \ll \min(d, k)$ .

- **Gradient-Based Layer Freezing:** At each training step or epoch, we compute the gradient norm for each LoRA layer. A lower percentage of gradient layers are marked as *frozen*.
- **Skip Connections for Frozen Layers:** Frozen layers are skipped during forward and backward passes using architectural bypasses (skip connections), effectively reducing both compute time.

### 3.3.1.3 Training Pipeline

The training pipeline is shown in Figure 3.2, where the steps are as follows:

- **Initialization:**
  - Inject **LoRA adapters** into **all trainable layers** of the base model.
  - **Freeze** all **non-LoRA** parameters.
  - **Train all LoRA layers** for the **first epoch**.
- **Dynamic Layer Selection (from Epoch 2 onward):**
  - At the **end of each epoch**, compute the **gradient norms** of all LoRA layers.
  - **Select the top 30%** of LoRA layers with the **highest gradient magnitudes**.
    - \* These selected layers **remain trainable** for the next epoch.
    - \* The remaining **70% of LoRA layers are frozen**.
    - \* The selected 30% **need not be contiguous**—they can come from any part of the model.
  - Repeat the selection and training process **after every epoch**.
  - Continue until:
    - \* **All training epochs** are completed, or
    - \* **Convergence** is achieved (e.g., via early stopping or validation loss plateau).



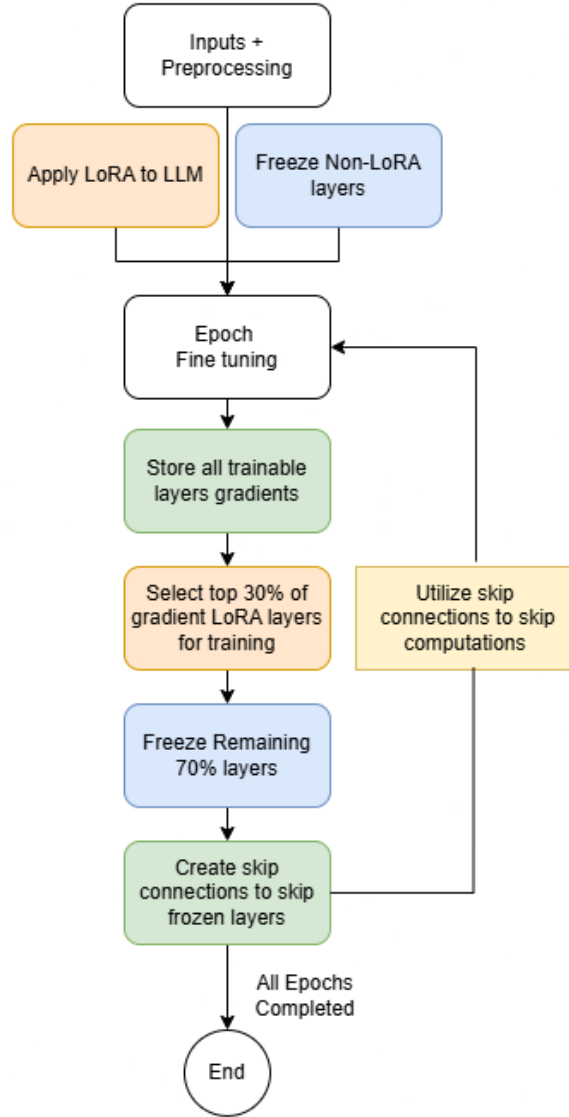


Figure 3.2: Dynamic Layer Selection for Encoder-Decoder LLM Architectures

### 3.3.2 Decoder-Only Architectures (e.g., LLaMA, GPT)

Decoder-only models, such as GPT and LLaMA, are widely used in generative tasks. These models are deeper, autoregressive, and operate over long contexts, making efficient fine-tuning particularly challenging.

To address this, we extend the layer freezing strategy with:

- **Activation caching for frozen layers:** To cache activations of known inputs for faster computations,
- **Weight offloading to host memory:** To reduce overall GPU usage required for fine-tuning such huge models,
- **First-order Taylor approximations:** to approximate LoRA updates of frozen LoRA layers over changing inputs.

### 3.3.2.1 Key Components

The key components (As shown in Figure 3.3) include:

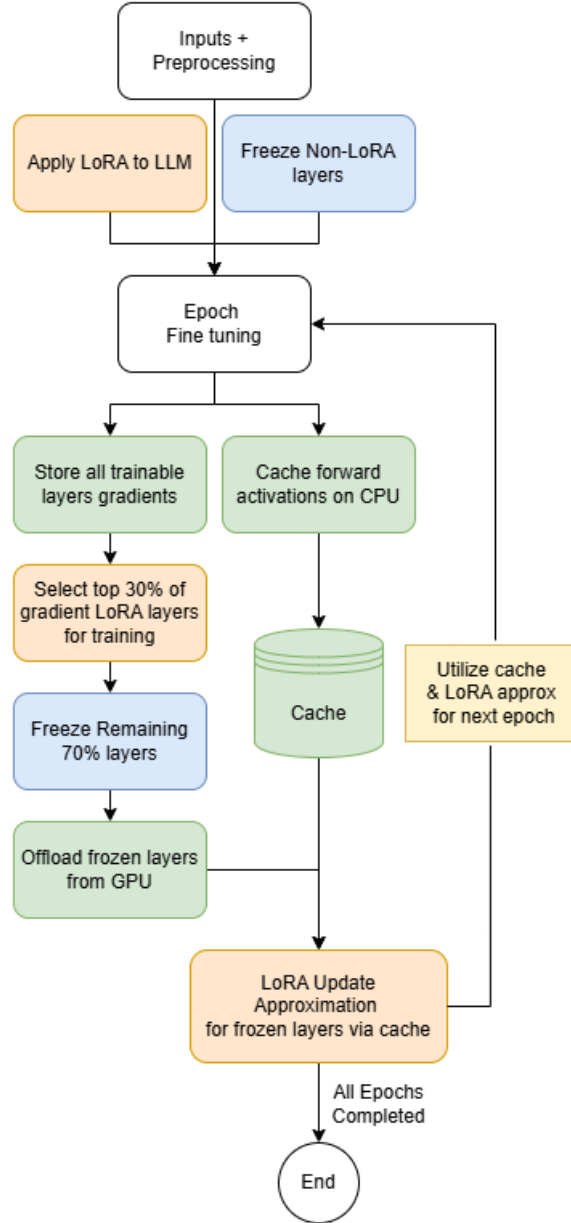


Figure 3.3: Dynamic Layer Selection for Encoder-Decoder LLM Architectures

- **Gradient-Based Freezing:** Identical to encoder-decoder setting.
- **Activation Caching:** For frozen layers, instead of recomputing outputs, we cache the layer's activations and reuse them across time steps or batches.
- **Taylor Series Approximation:** Since the inputs change, cached activations must be adjusted. For frozen layers, we apply:

$$f(W + \Delta W, x) \approx f(W, x) + \frac{\partial f}{\partial W} \cdot \Delta W$$

This approximation is exact or near-exact for linear projection layers (e.g., attention Q/K/V/O) and embeddings, and can be applied approximately to non-linear layers like SwiGLU via the chain rule.

- **Offloading:** Frozen weights and LoRA matrices are offloaded to CPU memory to reduce GPU memory consumption.

### 3.3.2.2 Taylor Approximation: Layer-Wise Summary

Layer Type	LoRA Applied?	Taylor Approximation	Remarks
Attention Q/K/V/O	Yes	First-order holds	Linear projection update
MLP (SwiGLU)	Yes	Chain-rule approx.	Nonlinear activation
Embedding	Yes	Exact linear match	No nonlinearity
Positional (Learned)	Yes	Linear update	Treated like embedding

### 3.3.3 Application to Attention Projection Layers (Q, K, V, O)

Transformer attention uses simple linear projections:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V, \quad O = AW_O,$$

where each mapping is just a matrix multiplication. When we apply LoRA to  $W_Q$ , we replace

$$W_Q \longrightarrow W'_Q = W_Q + AB,$$

so that

$$Q' = XW'_Q = X(W_Q + AB) = XW_Q + XAB = Q + \Delta Q.$$

In other words, the update  $\Delta Q$  is exactly  $XAB$ .

**First-Order Taylor Expansion:** Define

$$f(W_Q, x) = xW_Q.$$

Its derivative with respect to the weights is

$$\frac{\partial f}{\partial W_Q} = x.$$

Thus, the first-order Taylor approximation around  $W_Q$  gives

$$f(W_Q + \Delta W, x) \approx f(W_Q, x) + \frac{\partial f}{\partial W_Q} \Delta W = xW_Q + x\Delta W.$$

Substituting  $\Delta W = AB$  reproduces exactly the LoRA update:

$$f(W_Q + AB, x) \approx xW_Q + xAB.$$

Therefore, for any linear projection, LoRA’s rank-decomposition update is equivalent to taking one step of a Taylor expansion. Since the attention scores combine  $Q$  and  $K$  via

$$\text{Attention} = \text{softmax}(QK^\top / \sqrt{d_k}),$$

a shift  $\Delta Q = XAB$  simply perturbs the attention logits in the direction of the first-order gradient.

