ATLAS

Adaptive Task-aware Layered Aggregation for Split Learning

Your Name

December 23, 2025

Why ATLAS? - Challenge 1: Model Size

- LLMs are too large for edge devices
- Modern models: 7B to 70B parameters
- Far exceed memory constraints of:
  - Mobile phones
  - IoT devices
  - Edge servers

Problem: Inference and training are infeasible on weak devices

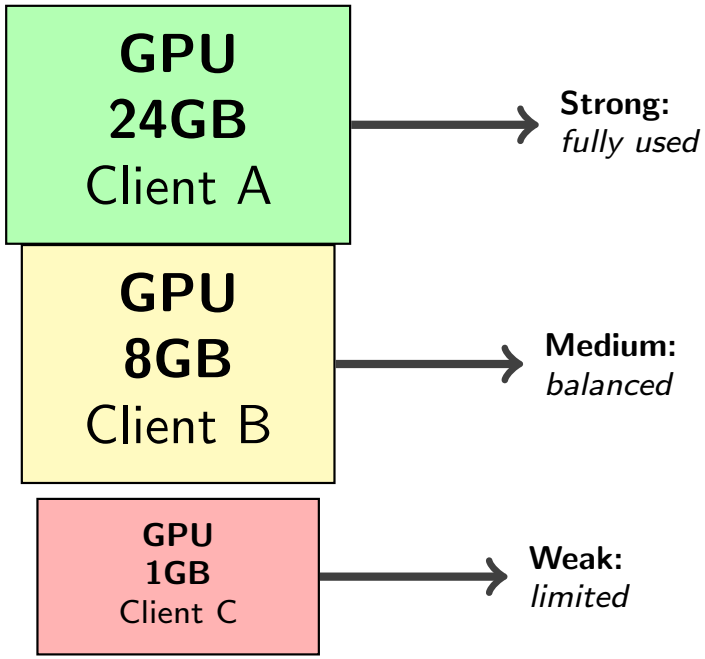
Why ATLAS? - Challenges & Device Heterogeneity

Challenge 2: Heterogeneity

- Different device budgets (1GB to 24GB GPU)
- Different task types (NLP, Q&A, summarization)
- Different network speeds (5G, 4G, WiFi)

Challenge 3: Efficiency

- One-size-fits-all approach
- Wastes resources on weak devices
- Underutilizes strong devices



Standing on Existing Work: SplitLoRA

SplitLoRA

- Split model between client and server
- Trainable LoRA adapters for efficiency
- Homogeneous setup (all clients identical)

Advantage: Efficient communication
Limitation: No heterogeneity support

Standing on Existing Work: HSplitLoRA

HSplitLoRA

- Heterogeneous LoRA ranks per client
- Adapted to device budgets
- Different configurations per device

Advantage: Efficient + heterogeneous
Limitation: No task awareness (ignores task similarity)

Standing on Existing Work: MIRA

MIRA

- Task clustering for small models
- Clients in same cluster share configurations
- Knowledge sharing between similar tasks

Advantage: Efficient + task-aware
Limitation: Not designed for split learning with LLMs

Comparison: ATLAS vs. Existing Methods

Method	**Heterogeneous**	**Task-Aware**	**Efficient**
SplitLoRA	No	No	Yes
HSplitLoRA	Yes	No	Yes
MIRA	Yes	Yes	No
ATLAS	**Yes**	**Yes**	**Yes**

ATLAS = Combining all three ideas for LLMs

ATLAS: System Overview

ATLAS operates in **four phases**:

1. **Phase 1: Discover** similar tasks through clustering
2. **Phase 2: Configure** device-specific LoRA ranks
3. **Phase 3: Train** using split learning
4. **Phase 4: Aggregate** updates per cluster

Result: Federated LLM training on heterogeneous devices with task awareness

Phase 1: Discover Similar Tasks - Goal

Goal: Group clients with similar tasks to enable knowledge sharing

Process:

1. Each client computes gradients on its own data (small batch)
2. Compress gradients into 64-dimensional "task fingerprint"
3. Server collects all fingerprints
4. k-Means clustering groups similar tasks

Privacy: Only low-dim vectors sent; no raw data or full gradients

Clients 1,2; [ellipse, draw, fill=green!20, minimum width=1.5cm, right=2.5cm of f3] (cl2) Cluster 2

Client 3;

[-¿, thick] (c1) – (f1); [-¿, thick] (c2) – (f2); [-¿, thick] (c3) – (f3); [-¿, thick] (f1) – (cl1); [-¿, thick] (f2) – (cl1); [-¿, thick] (f3) – (cl2);

Phase 2: Assign Heterogeneous Configurations

Goal: Choose LoRA rank and split point for each client

Process:

1. Determine weight importance per task cluster
2. Assign higher ranks to important weights
3. Respect each client's memory budget
4. Similar rank *patterns* within clusters, different *absolute values*

