EgoSpark: Efficient Egocentric Question Answering with Distributed Fine-Tuning and Edge Deployment

Sunidhi Tandel
M.S. in Computer Engineering
New York University
sdt9243@nyu.edu

Rahil Singhi
M.S. in Computer Engineering
New York University
rs9174@nyu.edu

1 Introduction

Most vision-language models excel at reasoning and QA on static, third-person images, but struggle with egocentric data that requires spatial understanding and contextual planning.

EgoSpark tackles this challenge by fine-tuning large VLMs (InternVL3-34B / 8B) on the **Ego4D** dataset for egocentric question answering. The focus is on **system performance**: optimizing training with FlashAttention, mixed precision, and distributed parallelism, while profiling and removing bottlenecks.

We further apply **distillation** and **AWQ quantization** to compress and deploy the model on edge devices, turning EgoSpark into a complete HPC-driven pipeline from large-scale training to efficient inference.

2 Objective

Our objective is to build an efficient EgoQA system that:

- Achieves $\geq 2\times$ training throughput and $\geq 50\%$ memory reduction.
- Maintains $\leq 1\%$ accuracy drop (EM/F1) compared to baseline.
- Supports optimized inference with INT8/FP8 precision for edge devices.

3 Challenges

- Learning spatial reasoning from egocentric multi-view data.
- Training large 2B+ parameter models under GPU memory constraints.
- Preserving accuracy with parameter-efficient methods.
- Deploying large VLMs on edge hardware with limited compute.

4 Approach

4.1 Fine-Tuning Strategies

Method	**Trainable %**	**Description**
Full Fine-Tuning	100%	Baseline end-to-end training
LoRA	$\approx$ 2–3%	Low-rank adapters on attention/projector layers
Layer Freezing	$\approx$ 10%	Vision encoder frozen, LLM tuned

4.2 Distributed Training and Optimizations

- **FlashAttention-2**: I/O-efficient attention to reduce memory and latency.
- **Automatic Mixed Precision (BF16/FP16)**: Tensor Core acceleration.
- **torch.compile**: Graph-level kernel fusion.
- **Gradient Checkpointing**: Reduces activation memory usage.
- **DDP, FSDP, ZeRO-3**: Benchmark across distributed strategies (4–8 A100 GPUs).

4.3 System-Level and Deployment Optimizations

- Dynamic sequence padding to reduce padding overhead.
- Knowledge distillation from InternVL3-2B to a compact student model.
- **AWQ quantization** for multimodal instruction-tuned inference.
- Deployment on **TensorRT-LLM** for edge devices.

5 Implementation Details

- **Hardware:** 4–8 $\times$ NVIDIA A100 (80 GB)
- **Frameworks:** PyTorch 2.x, PEFT, DeepSpeed, bitsandbytes
- **Dataset:** Ego4D egocentric QA
- **Training:** Image size = 448 px, batch size = 8–16, lr = 2e-5, epochs = 3–5
- **Metrics:** EM/F1, tokens/s, VRAM usage, latency
- **Profiling Tools:** torch.profiler, Nsight Compute

6 Timeline (6 Weeks)

Week	Deliverable
1	Baseline fine-tuning and performance metrics
2	LoRA fine-tuning with AMP
3	FlashAttention integration and checkpointing
4	Benchmark: DDP vs FSDP vs ZeRO-3
5	Knowledge distillation and AWQ quantization
6	Edge deployment tests and final demo

7 Division of Work

Member	Focus
Sunidhi Tandel	Fine-tuning, CUDA-level optimizations, profiling
Rahil Singhi	Distributed setup, distillation, edge deployment
Both	Preprocessing, evaluation, visualization, demo development

8 Planned Demo

- Gradio QA interface: egocentric image + question $\rightarrow$ spatial answer.
- Performance dashboard: throughput, memory, accuracy comparison.
- Profiler visualization: kernel fusion and GPU utilization.
- Edge demo: INT8 AWQ inference on laptop GPU.

Work	Contribution	Our Extension
Dao et al. (2022)	FlashAttention	Integrated into VLM training
Dettmers et al. (2022)	8-bit Optimizers	Used for optimizer memory reduction
DeepSpeed ZeRO	Sharded Training	Benchmarked vs FSDP/DDP
Qwen2-VL / InternVL3	Base VLMs	Fine-tuned and distilled
NVIDIA TensorRT-LLM	Edge Inference	AWQ deployment