### 3.3.4 Application to the MLP Layer with SwiGLU Activation

The SwiGLU activation is defined as

$$\text{SwiGLU}(z_1, z_2) = \text{SiLU}(z_1) \cdot z_2, \quad \text{SiLU}(z) = z \sigma(z).$$

Let the input to the MLP be  $x$ , and let the first weight matrix be  $W_1$ . Then

$$[xW_1] = [z_1, z_2] \implies h = \text{SwiGLU}(z_1, z_2).$$

Applying LoRA gives  $W'_1 = W_1 + \Delta W_1$  with  $\Delta W_1 = AB$ , so

$$xW'_1 = x(W_1 + \Delta W_1) = xW_1 + x\Delta W_1 = z + \Delta z,$$

and the post-activation becomes

$$f(W'_1, x) = \text{SwiGLU}(z + \Delta z).$$

**First-Order Taylor Expansion for a Nonlinear Activation:** For a smooth function  $f$  at point  $z$ ,

$$f(z + \Delta z) \approx f(z) + J_f(z) \Delta z,$$

where  $J_f(z)$  is the Jacobian of  $f$  at  $z$ . In our case,

$$f(z) = \text{SwiGLU}(z_1, z_2), \quad J_{\text{SwiGLU}}(z) = [\partial \text{SwiGLU} / \partial z_1 \quad \partial \text{SwiGLU} / \partial z_2].$$

We compute

$$\frac{\partial \text{SwiGLU}}{\partial z_1} = \text{SiLU}'(z_1) z_2, \quad \frac{\partial \text{SwiGLU}}{\partial z_2} = \text{SiLU}(z_1).$$

Hence the change in the activation is

$$\Delta h \approx J_{\text{SwiGLU}}(z) \Delta z = \text{SiLU}'(z_1) z_2 \Delta z_1 + \text{SiLU}(z_1) \Delta z_2 \approx J_{\text{SwiGLU}}(z) xAB.$$

This shows that even for the nonlinear SwiGLU block, the LoRA update corresponds to a first-order (chain-rule) Taylor step in weight space.

**Conclusion:** Across both purely linear projections and more complex nonlinear blocks like SwiGLU, the low-rank LoRA updates precisely match the first-order Taylor approximation. This alignment demonstrates the theoretical feasibility of using Taylor expansions to understand, predict, and potentially guide LoRA updates in large language model layers.

### 3.4 Summary

Our approach integrates two key components - **Adaptive Split Federated Learning** and **Dynamic Layer selection for parameter-efficient finetuning** - to optimize training under resource constraints while preserving performance.

- **Adaptive Split Federated Learning:**

- Supervised Contrastive Loss is used to improve generalization and robustness to OOD samples
- Using caching, during generalization, only the **key** for each sample is transmitted, and the cache is utilized for retrieving the activations on the server, drastically reducing computational and communication cost.
- Similarly, for personalization, cached activations are reused locally, further minimizing client-server communication and reducing latency.

- **Dynamic Layer Selection for Finetuning:**

- LoRA adapters are added to all layers, with non-LoRA weights frozen for parameter-efficient training.
- In the first epoch, all LoRA layers are trained to initialize gradients.
- In subsequent epochs, only the **top 30% of LoRA layers** (by gradient magnitude) are retained for training, while the rest are frozen.
- This strategy reduces unnecessary updates and focuses computational resources on the most impactful parameters, improving convergence efficiency.
- The strategy can further be optimized using techniques like Skip Connections, LoRA update approximations, Layer offloading etc.

Together, these techniques form a cohesive methodology that prioritizes communication and computation efficiency, making scalable and personalized training feasible across resource-constrained real-world environments.

## CHAPTER 4

### Experimental Setup

#### 4.1 Introduction

This chapter outlines the experimental setup adopted to evaluate the contributions presented across the projects in this thesis. All experiments are designed to rigorously assess both performance and efficiency trade-offs in training and fine-tuning models across vision and language domains under resource-constrained settings. The experiments are conducted with a view toward real-world deployment constraints, such as computational efficiency, communication cost, and generalization under distributional shifts.

#### 4.2 Experimental Goals and Research Questions

The experimental design is guided by the following overarching goals:

- **G1:** Evaluate the performance and efficiency of the proposed Adaptive Split Federated Learning technique across diverse computer vision datasets with domain shift.
- **G2:** Assess the effectiveness of dynamic layer selection and reduction for fine-tuning large language models.

#### 4.3 Research Questions

##### 4.3.1 Adaptive Split Federated Learning

This project aims to enhance federated learning performance under heterogeneous data conditions. The following research questions are addressed:

- **ASFL-RQ1: Performance Comparison:** How do the proposed methods (*ASFL-G*, *ASFL-P*, *ASFL-Hybrid*) compare to existing algorithms in terms of test accuracy across datasets, under real-world type distributions and shifts across clients?
- **ASFL-RQ2: Client-wise Performance Gains in OOD Scenarios:** How does hybridization enhance the performance of individual clients, particularly in extreme scenarios?

- **ASFL-RQ3: Benefits of Contrastive Loss:** Does incorporating contrastive loss provide improvements in model performance?
- **ASFL-RQ4: Efficiency Gains:** How does the proposed method compare to existing algorithms in terms of computational cost and communication overhead?

#### 4.3.2 Dynamic Layer Selection in LLMs

This project targets efficiency in large-scale model fine-tuning. It investigates:

- **LLM-RQ1: Performance Comparison:** How does performance change when progressively freezing layers during training?
- **LLM-RQ2: Skip Connections Optimization:** What savings were achieved by using Skip Connections for optimization in encoder-decoder architectures?

### 4.4 Hardware and Software Environment

The experiments in this thesis were conducted across two distinct high-performance computing environments tailored for different project requirements.

For the **Federated Split Learning** experiments, a workstation equipped with an Intel Xeon Gold 5218 CPU running at 2.30 GHz and an NVIDIA RTX A6000 GPU was utilized. This setup provided sufficient computational capacity for training computer vision models in a federated setting, particularly under resource-constrained simulation.

All **LLM-based experiments**, including dynamic layer selection, were performed on a high-performance computing cluster comprising 56 physical processor cores and four NVIDIA Quadro RTX 6000 GPUs, each with 24 GB (23,040 MiB) of VRAM. The system was configured with NVIDIA System Management Interface (NVIDIA-SMI) version 550.127.08, driver version 550.127.08, and CUDA version 12.4. Each GPU had a maximum power draw of 250W, ensuring stable performance during extended training sessions.

This hardware configuration allowed for efficient execution of compute-intensive operations, particularly those involving large-scale model fine-tuning and inference. GPU acceleration was fully leveraged across all experiments, significantly reducing training time and improving throughput.

### 4.5 Datasets and models

#### 4.5.1 Computer Vision Datasets

To evaluate the Adaptive Split Federated Learning framework, we employed several benchmark datasets known for domain generalization and covariate shift characteristics. These datasets

span different application domains and exhibit varying degrees of visual domain shift, simulating real-world federated environments with heterogeneous data.

- **VLCS** [51]: Comprises data from four domains—VOC2007, LabelMe, Caltech101, and SUN09. Each domain has distinct data acquisition styles and object representations, creating covariate shifts while maintaining consistent label spaces. This dataset is particularly suitable for domain generalization under natural distribution gaps.
- **PACS** [52]: Includes four distinct visual styles—Photo, Art Painting, Cartoon, and Sketch—capturing extreme feature-level variation while preserving semantic class consistency. This diversity makes PACS an ideal testbed for evaluating cross-domain generalization.
- **OfficeHome** [53]: Contains 65 object categories across four domains: Art, Clipart, Product, and Real-World. The dataset introduces both label imbalance and domain-specific feature distortions, enabling realistic simulation of federated client heterogeneity.
- **ISIC 2019** [54]: A medical imaging dataset for skin lesion classification. Differences in imaging equipment and clinical settings introduce real-world domain shifts across samples. Used here to test generalization in critical, high-stakes domains.
- **FashionMNIST (FMNIST)** [55]: Contains grayscale images of clothing items from 10 categories. Often used as a simpler baseline for evaluating federated learning algorithms.
- **CIFAR10** [56]: A classic object classification dataset with natural image variability. Used primarily for performance benchmarking under federated data partitioning.

The combination of these datasets allows for comprehensive evaluation across synthetic and natural domain shifts, simulating federated settings where client data may be both IID (e.g., FMNIST, CIFAR10) and highly OOD (e.g., PACS, ISIC2019).

#### 4.5.2 Computer Vision Models

For all experiments under the Adaptive Split Federated Learning framework, we adopt a single backbone architecture, ResNet-18, across all baseline and proposed distributed learning methods. The reasons for choosing ResNet-18 include:

1. **Parameter Efficiency vs. Representation Quality:** ResNet-18’s initial layers is substantially lighter than VGG16 and GoogleNet, while still delivering richer feature embeddings than MobileNet. This balance reduces client-side computation and communication overhead without sacrificing representational capacity.
2. **Stable Deep Training via Residual Connections:** The residual (skip) connections in ResNet architectures facilitate direct gradient pathways, mitigating vanishing gradients and enabling robust optimization of deeper networks. By contrast, VGG and GoogleNet require significantly more parameters and can suffer from unstable gradients, while MobileNet’s heavy reliance on depthwise separable convolutions, though efficient, limits the expressiveness of learned features.



### 4.5.3 LLM Datasets

The evaluation of the Dynamic Layer Selection and Reduction strategy for large language models employed widely used sentence-level NLP benchmarks, selected for their linguistic diversity and task complexity:

- **SST-2 (Stanford Sentiment Treebank)** [57]: A binary sentiment classification dataset derived from movie reviews. It tests the model’s ability to understand nuanced sentiment cues and compositional semantics.
- **QQP (Quora Question Pairs)** [58]: This dataset involves identifying semantically equivalent questions. It challenges models to capture paraphrastic relationships and is sensitive to both syntactic and lexical variations.

### 4.5.4 LLM Models

To assess the effectiveness of our Dynamic Layer Selection method, we evaluate two widely adopted LLM architectures:

- **Encoder–Decoder (RoBERTa)**: RoBERTa’s strong bidirectional representations and masked-language pretraining make it a natural choice for tasks requiring deep contextual understanding. Its uniform layer structure allows us to measure how selectively updating subsets of layers impacts both efficiency and downstream performance.
- **Decoder-Only (Qwen-2.5)**: Qwen-2.5 exemplifies modern autoregressive transformers optimized for generation. By applying Dynamic Layer Selection to its decoding stack, we can evaluate how our approach preserves performance.

