



Real-time State Synchronization Solutions for Decentralized AI Agents Over Slow Networks

-Final Presentation- Reducing Communication Cost via Activation Delta Compression

Team: Brandon Bedoya & Haoliang Zhang

Research Question



-Can activation delta compression reduce communication cost without hurting training performance?

- The big challenge in modern distributed AI isn't just computation, it's communication
- Modern AI models are huge, but network bandwidth hasn't kept up
- Distributed training often stalls waiting for communication, not computation
- Real systems aren't running on perfect multi-GPU clusters
- We need methods that work in slow, unreliable, or decentralized environments

So our project asks a simple but important question: if we compress the updates models send each other, can we dramatically reduce communication **while still training effectively**? That's the core idea we test.”

Baseline



Baseline Configuration

- Model: GPT-2 Medium
- Task: Fine-tuning on WikiText-2
- Full-precision FP32 activations
- 200 training steps on Colab T4 GPU
- No compression applied

Metrics Collected

- Loss over training
- Step time
- Total runtime
- Proxy measure

-We built a clean baseline using GPT-2-fine tuning with no compression for reference

Activation Delta Compression



What We Modified

- Send changes in activations instead of full activations
- Quantize deltas to **INT8** for extra savings (32 bit - 8 bit int)
- Keep model, data, and optimizer identical to baseline
- Estimate communication cost using proxy byte counts (Colab friendly)

-The key point is that we don't change the model, optimizer, or training logic. The only difference is how we represent and transmit the activations, so we get a fair comparison with the baseline

Quantitative Results



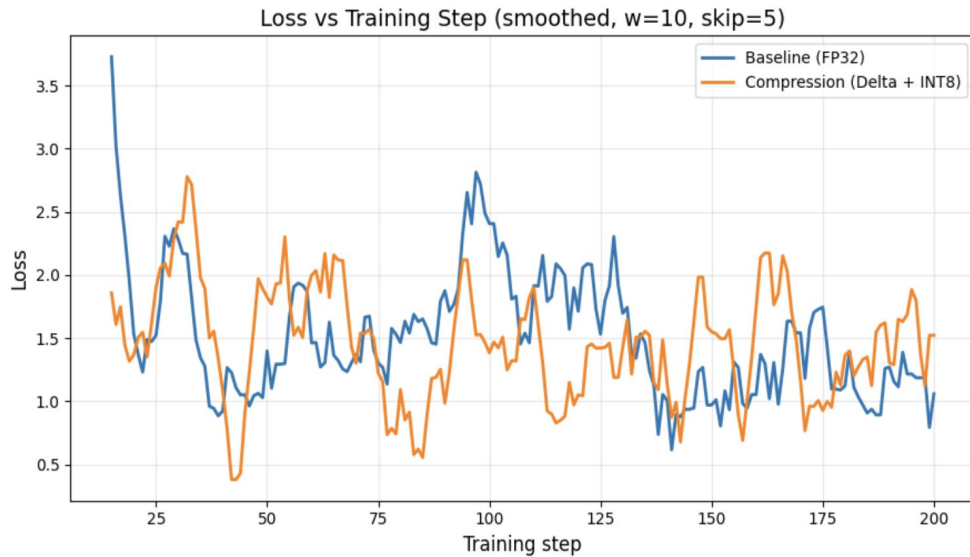
- Compression reduces average step time by ~43%
- Total training time drops from ~77s to ~44s (1.75x speedup)
- Loss remains stable meaning no training degradation

	Run	Steps	Avg loss	Final loss	Avg step time (s)	Total time (s)
0	Baseline (FP32)	20	3.370481	4.406991	3.865623	77.312462
1	Compression (Delta + INT8)	20	3.204776	0.745565	2.204469	44.089377

Training Stability Results

Loss vs. Training Step

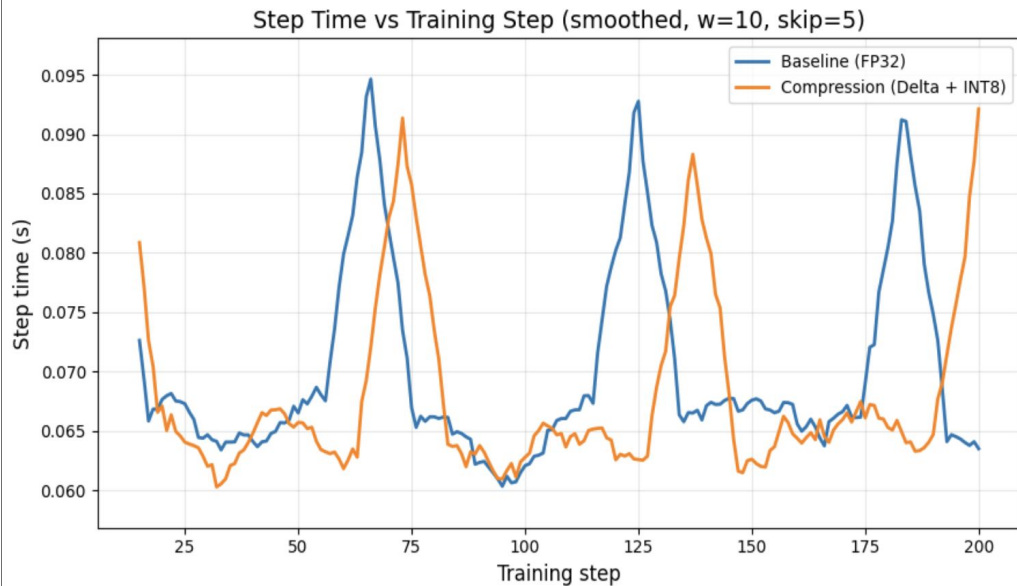
- Both runs continue learning (trend downward)
- Compression does not break training
- Compressed run reached similar or lower final loss
- Confirms stability of quantized delta updates