Example:

- Client A (24GB): ranks [8, 6, 4]
- Client B (8GB): ranks [6, 4, 2]
- Client C (1GB): ranks [4, 2, 1]

Phase 3: Train with Split Learning

Goal: Train LoRA adapters using client-server split architecture

Architecture:

- **Client:** Input processing + early layers + LoRA adapters
- **Server:** Task layers + loss head + LoRA adapters

Per round:

1. Client sends intermediate activations h to server
2. Server computes loss and gradients ∇h
3. Server sends gradients back to client
4. Both update LoRA adapters using local SGD

Key: No full model or gradients transmitted; minimal communication

Phase 4: Aggregate Updates Per Cluster

Goal: Merge LoRA adapters from same cluster for global model

Key Innovation: Noise-Free Aggregation
Even with different ranks, we can merge perfectly:

- Client 1: rank 4, Client 2: rank 6, Client 3: rank 4
- Concatenate low-rank factors $\Rightarrow$ rank 14
- Merge using aggregation formula
- Result: quality of averaging, no information loss

Process:

1. Stack all client adapters
2. Apply weighted average in merged space
3. Broadcast updated adapter back
4. Clients decompose for next round

ATLAS Advantages: Efficiency (30-40% Memory Saved)

- Weak devices get small LoRA ranks $\Rightarrow$ fit within budget
- Strong devices use larger ranks $\Rightarrow$ utilize resources
- Split learning minimizes communication overhead
- Heterogeneous config prevents over-provisioning

Result: Significant memory savings while maintaining performance

ATLAS Advantages: Accuracy (20-30% Gain)

- Task clustering enables knowledge transfer
- Clients specialize within clusters
- Shared cluster-level model provides regularization
- LoRA ranks ensure high-capacity learning

Result: Improved convergence and final accuracy

ATLAS Advantages: Privacy

- Only 64-D fingerprints sent (not raw data)
- No full gradients or model updates transmitted
- Optional: add differential privacy (DP) noise
- Evaluated using VFLAIR-LLM framework

Result: Privacy-aware federated learning

12-Week Implementation Plan

1. **Weeks 1–2:** PyTorch + Transformers setup + basic split learning
2. **Weeks 3–4:** Add LoRA from PEFT, test on GPT-2
3. **Weeks 5–6:** Phase 1 (task clustering with gradient PCA)
4. **Weeks 7–8:** Phase 2 (weight importance + rank selection)
5. **Weeks 9–10:** Phase 3 training loop + Phase 4 aggregation
6. **Weeks 11–12:** Evaluation + privacy check with VFLAIR

Tech Stack: PyTorch, HF Transformers, PEFT (LoRA), scikit-learn (k-means, PCA)

Evaluation Plan

Metrics:

- **Accuracy:** Task-specific metrics, convergence speed
- **Efficiency:** GPU memory, communication cost, training time
- **Privacy:** Attack success rate (VFLAIR), privacy-utility trade-off

Baselines:

- SplitLoRA (homogeneous)
- HSplitLoRA (heterogeneous, no clustering)
- Centralized fine-tuning (upper bound)

Expected Results:

- Accuracy: +15–25% vs HSplitLoRA
- Memory: 30–40% reduction
- Privacy: DCS ≥ 0.7

Open Questions for Supervisors

System Design:

- Fixed or adaptive split point?
- How many clusters K? (e.g., 3, 5, 10)
- Within-cluster regularizer?

Experiments:

- Which datasets? (FLAN, multi-task NLP)
- How many clients? (10, 50, 100, 1000)
- Which LLM? (GPT-2 first, then LLaMA-7B)

Resources:

- VFLAIR-LLM access or implement privacy attacks?
- GPU hour budget?
- Timeline flexibility?

Summary: ATLAS

ATLAS = Smart heterogeneity + Task awareness + Efficient aggregation

Core operations:

1. Discover similar tasks through clustering
2. Configure budget-aware LoRA ranks
3. Train via split learning
4. Aggregate without information loss

Novelty: Combines 3 existing ideas (MIRA + HSplitLoRA + SplitLoRA) for LLMs

Feasibility: 12-week timeline, all open-source, no novel breakthroughs

Technology Stack (Open-Source Guarantee)

All open-source:

- PyTorch (BSD)
- HuggingFace Transformers (Apache 2.0)
- PEFT LoRA (Apache 2.0)
- scikit-learn (BSD)
- VFLAIR-LLM (MIT)

NOT used:

- Titanic (proprietary)
- Any closed-source framework

What We Need from Supervisors

1. ✓**Research Direction Approval**
2. ✓**GPU Compute Budget** (e.g., 1000 GPU-hours)
3. ✓**Dataset/Task Specification** (which NLP tasks)
4. ✓**Timeline Confirmation** (12 weeks feasible?)

ATLAS is feasible, novel, and ready to build!