## 4.6 Experimental Design and Protocol

Each project follows a distinct but methodologically consistent experimental pipeline:

### 4.6.1 Adaptive Split Federated Learning

#### 4.6.1.1 Evaluation Settings

To thoroughly assess the proposed framework, we designed three experimental settings using benchmark datasets. These settings simulate practical scenarios with varying levels of distribution shift between training and test data, including both in-distribution (IID) and out-of-distribution (OOD) conditions:

1. **Setting-1: IID Test Scenario.** Clients are evaluated on test data that follows the same class distribution as their training data, with feature shifts in selected datasets. This setting serves as a baseline, highlighting the model’s core performance without significant distributional shifts.

2. **Setting-2: OOD Test Scenario.** Test data for each client is disjoint from their training distribution, again with feature shifts present in certain datasets. This strong setting measures the framework’s robustness under major shifts in data, a critical requirement for real-world deployments.
3. **Setting-3: Mixed IID/OOD Test Scenario.** Clients are tested on a mix of IID and OOD samples to simulate partial distributional shifts commonly encountered in practice. We define three sub-settings:
  - **3a: 20% OOD, 80% IID** — Simulates slight distributional drift where most test data resembles the training distribution.
  - **3b: 50% OOD, 50% IID** — Represents a balanced test environment with equal distributional exposure.
  - **3c: 80% OOD, 20% IID** — Models heavily shifted scenarios dominated by unfamiliar data.

These mixed settings provide nuanced insights into the model’s adaptability under gradual to severe domain shifts.

#### **4.6.1.2 Protocol**

Each client in the federated network receives data from a designated domain to simulate non-IID distributions. The split neural network architecture places lower layers on clients and upper layers on the server, with caching mechanisms to reduce communication overhead. Out-of-distribution (OOD) detection guides selective sample processing, while contrastive loss improves feature separation. Training runs for a fixed number of federated rounds and early stopping on validation loss, with local batch updates and periodic global aggregation.

### **4.6.2 Dynamic Layer Selection in Large Language Models**

#### **4.6.2.1 LLM Fine-tuning Settings**

Unlike the federated setting, the LLM experiments focus on centralized fine-tuning with the goal of reducing compute and memory requirements while maintaining model quality. The evaluation uses standard train-validation-test splits of SST-2 and QQP datasets, without explicit domain shifts, but emphasizes efficiency gains.

#### **4.6.2.2 Protocol**

The dynamic layer selection method gradually reduces the number of trainable layers over epochs. Key components include using a Layer Reduction Schedule, starting from full fine-tuning, layers are frozen progressively based on their gradients, reducing computational overhead.

## 4.7 Evaluation Metrics

### 4.7.1 Model Performance: Accuracy

Standard task-specific metrics were used for evaluation:

- **Classification Accuracy:** For vision and NLP classification tasks.
- **Loss:** For LLM convergence analysis

### 4.7.2 Efficiency Measurements

Efficiency, a core focus of this thesis, was measured using:

- **Communication Cost:** Amount of data transmitted between clients and server per round in federated settings.
- **Computation Cost:** FLOPs measured during training and inference phases.
- **Memory Usage:** Peak GPU memory consumed across different stages.
- **Training Time Reduction:** Comparison between full fine-tuning and dynamic/reduced training schemes.

## 4.8 Summary

This chapter established the experimental foundation for evaluating the proposed methodologies. Through careful dataset selection, task-specific design, and detailed evaluation metrics, the experiments aim to provide a comprehensive understanding of the trade-offs between performance and efficiency. The next chapter will delve into the experimental results, highlighting how the proposed strategies advance the state of efficient model training and fine-tuning in both federated and centralized paradigms.

## CHAPTER 5

### Results and Discussion

#### 5.1 Introduction

This chapter presents the empirical evaluation outcomes of the proposed methodologies described in previous chapters. The results are organized according to the two primary research projects underpinning this thesis: Adaptive Split Federated Learning and Dynamic Layer Selection for Large Language Models.

The first part focuses on the federated learning framework, examining its performance and efficiency across diverse domain generalization benchmarks under varying degrees of data distribution shifts. We analyze the impact of the proposed caching strategies, contrastive loss incorporation, and out-of-distribution detection mechanisms on classification accuracy, communication overhead, and computational cost. The experimental settings include in-distribution (IID), out-of-distribution (OOD), and mixed scenarios, providing a comprehensive view of the framework’s robustness and scalability in realistic federated environments.

Following this, we turn to the results of dynamic layer selection in LLM fine-tuning. This section evaluates the effectiveness of progressively reducing trainable layers combined with approximation and caching mechanisms on widely-used natural language understanding benchmarks. We assess trade-offs between training efficiency—in terms of computational resources and memory usage—and task performance, highlighting how the proposed method maintains accuracy while significantly reducing fine-tuning costs.

Together, these results demonstrate the feasibility and advantages of efficient training and fine-tuning strategies for both vision models in federated learning contexts and large-scale language models, advancing state-of-the-art practices in resource-constrained model adaptation.

#### 5.2 Results for Federated Split Learning

In this section, we analyze and discuss the performance of our proposed methods (*ASFL-G*, *ASFL-P*, and *ASFL-Hybrid*) relative to several baseline FL algorithms. We address two key questions: (i) How do the proposed methods compare to existing algorithms under different test conditions, and (ii) how does the hybridization of global and personalized approaches influence the trade-off between generalization and personalization? The discussion is organized by the test scenario settings. In all the test accuracy tables, we provided the best test accuracy values for PACS and VLCS.

### 5.2.1 ASFL-Q1: Performance Comparisons

#### 5.2.1.1 Setting-1: IID Test Scenario

In Setting-1, where clients receive test data that is entirely IID with respect to their training distribution, personalization appears to be highly beneficial. As shown in Table 5.1, the personalized approach (**ASFL-P**) achieves the highest average test accuracy of **83.87%**, outperforming both the global approach (*ASFL-G*, 79.06%) and the hybrid approach (*ASFL-Hybrid*, 82.52%). These results suggest that when the test data mirrors the training distribution, leveraging client-specific adaptations (as in ASFL-P) is most effective.

Algorithm	CIFAR10	FMNIST	ISIC2019	OFFICE-HOME	VLCS	PACS	Average
FedAvg	56.6 $\pm$ 4.3	84.1 $\pm$ 3	37.61 $\pm$ 4.9	4.32 $\pm$ 2.3	35.67	32.50	41.79
FedAvg-TL	87 $\pm$ 1.5	89.8 $\pm$ 0.9	55.49 $\pm$ 1.8	62.57 $\pm$ 8.7	71.04	73.44	73.22
FedProx	55.56 $\pm$ 3.2	83.5 $\pm$ 2.2	39.17 $\pm$ 4.1	10.79 $\pm$ 2.5	55.59	24.69	44.88
FedBN	42.8 $\pm$ 7.6	86.5 $\pm$ 3.4	35.62 $\pm$ 2.3	14.65	58.34	24.07	43.66
SL	67.1 $\pm$ 2.7	83.41 $\pm$ 1.9	39.72	9.45 $\pm$ 1.6	54.49	43.56	49.62
Ditto	73.5 $\pm$ 1.1	89.7 $\pm$ 0.7	51.06 $\pm$ 6.8	16.57 $\pm$ 3.4	65.45	48.65	57.48
FedIIR	87.2 $\pm$ 4	88.26 $\pm$ 2.2	47.21	74.74 $\pm$ 1.59	40.14	47.88	64.23
ASFL-G	87.4 $\pm$ 1	87.5 $\pm$ 1	62.85	73.23 $\pm$ 1.7	73.54	89.84	79.06
ASFL-P	<b>90.7 <math>\pm</math> 1</b>	<b>92.39 <math>\pm</math> 0.5</b>	<b>66.47</b>	<b>80.80 <math>\pm</math> 1.9</b>	81.14	<b>91.74</b>	<b>83.87</b>
ASFL-Hybrid	89.2 $\pm$ 1.2	90.3 $\pm$ 0.8	66.33	75.28	<b>83.60</b>	90.44	82.52

Table 5.1: Client test accuracies in Setting-1 (IID test)

#### 5.2.1.2 Setting-2: OOD Test Scenario

In stark contrast, Setting-2 involves clients receiving test data that is completely out-of-distribution relative to their training data. Here, the need for robust generalization is paramount. Table 5.2 indicates that the global method (**ASFL-G**) achieves superior performance with an average accuracy of **78.51%**. In this scenario, the personalized model (*ASFL-P*) suffers a significant performance drop (60.67% average accuracy), highlighting its limitation in adapting to unseen, shifted distributions. The global approach, by contrast, is better at capturing domain-invariant features necessary for generalization.

#### 5.2.1.3 Setting-3: Mixed Test Scenario

In real-world FL deployments, the true nature of the test distribution is rarely known *a priori*. In practice, clients may encounter a mixture of ID and OOD samples. This uncertainty motivates the need for an approach that can offer robust performance without relying on prior knowledge of the test data characteristics. Our *ASFL-Hybrid* method, which seamlessly integrates global and personalized components, is designed to address this challenge.