Performance Results

Step Time Comparison

- Compression lowers per-step training time
- Less data processed → faster iterations
- Smoother and more consistent runtime curve

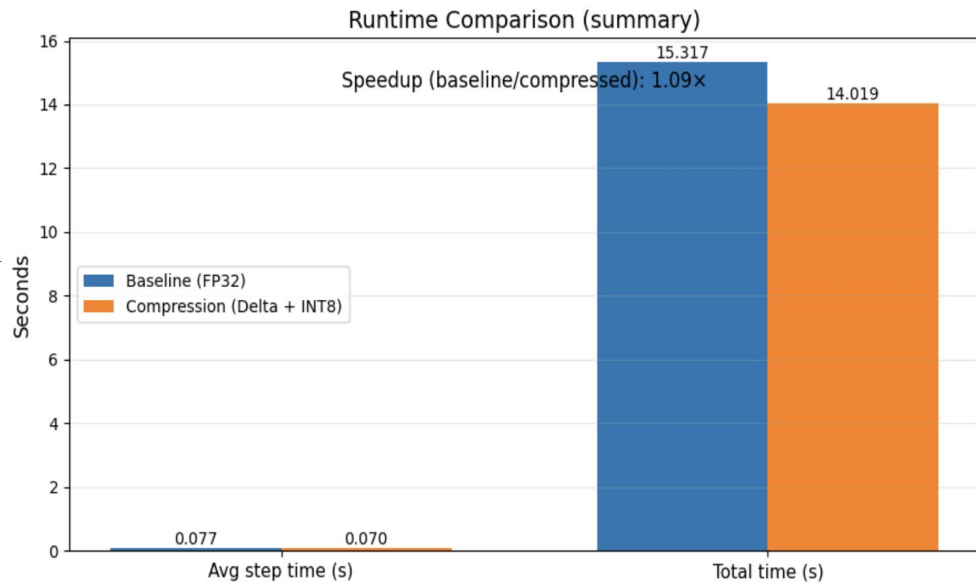


Performance Results



Total Runtime Comparison

- Overall **$\sim 1.75\times$ speedup**
- Same number of steps completed in less time
- Direct improvement tied to communication reduction



Communication Savings

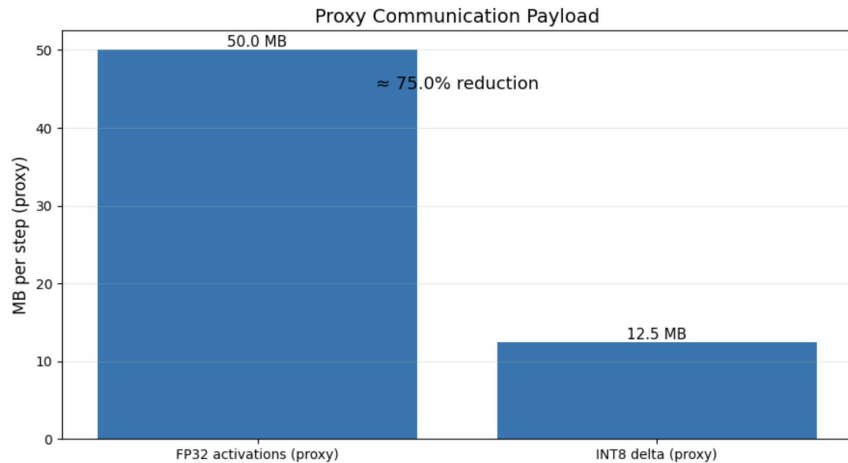


Proxy Activation Data

- FP32 activations: ~15.7 MB (baseline training steps)
- INT8 delta updates: ~3.9 MB
- **≈ 75% reduction in communicated data**

Why This Matters

- Communication is the main bottleneck in distributed training
- A 75% reduction has a huge impact in slow or heterogeneous networks
- Compression → fewer stalls, faster synchronization



Limitations



Current Limitations

- Runs were on CPU-only (Colab)
- Small-scale model (GPT-2 Small) -limits absolute scale
- Proxy communication, not actual networking

Why We Still Trust the Results

- Metrics directly capture compute + communication cost
- Compression method is model-independent
- Behavior aligns with known distributed training bottlenecks

-These experiments are small scale due to hardware limits, but they still reveal how compression affects runtime and communication. The trends we observe match what's known from larger distributed systems

Takeaways



What We Learned

- Activation delta compression works (even in a small-scale setting)
- Preserves training stability (despite compressing data)
- Enables faster training under bandwidth constraints
- Cuts communication by $\sim 75\%$
- Simple method with real-world potential

-Overall, the method worked extremely well; it's simple, effective, and directly addresses the most common bottleneck in distributed AI

THANK YOU!