The experimental results in Setting-3 (see Tables 5.3, 5.4, and 5.5) reveal that while neither the purely personalized (*ASFL-P*) nor the purely global (*ASFL-G*) method uniformly dominates,

Algorithm	CIFAR10	FMNIST	ISIC2019	OFFICE-HOME	VLCS	PACS	Average
FedAvg	56.6 $\pm$ 4.3	84.04 $\pm$ 3.1	37.61 $\pm$ 4.9	4.32 $\pm$ 2.3	35.67	32.50	41.79
FedAvg_TL	87.1 $\pm$ 1.4	<b>90.02 <math>\pm</math> 1.1</b>	56.56 $\pm$ 1.6	62.58 $\pm$ 8.7	71.04	73.44	73.45
FedProx	55.5 $\pm$ 3.2	83.7 $\pm$ 2.1	38.67 $\pm$ 4	4.2	22.31	12.89	36.21
FedBN	40.6 $\pm$ 10.3	76.5 $\pm$ 4.9	28.46 $\pm$ 2.4	4.57	23.86	15.72	36.88
SL	40 $\pm$ 5.3	62.12 $\pm$ 6.4	29.83	4.64	28.82	24.88	36.73
Ditto	47.3 $\pm$ 5.9	75.05 $\pm$ 2.9	39.95 $\pm$ 4.5	2.04	15.10	11.35	31.79
FedIIR	87.2 $\pm$ 4	88.26 $\pm$ 2.2	46.84	73.5 $\pm$ 2.5	42.22	47.88	64.31
ASFL-G	<b>87.4 <math>\pm</math> 1</b>	86.96 $\pm$ 1.7	<b>62.85</b>	72.88 $\pm$ 2.2	<b>71.72</b>	<b>89.26</b>	<b>78.51</b>
ASFL-P	69.3 $\pm$ 7.8	71.09 $\pm$ 1	51.65	62.82 $\pm$ 4.3	50.82	58.37	60.67
ASFL-Hybrid	84.8 $\pm$ 3.3	85.97 $\pm$ 0.95	53.98	<b>74.77</b>	71.34	80.59	75.24

Table 5.2: Client test accuracies in Setting-2 (OOD Test )

Algorithm	CIFAR10	FMNIST	ISIC2019	OFFICE-HOME	VLCS	PACS	Average
FedAvg	56.1 $\pm$ 4.3	84.10 $\pm$ 2.7	32.75 $\pm$ 4.2	4.36 $\pm$ 2.3	40.18	32.29	41.63
FedAvg_TL	86.9 $\pm$ 1.3	<b>89.58 <math>\pm</math> 0.9</b>	51.20	62.2 $\pm$ 8.9	70.49	74.10	72.37
FedProx	55.3 $\pm$ 3.6	83.03	33.61 $\pm$ 3.5	9.77 $\pm$ 3.1	59.12	22.32	43.85
FedBN	42.6 $\pm$ 7.7	85.63 $\pm$ 2.1	29.33 $\pm$ 1.9	13.46	48.17	25.01	40.70
SL	61.8 $\pm$ 2.4	79.88 $\pm$ 1.3	34.91	8.89 $\pm$ 1.3	52.65	37.28	45.90
Ditto	69 $\pm$ 0.6	87.16 $\pm$ 0.8	37.85	13.50 $\pm$ 1.9	59.27	44.30	51.80
FedIIR	86.96 $\pm$ 3.8	88.31 $\pm$ 2.2	41.74	73.15 $\pm$ 1.9	46.27	48.67	64.18
ASFL-G	84.54 $\pm$ 1.6	87.33 $\pm$ 1.4	<b>62.85</b>	70.97 $\pm$ 1.6	71.64	<b>90.36</b>	77.94
ASFL-P	<b>85.59 <math>\pm</math> 1.6</b>	88.2 $\pm$ 1.3	61.13	<b>76.45 <math>\pm</math> 1.5</b>	<b>80.95</b>	89.25	<b>80.20</b>
ASFL-Hybrid	84.63 $\pm$ 0.9	88.07 $\pm$ 1.1	62.01	73.29	73.07	89.85	78.51

Table 5.3: Client test accuracy in Setting-3a (20%-OOD-80%-IID)

Algorithm	CIFAR10	FMNIST	ISIC2019	OFFICE-HOME	VLCS	PACS	Average
FedAvg	56.15 $\pm$ 4.3	83.99 $\pm$ 2.6	30.10 $\pm$ 3.6	4.43 $\pm$ 2.3	40.61	32.65	41.32
FedAvg_TL	87.1 $\pm$ 1.8	89.28 $\pm$ 1	39.74	63.1 $\pm$ 9.0	71.21	73.60	70.67
FedProx	55.1 $\pm$ 3.3	83.90 $\pm$ 2.1	32.23 $\pm$ 2.6	7.3 $\pm$ 4.2	47.73	23.13	41.56
FedBN	42.3 $\pm$ 7.2	82.3 $\pm$ 2	27.96 $\pm$ 2	10.56	43.94	22.23	38.21
SL	54 $\pm$ 3.3	75.33 $\pm$ 3.7	31.29	7.41 $\pm$ 0.9	43.77	33.34	40.85
Ditto	62.0 $\pm$ 3.2	82.69 $\pm$ 1.5	42.33 $\pm$ 1.3	9.68 $\pm$ 1.6	47.95	35.68	46.72
FedIIR	87.1 $\pm$ 3.8	88.4 $\pm$ 2.4	42.67	<b>73.38 <math>\pm</math> 0.8</b>	47.09	49.00	64.60
ASFL-G	<b>84.6 <math>\pm</math> 0.7</b>	<b>87.54 <math>\pm</math> 1.7</b>	<b>62.85</b>	73.33	<b>74.03</b>	<b>90.69</b>	<b>78.84</b>
ASFL-P	80.3 $\pm$ 3.7	81.4 $\pm$ 2.5	58.84	69.77 $\pm$ 3.6	71.93	78.31	73.42
ASFL-Hybrid	84.5 $\pm$ 0.7	86.74 $\pm$ 1.8	60.23	73.07	72.93	88.52	77.66

Table 5.4: Client test accuracy in Setting-3b (50%-OOD-50%-IID)

Algorithm	CIFAR10	FMNIST	ISIC2019	OFFICE-HOME	VLCS	PACS	Average
FedAvg	56.1 $\pm$ 4.3	84.24 $\pm$ 2.6	31.32 $\pm$ 4.8	4.38 $\pm$ 2.3	41.29	31.86	41.53
FedAvg_TL	86.8 $\pm$ 1.5	89.99 $\pm$ 1	46.28 $\pm$ 2.2	62.07 $\pm$ 9	<b>73.44</b>	75.21	72.29
FedProx	55.2 $\pm$ 3	83.77 $\pm$ 2.3	31.60 $\pm$ 4.3	4.96 $\pm$ 5.1	34.07	22.77	38.72
FedBN	42.4 $\pm$ 7.3	79.38 $\pm$ 2.9	27.55 $\pm$ 2.8	7.61	39.37	24.14	36.74
SL	46.1 $\pm$ 4.9	67.65 $\pm$ 1.8	30.22	5.62 $\pm$ 0.8	47.04	25.41	37.00
Ditto	54.8 $\pm$ 5.7	78.46 $\pm$ 2.2	40.05 $\pm$ 2.3	5.6 $\pm$ 1.1	41.68	26.18	41.12
FedIIR	<b>87.2 <math>\pm</math> 3.9</b>	88.22 $\pm$ 1.9	43.96	71.29 $\pm$ 0.2	43.73	49.42	63.97
ASFL-G	84.5 $\pm$ 0.8	<b>87.34 <math>\pm</math> 1.4</b>	<b>62.85</b>	72.72	71.83	<b>90.11</b>	<b>78.22</b>
ASFL-P	75.2 $\pm$ 5.5	74.63 $\pm$ 3.4	54.31	62.35	56.82	70.19	65.58
ASFL-Hybrid	84.2 $\pm$ 0.9	85.55 $\pm$ 2.2	55.33	<b>72.81</b>	68.61	87.04	75.59

Table 5.5: Client test accuracy in Setting-3c (80%-OOD-20%-IID)

the hybrid approach consistently achieves balanced performance across subsettings with varying IID–OOD ratios, where the key insights include: **Setting-3a (80% IID – 20% OOD):** Although Table 5.3 shows that *ASFL-P* records the highest average accuracy (80.20%), the differences among the methods are marginal, and both *ASFL-G* (77.94%) and *ASFL-Hybrid* (78.51%) remain competitive. **Setting-3b (50% IID – 50% OOD):** In Table 5.4, the global approach (*ASFL-G*) slightly outperforms the hybrid method (78.84% vs. 77.66% average accuracy), while *ASFL-P* lags behind (73.42%). **Setting-3c (20% IID – 80% OOD):** As observed in Table 5.5, both *ASFL-G* and *ASFL-Hybrid* demonstrate robust performance (78.22% and 75.92% respectively), whereas the personalized method (*ASFL-P*) suffers, achieving only 65.42% average accuracy.

This balanced performance is critical in scenarios where the test data distribution is unpredictable. Rather than optimizing exclusively for either ID or OOD, *ASFL-Hybrid* mitigates the risk associated with an incorrect assumption about the test environment.

#### 5.2.1.4 Overall Performance Comparison

An examination of the average test accuracies across all settings reveals a clear trend: In IID scenarios (Setting-1), personalization (*ASFL-P*) provides the best results. In OOD scenarios (Setting-2), the global model (*ASFL-G*) is more robust. In mixed scenarios (Setting-3), while *ASFL-P* and *ASFL-G* occasionally outperform one another depending on the exact IID/OOD ratio, the *ASFL-Hybrid* method offers a balanced performance.

These findings are reinforced by the overall averages reported in each table, which collectively demonstrate that the proposed methods outperform the baselines (e.g., FedAvg, FedProx, FedBN, SL, Ditto, FedIIR) under various conditions. The average column in the tables is particularly indicative of the proposed methods’ competitiveness across diverse datasets and test distributions.

#### 5.2.2 ASFL-Q2: Client-wise performance gains in OOD Scenarios

We evaluate client-wise performance under a complete OOD scenario on the CIFAR-10 dataset. Figure 5.1 show the accuracies for three variants—*ASFL-General*, *ASFL-Hybrid*, and *ASFL-Personal*

across ten clients. The results indicate that *ASFL\_Personal* consistently underperforms in OOD

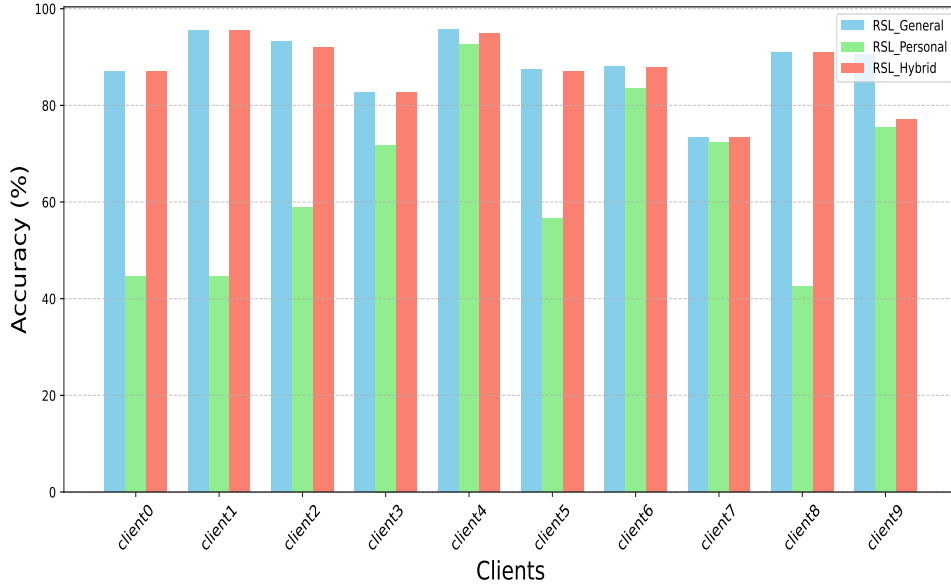


Figure 5.1: Hybridization benefit in clients under OOD scenarios for CIFAR-10 Dataset.

conditions, while both *ASFL\_General* and *ASFL\_Hybrid* achieve significantly higher accuracies. For instance, Clients 0 and 1 record around 87–95% accuracy with global/hybrid models, compared to roughly 44% for the personalized variant. Similar trends are observed for other clients, although some (e.g., Client4) exhibit smaller differences, likely due to local data characteristics. In summary, these findings underscore that leveraging global or hybrid strategies improves test-time generalization in OOD scenarios, benefiting all clients.

### 5.2.3 ASFL-Q3: Benefits of Contrastive Loss Strategy

To evaluate the impact of contrastive loss, we compare the performance of ASFL-G and ASFL-P with and without this component on Setting 3c (80% OOD – 20% IID). As shown in Table 5.6, variants trained without contrastive loss are marked with \*. The results indicate consistent improvements across most datasets when contrastive loss is included, proving that contrastive loss enhances representation learning, contributing to improved generalization and personalization.

Algorithm	ISIC2019	OfficeHome	VLCS	PACS
ASFL-G	62.8	72.7	71.8	90.1
ASFL-G*	59.4	63.5	68.5	83.1
ASFL-P	54.3	62.3	56.8	70.2
ASFL-P*	54.3	57.6	50.5	82.6

Table 5.6: Accuracy of ASFL-G and ASFL-P with/without contrastive loss. Runs without contrastive loss are denoted by \*.



## 5.2.4 ASFL-Q4: Efficiency Gains

### 5.2.4.1 Communication Efficiency

Table 5.7 summarizes the communication overhead across multiple benchmark datasets. The key observations are as follows: **Simplicity and Rapid Convergence:** On simpler datasets such as FMNIST and CIFAR10, ASFL converges in only 6 epochs, with a total data transfer of just 0.75 GB in each case. This is a dramatic reduction compared to traditional methods like FedAvg and FedProx, which require more than 20 epochs and transfer over 70 GB. **Efficiency on Complex Datasets:** For datasets with additional complexity, such as OFFICEHOME, VLCS, and PACS, ASFL maintains a minimal communication footprint (e.g., 21 epochs and 0.63 GB on OFFICEHOME, 6 epochs and 0.21 GB on VLCS, and 14 epochs and 0.62 GB on PACS). Although on ISIC2019, ASFL requires 4 epochs with a total of 14.25 GB—slightly higher than some alternatives—the overall trend demonstrates substantially lower communication requirements.

These results indicate that the proposed method can significantly reduce the communication overhead, which is crucial for FL environments with constrained bandwidth.

Algorithm	FMNIST	CIFAR10	ISIC2019	OFFICEHOME	VLCS	PACS
FedAvg	2.63	3.09	0.10	2.76	9.41	6.01
FedProx	3.89	6.88	0.10	4.41	1.22	2.80
FedBN	2.63	1.03	0.15	2.34	1.22	2.00
FedIIR	7.68	0.22	0.01	5.10	17.19	20.43
Ditto	4.12	5.96	0.48	3.17	2.45	1.60
SL	17.81	4.94	5.96	7.14	0.49	6.29

Table 5.7: Number of Times Increase in Communication Overhead Compared to Our Method (ASFL)

### 5.2.4.2 Computational Efficiency

The computational statistics are examined in Table 5.10, which draws attention to the number of times reduction in floating-point operations performed at the client end for different approaches on all datasets. Since we do not have to recompute the same activation when using the pretrained model, Our results requires 70.36 times less calculation on average compared to standard FedAvg. The detailed calculation of communication and computation is available in Table 5.8 and 5.9.

Our Split system aligns with the design in [59], which demonstrated scalability, supporting our claim that the proposed architecture is highly scalable and efficiently accommodates a large number of clients. By minimizing data transfer and reducing computational overhead, our framework optimizes communication efficiency, allowing seamless client participation without imposing excessive resource demands. In real-world scenarios, where numerous clients train on limited labeled data, challenges such as client dropout—where some clients initiate training but fail to complete all phases—can impact model convergence. However, our framework is robust to such disruptions, ensuring that meaningful model updates are consistently achieved, even with variable client participation.

Table 5.8: Communication overhead (GB) of various algorithms on benchmark datasets till convergence

algorithm	fmnist		cifar10		isic2019		officehome		vlcs		pacs	
	epoch	total data	epoch	total data	epoch	total data	epoch	total data	epoch	total data	epoch	total data
FedAvg	23	<u>2.00</u>	27	2.35	18	<u>1.56</u>	20	1.74	23	2.00	15	1.30
FedProx	34	2.96	60	5.22	18	1.56	32	2.78	3	0.26	7	0.60
FedBN	23	2.00	9	0.78	26	2.26	17	<u>1.48</u>	3	0.26	5	0.43
FedIIR	67	5.83	2	<b>0.17</b>	3	<b>0.26</b>	37	3.22	42	3.65	51	4.44
Ditto	36	3.13	52	4.52	79	6.88	23	2.00	6	0.52	4	<u>0.34</u>
SL	36	13.52	10	3.75	12	85.09	15	4.50	1	<b>0.10</b>	13	1.36
ASFL	6	<b>0.75</b>	6	<u>0.75</u>	4	14.25	21	<b>0.63</b>	6	<u>0.21</u>	14	<b>0.21</b>

Table 5.9: Floating-point operations at the client in various Algorithms till convergence

Algorithm	fmnist		cifar10		isic2019		officehome		vlcs		pacs	
	epoch	total Gflops	epoch	total Gflops	epoch	total Gflops	epoch	total Gflops	epoch	total Gflops	epoch	total Gflops
FedAvg	23	251.6	27	295.4	18	196.9	20	218.8	23	251.6	15	164.1
FedProx	34	371.9	60	656.4	18	196.9	32	350.1	3	32.8	7	76.5
FedBN	23	251.6	9	98.4	26	284.4	17	185.9	3	32.8	5	54.7
FedIIR	67	733.1	2	21.8	3	21.8	37	404.8	42	459.5	51	557.9
Ditto	36	787.7	52	1137.8	79	1728.7	23	503.2	6	131.2	4	87.5
SL	36	<u>26.1</u>	10	<u>7.2</u>	12	<u>8.7</u>	15	<u>10.9</u>	1	<b>0.7</b>	13	<u>9.4</u>
ASFL	6	<b>2.9</b>	6	<b>2.9</b>	4	<b>1.9</b>	21	<b>10.1</b>	6	<u>2.9</u>	14	<b>6.7</b>

### 5.3 Results for Efficient Fine-tuning of LLMs

This section reports quantitative and qualitative results from experiments on dynamic layer selection.

#### 5.3.1 LLM-RQ1: Performance Comparison

In the domain of parameter-efficient fine-tuning of large language models (LLMs), our experiments explored various LoRA (Low-Rank Adaptation) configurations to balance performance and computational efficiency. Among these, the strategy that combines **LoRA layer selection** with a **momentum-based gradient optimization** mechanism demonstrated notable advantages.

**Momentum Gradient Integration:** To enhance convergence, we incorporated a momentum-based gradient strategy during fine-tuning. This helps in smoothing noisy updates, accelerating convergence, and avoiding sharp local minima—especially in low-rank adaptation contexts where updates are limited in scope.

**Empirical Observations:** As illustrated in Figure 5.2, the **Layer Selection and Reduction** strategy achieves a convergence curve that closely follows the performance of full *All-Layer LoRA* training, while requiring significantly fewer trainable parameters and compute resources. Furthermore, it reaches near-optimal accuracy faster, demonstrating superior training efficiency.

Algorithm	FMNIST	CIFAR10	ISIC2019	OFFICEHOME	VLCS	PACS
FedAvg	86.58	101.62	101.65	21.52	86.61	24.21
FedProx	127.98	225.82	101.82	34.44	11.29	11.30
FedBN	86.58	33.87	146.82	18.29	11.29	8.07
FedIIR	252.21	7.52	11.29	39.82	158.17	82.33
Ditto	271.03	391.43	892.27	49.50	45.19	12.91
SL	9.00	2.50	4.50	1.07	0.25	1.39

Table 5.10: Number of Times Increase in Computation Overhead Compared to Our Method (ASFL)

**Comparative Analysis:** To evaluate the effectiveness of various LoRA-based fine-tuning strategies, we conducted experiments on two distinct datasets: QQP (a pairwise sentence classification task) and SST-2 (a sentiment analysis task). Across both datasets, results consistently show that our proposed method—Layer Selection combined with Reduction—offers the best trade-off between:

- **Final accuracy** (comparable to full-layer training),
- **Memory and compute efficiency** (significantly fewer parameters updated).

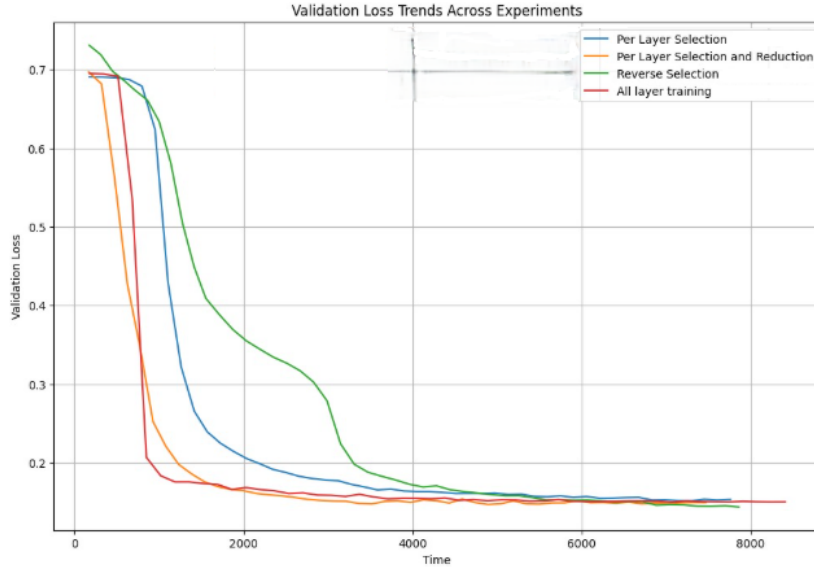


Figure 5.2: Validation loss over time on the QQP dataset across different LoRA strategies. “Layer Selection + Reduction” (orange) achieves a strong balance between convergence speed and final loss, converging at approximately 3211.68 seconds. This is close to full-layer training (red), which converges fastest at 2870.48 seconds. “Per Layer Selection” (blue) follows at 3585.39 seconds, while “Reverse Selection” (green) lags significantly at 5528.22 seconds.

As shown in Figure 5.2, selective fine-tuning strategies, especially those using targeted layer adaptation, tend to converge more slowly than full-layer training. This is expected, as restricting updates to only a subset of layers reduces parameter flexibility. While full-layer training can quickly leverage the entire model capacity, selective methods must learn to propagate meaningful gradients through fewer trainable parameters. This increases the time to convergence but dramatically reduces computational overhead.

Interestingly, this pattern holds across different datasets. Figure 5.3 shows evaluation loss on the SST-2 dataset, where all strategies again reach similar final loss values. Full-layer training and “Layer Selection + Reduction” converge quickly, while “Per Layer Selection” and “Reverse Selection” are slightly slower in early epochs. Nevertheless, all methods stabilize to comparable performance. This is further validated by using validation accuracy, which is depicted in Figure 5.4

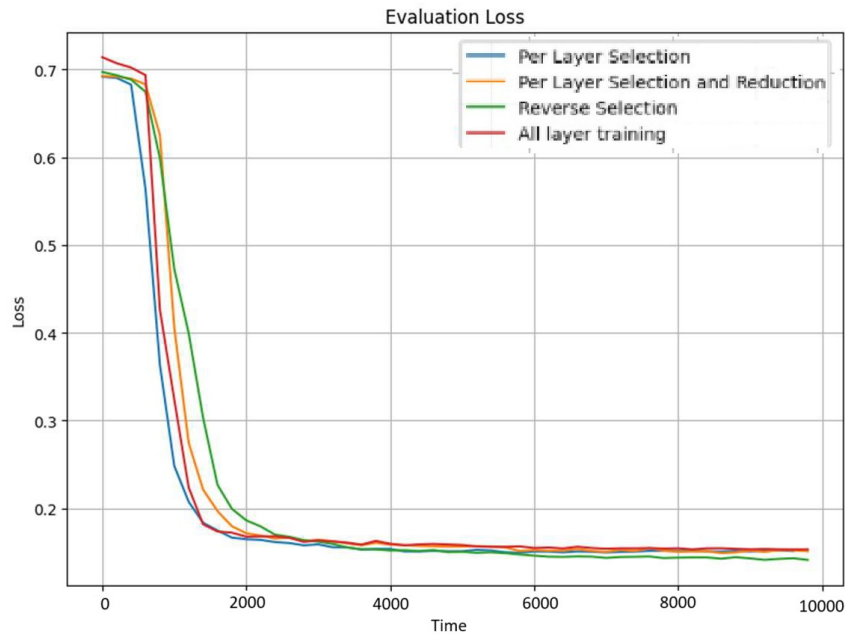


Figure 5.3: Evaluation loss over epochs on the SST-2 dataset across different LoRA strategies. All methods reach similar final loss, with full-layer training (red) converging fastest, followed closely by “Layer Selection + Reduction” (orange). “Per Layer Selection” (blue) and “Reverse Selection” (green) converge more slowly, though differences diminish after epoch 10.

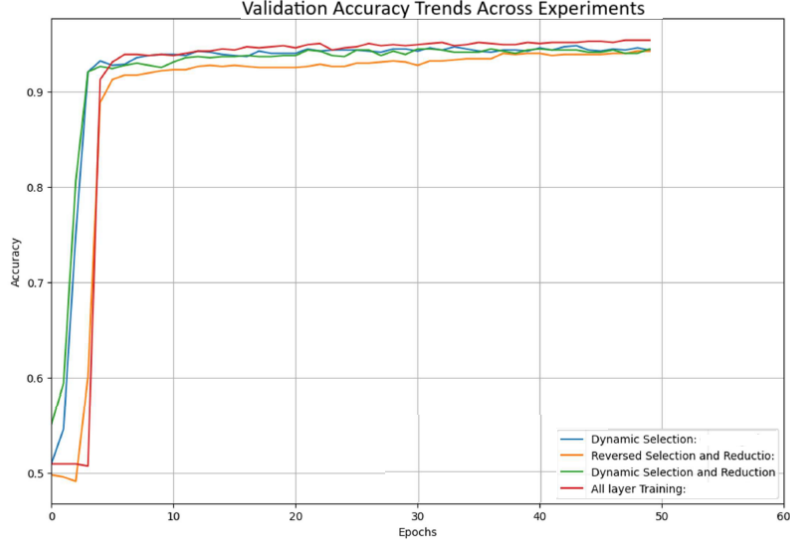


Figure 5.4: Validation accuracy over epochs on the SST-2 dataset across different LoRA strategies. All methods reach similar final accuracy, with full-layer training (red) converging fastest, followed closely by “Layer Selection + Reduction” (green). “Per Layer Selection” (blue) and “Reverse Selection” (orange) converge more slowly

This trend is further substantiated in Figure 5.5, which presents the training loss progression of the Qwen 2.5 model, a decoder-only architecture, on the SQuAD dataset using the proposed layer selection strategy. The graph displays both raw and smoothed loss values over training steps, with vertical markers denoting the ends of the first and second epochs. As seen in the plot, the training loss undergoes a significant drop during the first epoch and continues to decline steadily throughout the second. This consistent downward trend demonstrates stable convergence behavior even in the presence of constrained fine-tuning.

What makes this result especially compelling is that it validates the applicability of the approach beyond the encoder-decoder paradigm. Decoder-only models, such as Qwen 2.5, are essential in many high-performance language modeling tasks. Achieving stable and efficient convergence in such architectures using only a subset of tunable parameters illustrates the adaptability of the method.

Additionally, the use of the SQuAD dataset, a well-established benchmark for extractive question answering, introduces a task-level contrast to prior experiments conducted on classification tasks such as QQP and SST2. Despite these task-level differences the proposed layer selection strategy maintains robust convergence dynamics across the board. This cross-task consistency reinforces the task-agnostic nature of the method.

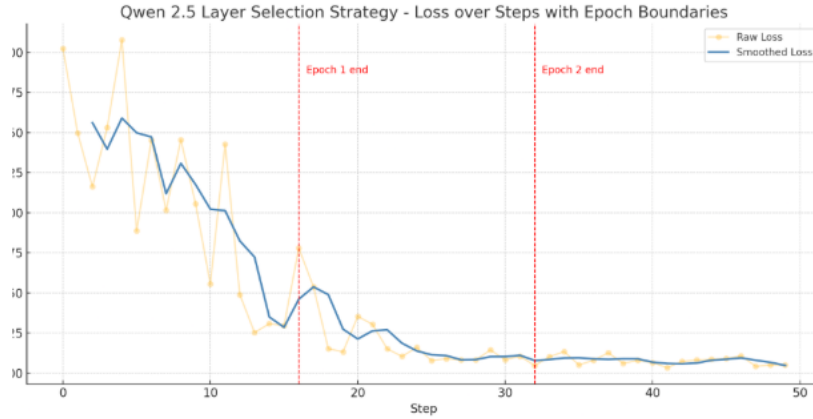


Figure 5.5: Evaluation loss of Qwen 2.5 using the proposed layer selection strategy. The model converges reliably, with steep early loss reduction and smooth stabilization across epochs, despite selective fine-tuning.

Altogether, the results highlight two critical insights:

- The proposed selective fine-tuning method is architecture-agnostic, functioning effectively across both encoder-decoder and decoder-only transformer models.
- It is also task-independent, showing stable and consistent convergence across diverse NLP tasks, from sentence classification to question answering.

These findings make a strong case for the broader utility of the approach, particularly in scenarios where full-model fine-tuning is computationally infeasible. The strategy offers a compelling balance between training efficiency and model performance, regardless of the underlying architecture or the nature of the downstream task.

### 5.3.2 LLM-RQ2: Skip Connections Optimization:

To further optimize the fine-tuning process of large language models using Low-Rank Adaptation (LoRA), we introduce and evaluate the use of **skip connections** for frozen transformer layers. This mechanism conditionally bypasses certain layers, specifically those that are not being adapted via LoRA, during the forward and backward passes. Skip connections were implemented by conditionally bypassing the forward computation of frozen layers during training. For layers where LoRA adapters are not applied, we replace the standard transformer computation with an identity function (i.e., no operation), effectively skipping them while preserving model compatibility and output shape.

**Results:** As shown in Figure 5.6, this approach led to a **10–12% reduction in training time**, with **no measurable loss in downstream task performance**. Accuracy metrics across all benchmark datasets remained within 0.5% of their baseline values, confirming the effectiveness and safety of this optimization.

**Implications:** The results validate that computation associated with frozen transformer blocks can be safely bypassed, especially in resource-constrained scenarios. This leads to faster training and reduced energy consumption, without impacting the integrity of model outputs.

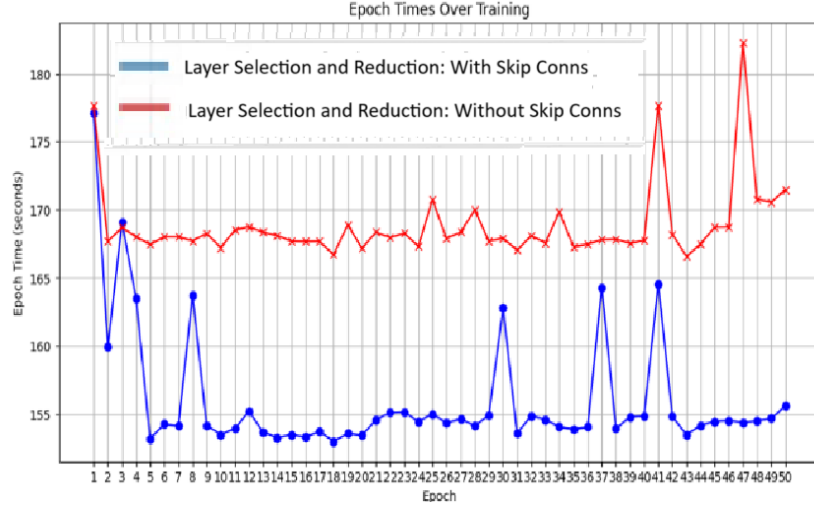


Figure 5.6: Training time comparison with skip connections (blue curve) and without skip connections (red curve) on the implemented algorithm for Encoder-Decoder architecture on QQP dataset. The skip-optimized model achieves up to 12% speedup without sacrificing performance.

## 5.4 Summary

Our empirical evaluation confirms that both Adaptive Split Federated Learning (ASFL) and Dynamic Layer Selection for LLM fine-tuning deliver gains without sacrificing task performance.

### Federated Split Learning

- **Accuracy and Robustness:** ASFL-P leads in IID settings, ASFL-G excels under OOD shifts, and ASFL-Hybrid achieves a balanced trade-off across mixed scenarios, always within 1–2 % of the best specialist.
- **Significant Communication Savings and Compute Efficiency:** ASFL shows significantly reduced communication and computational costs across datasets and settings, without any reductions in performance

### LLM Fine-Tuning

- **Task Performance:** On QQP, SST-2 and SQuAD, Dynamic Layer Selection (with or without skip-connections) matches full fine-tuning metrics within 0.5–1.0 point.

- **Time and Memory Reduction:** Training time drops by 12%, while convergence curves remain smooth and stable.
- **General Applicability:** The method works equally well for encoder–decoder (RoBERTa) and decoder-only (Qwen-2.5) models, and across classification and QA tasks.

Taken together, these results demonstrate that the methods can *replace* traditional full-model updates in both federated vision and LLM domains, achieving nearly identical accuracy with lower costs.

The next chapter concludes and outlines future work.



## CHAPTER 6

### Conclusion and Future work

#### 6.1 Introduction

This chapter summarizes the key findings of the thesis, highlights the primary contributions, and reflects on the practical implications of the proposed methods. It also outlines potential directions for future research aimed at improving scalability, adaptability, and efficiency in the training of models.

#### 6.2 Summary of Findings

This thesis demonstrates significant efficiency gains and practical benefits in three key areas:

- **Federated Split Learning for Decentralized Models:** The hybrid split-federated architecture with activation caching achieved on an average of 96× computation savings and 4.5× reduction in communication traffic while maintaining high accuracy under heterogeneous data.
- **Efficient Fine-tuning of LLMs:** Dynamic layer selection reduced fine-tuning time by 10% and lowered parameter updates significantly.

#### 6.3 Contributions of the Thesis

The principal contributions of this work are:

- **Novel Hybrid Split Learning Framework:** A two-phase protocol that balances global generalization with local personalization, leveraging combined loss regularization and activation caching.
- **Layer Selection Strategies for LLM Fine-tuning:** A gradient-norm-based algorithm integrated with LoRA and skip connections, optimizing resource usage without sacrificing task performance.

#### 6.4 Practical Implications

The methods developed in this thesis have broad applicability:

- **Applications in Telecom Data Processing:** Resource-efficient decentralized training can be deployed on edge routers and network appliances for real-time analytics and fault detection.
- **Deploying LLMs on Edge Devices:** Dynamic fine-tuning makes on-device adaptation of language models feasible for personalized assistants and field-support agents.
- **Real-World AI System Optimization:** The proposed frameworks inform best practices for balancing accuracy, privacy, and resource constraints in enterprise and industrial AI deployments.

## 6.5 Scope for Future Work

While this thesis lays a strong foundation, several avenues remain open:

- **Privacy Enhancements:** Incorporating differential privacy or secure multi-party computation within the hybrid split learning protocol.
- **Optimization Enhancements:** Incorporate techniques like caching, offloading to reduce GPU utilization for fine tuning in LLMs

## 6.6 Summary

In summary, this thesis advances efficient training techniques for both deep learning and LLMs and implements novel efficient training techniques that will serve as a foundation for future research and deployment.

## BIBLIOGRAPHY

- [1] H. Brendan McMahan et al. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. 2023. arXiv: 1602.05629 [cs.LG]. URL: <https://arxiv.org/abs/1602.05629>.
- [2] Tian Li et al. *Federated Optimization in Heterogeneous Networks*. 2020. arXiv: 1812.06127 [cs.LG]. URL: <https://arxiv.org/abs/1812.06127>.
- [3] Praneeth Vepakomma et al. *Split learning for health: Distributed deep learning without sharing raw patient data*. 2018. arXiv: 1812.00564 [cs.LG]. URL: <https://arxiv.org/abs/1812.00564>.
- [4] Peter Kairouz et al. *Advances and Open Problems in Federated Learning*. 2021. arXiv: 1912.04977 [cs.LG]. URL: <https://arxiv.org/abs/1912.04977>.
- [5] Yue Tan et al. “Is Heterogeneity Notorious? Taming Heterogeneity to Handle Test-Time Shift in Federated Learning”. *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: <https://openreview.net/forum?id=qJJmu4qsL0>.
- [6] Tao Yu, Eugene Bagdasarian, and Vitaly Shmatikov. “Salvaging Federated Learning by Local Adaptation”. *ArXiv abs/2002.04758* (2020). URL: <https://api.semanticscholar.org/CorpusID:211082601>.
- [7] Yue Zhao et al. “Federated Learning with Non-IID Data” (2018). DOI: 10.48550/ARXIV.1806.00582. URL: <https://arxiv.org/abs/1806.00582>.
- [8] Yihan Jiang et al. *Improving Federated Learning Personalization via Model Agnostic Meta Learning*. 2023. arXiv: 1909.12488 [cs.LG]. URL: <https://arxiv.org/abs/1909.12488>.
- [9] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. “Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach”. *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 3557–3568. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/24389bfe4fe2eba8bf9aa9203a44cdad-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/24389bfe4fe2eba8bf9aa9203a44cdad-Paper.pdf).
- [10] Karan Singhal et al. *Federated Reconstruction: Partially Local Federated Learning*. 2022. arXiv: 2102.03448 [cs.LG]. URL: <https://arxiv.org/abs/2102.03448>.
- [11] Fei Chen et al. “Federated Meta-Learning with Fast Convergence and Efficient Communication”. *arXiv: Learning* (2018). URL: <https://api.semanticscholar.org/CorpusID:209376818>.
- [12] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. “Adaptive Gradient-Based Meta-Learning Methods”. *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2019/file/f4aa0dd960521e045ae2f20621fb4ee9-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2019/file/f4aa0dd960521e045ae2f20621fb4ee9-Paper.pdf).

- [13] Xiaoxiao Li et al. “FedBN: Federated Learning on Non-IID Features via Local Batch Normalization”. *International Conference on Learning Representations*. 2021. URL: <https://openreview.net/forum?id=6YEQUn0QICG>.
- [14] Liam Collins et al. “Exploiting Shared Representations for Personalized Federated Learning”. *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 2089–2099. URL: <https://proceedings.mlr.press/v139/collins21a.html>.
- [15] Aviv Shamsian et al. “Personalized Federated Learning using Hypernetworks”. *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 9489–9502. URL: <https://proceedings.mlr.press/v139/shamsian21a.html>.
- [16] Michael Zhang et al. “Personalized Federated Learning with First Order Model Optimization”. *International Conference on Learning Representations*. 2021. URL: <https://openreview.net/forum?id=ehJqJQk9cw>.
- [17] Canh T. Dinh, Nguyen Tran, and Josh Nguyen. “Personalized Federated Learning with Moreau Envelopes”. *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 21394–21405. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/f4f1f13c8289ac1b1ee0ff176b5Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/f4f1f13c8289ac1b1ee0ff176b5Paper.pdf).
- [18] Filip Hanzely et al. “Lower Bounds and Optimal Algorithms for Personalized Federated Learning”. *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 2304–2315. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/187acf7982f3c169b3075132380986e4-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/187acf7982f3c169b3075132380986e4-Paper.pdf).
- [19] Tian Li et al. “Ditto: Fair and Robust Federated Learning Through Personalization”. *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 6357–6368. URL: <https://proceedings.mlr.press/v139/li21h.html>.
- [20] Minghui Chen et al. “FedSoup: Improving Generalization and Personalization in Federated Learning via Selective Model Interpolation”. *Medical Image Computing and Computer Assisted Intervention – MICCAI 2023*. Ed. by Hayit Greenspan et al. Cham: Springer Nature Switzerland, 2023, pp. 318–328. ISBN: 978-3-031-43895-0.
- [21] Dequan Wang et al. “Tent: Fully Test-Time Adaptation by Entropy Minimization”. *International Conference on Learning Representations*. 2021. URL: <https://openreview.net/forum?id=uXl3bZLkr3c>.
- [22] Yoonho Lee et al. “Surgical Fine-Tuning Improves Adaptation to Distribution Shifts”. *The Eleventh International Conference on Learning Representations*. 2023. URL: <https://openreview.net/forum?id=APuPRxjHvZ>.
- [23] Alexander Bartler et al. *MT3: Meta Test-Time Training for Self-Supervised Test-Time Adaption*. 2022. arXiv: 2103.16201 [cs.CV]. URL: <https://arxiv.org/abs/2103.16201>.

- [24] Yu Sun et al. “Test-Time Training with Self-Supervision for Generalization under Distribution Shifts”. *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 9229–9248. URL: <https://proceedings.mlr.press/v119/sun20b.html>.
- [25] Zhuang Qi et al. “Attentive Modeling and Distillation for Out-of-Distribution Generalization of Federated Learning”. *2024 IEEE International Conference on Multimedia and Expo (ICME)*. 2024, pp. 1–6. DOI: 10.1109/ICME57554.2024.10687423.
- [26] Xinting Liao et al. “FOOGD: Federated Collaboration for Both Out-of-distribution Generalization and Detection”. *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024. URL: <https://openreview.net/forum?id=D6MQrw9HFu>.
- [27] Yaming Guo et al. “Out-of-Distribution Generalization of Federated Learning via Implicit Invariant Relationships”. *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 23–29 Jul 2023, pp. 11905–11933. URL: <https://proceedings.mlr.press/v202/guo23b.html>.
- [28] Ting Chen et al. “A Simple Framework for Contrastive Learning of Visual Representations”. *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 1597–1607. URL: <https://proceedings.mlr.press/v119/chen20j.html>.
- [29] Ting Chen et al. “Big Self-Supervised Models are Strong Semi-Supervised Learners”. *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 22243–22255. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/fcbc95ccdd551da181207c0c1400c655-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/fcbc95ccdd551da181207c0c1400c655-Paper.pdf).
- [30] Ishan Misra and Laurens van der Maaten. *Self-Supervised Learning of Pretext-Invariant Representations*. 2019. arXiv: 1912.01991 [cs.CV]. URL: <https://arxiv.org/abs/1912.01991>.
- [31] Kaiming He et al. *Momentum Contrast for Unsupervised Visual Representation Learning*. 2020. arXiv: 1911.05722 [cs.CV]. URL: <https://arxiv.org/abs/1911.05722>.
- [32] Hyun-Jae Choi, Eric Jang, and Alexander A. Alemi. “WAIC, but Why? Generative Ensembles for Robust Anomaly Detection”. *arXiv: Machine Learning* (2018). URL: <https://api.semanticscholar.org/CorpusID:59553550>.
- [33] Eric Nalisnick et al. “Do Deep Generative Models Know What They Don’t Know?” *International Conference on Learning Representations*. 2019. URL: <https://openreview.net/forum?id=H1xwNhCcYm>.
- [34] Sam Dauncey et al. “Approximations to the Fisher Information Metric of Deep Generative Models for Out-Of-Distribution Detection”. *Transactions on Machine Learning Research* (2024). ISSN: 2835-8856. URL: <https://openreview.net/forum?id=EcuwtinFs9>.
- [35] Dan Hendrycks and Kevin Gimpel. “A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks”. *International Conference on Learning Representations*. 2017. URL: <https://openreview.net/forum?id=Hkg4TI9xl>.

- [36] Yen-Chang Hsu et al. *Generalized ODIN: Detecting Out-of-distribution Image without Learning from Out-of-distribution Data*. 2020. arXiv: 2002.11297 [cs.CV]. URL: <https://arxiv.org/abs/2002.11297>.
- [37] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles”. *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf).
- [38] Shiyu Liang, Yixuan Li, and R. Srikant. “Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks”. *International Conference on Learning Representations*. 2018. URL: <https://openreview.net/forum?id=H1VGkIxRZ>.
- [39] Weitang Liu et al. “Energy-based Out-of-distribution Detection”. *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 21464–21475. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/f5496252609c43eb8a3d147ab9b9c006-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/f5496252609c43eb8a3d147ab9b9c006-Paper.pdf).
- [40] Kimin Lee et al. “A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks”. *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2018/file/abdeb6f575ac5c6676b747bca8d09cc2-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2018/file/abdeb6f575ac5c6676b747bca8d09cc2-Paper.pdf).
- [41] Shiyu Liang, Yixuan Li, and R. Srikant. “Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks”. *International Conference on Learning Representations*. 2018. URL: <https://openreview.net/forum?id=H1VGkIxRZ>.
- [42] Rui Huang, Andrew Geng, and Yixuan Li. “On the Importance of Gradients for Detecting Distributional Shifts in the Wild”. *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021. URL: <https://openreview.net/forum?id=fmiwLdJCmLS>.
- [43] Conor Igoe et al. *How Useful are Gradients for OOD Detection Really?* 2023. URL: <https://openreview.net/forum?id=s0ceCGfcIKb>.
- [44] Sima Behpour et al. “GradOrth: A Simple yet Efficient Out-of-Distribution Detection with Orthogonal Projection of Gradients”. *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: <https://openreview.net/forum?id=L9nTuSbAws>.
- [45] Gal Kaplun et al. “Less is More: Selective Layer Finetuning with SubTuning”. *arXiv preprint arXiv:2305.17138* (2023).
- [46] Haoling Li et al. “Gradient-Mask Tuning Elevates the Upper Limits of LLM Performance”. *arXiv preprint arXiv:2402.05967* (2024).
- [47] Abhilasha Lodha et al. “On Surgical Fine-tuning for Language Encoders”. *arXiv preprint arXiv:2305.14314* (2023).
- [48] Arash Ardakani et al. “SlimFit: Memory-Efficient Fine-Tuning of Transformer-based Models Using Training Dynamics”. *arXiv preprint arXiv:2310.02220* (2023).
- [49] Yuchang Sun et al. “Exploring Selective Layer Fine-Tuning in Federated Learning”. *arXiv preprint arXiv:2403.01128* (2024).

- [50] Andrey Gromov et al. “The Unreasonable Ineffectiveness of the Deeper Layers”. *arXiv preprint arXiv:2402.09353* (2024).
- [51] Antonio Torralba and Alexei A Efros. “Unbiased look at dataset bias”. *CVPR 2011*. IEEE. 2011, pp. 1521–1528.
- [52] Da Li et al. “Deeper, broader and artier domain generalization”. *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5542–5550.
- [53] Hemanth Venkateswara et al. “Deep hashing network for unsupervised domain adaptation”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5018–5027.
- [54] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. “The HAM10000 dataset: A large collection of multi-source dermatoscopic images of common pigmented skin lesions”. *Scientific data* 5.1 (2018), pp. 1–9.
- [55] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms”. *arXiv preprint arXiv:1708.07747* (2017).
- [56] Alex Krizhevsky and Geoffrey Hinton. *Learning multiple layers of features from tiny images*. Technical Report, University of Toronto. 2009.
- [57] Richard Socher et al. “Recursive deep models for semantic compositionality over a sentiment treebank”. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013, pp. 1631–1642.
- [58] Shankar Iyer. *First Quora Dataset Release: Question Pairs*. <https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs>. Accessed: 2024-06-01. 2017.
- [59] Manas Wadhwa et al. “PFSL: Personalized and Fair Split Learning with Data and Label Privacy for thin clients”. *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. 2023, pp. 377–390. DOI: 10.1109/CCGrid57682.2023.00043.

## List of Publications

### 1. Conference

- (a) Under Review - Anumula Chaitanya Sai, **Moyank Giri**, Manisha Chawla and Gagan Raj Gupta, "Adaptive Split Federated Learning: Efficient Training and Robust against Distribution Shifts at Test Time", Conference on Information and Knowledge Management (CIKM) 2025, on May 24, 2025.
- (b) Under Review - Anshul Kumar, Gagan Raj Gupta, Manish Rai, Apu Chakraborty, Ashutosh Modi, Abdelaali Chaoub, Soumajit Pramanik, **Moyank Giri**, Sunny Kumar, Yashwanth Holla, M. V. Kiran Sooraj, "MM-Telco: Benchmarks and Multimodal Large Language Models for Telecom Applications", 31st Annual International Conference On Mobile Computing And Networking (MobiCom 2025) on March 19, 2025.