Paper Review - Apparate: Rethinking Early Exits to Tame Latency-Throughput Tensions in ML Serving

Sai Venkat Malreddy

7 november 2024

1 Idea and Contribution

Apparate addresses a critical challenge in ML inference platforms: the tension between achieving high throughput (processing many requests efficiently) and maintaining low latency (quick response times) for interactive applications. Traditional solutions like adjusting batch sizes force a harsh trade-off between these goals. Apparate introduces a novel approach using "early exits"(EEs) - points in the ML model where certain inputs can get early results - but with a key innovation: unlike traditional EE approaches that stop computation early, Apparate continues processing inputs even after they exit. This seemingly counterintuitive choice enables continuous accuracy monitoring and adaptation while maintaining compatibility with batch processing for high throughput. The system automatically injects these exits into models and manages them through a two-level adaptation strategy: frequent threshold tuning for accuracy preservation and periodic ramp adjustments for latency optimization. The results are impressive: 40.5-91.5% lower median latencies for computer vision tasks and 10.0-24.2% for NLP tasks, while maintaining accuracy within 1% of the original model and preserving throughput. This makes Apparate a practical solution for real-world deployment, as it requires no developer expertise and works seamlessly with existing serving platforms.

2 Three Positive Comments

- The paper provides comprehensive evaluation of Apparate across multiple platforms (TensorFlow-Serving, Clockwork, HuggingFace Pipelines) and diverse workloads (CV and NLP tasks). The experimental results convincingly demonstrate significant latency improvements while maintaining accuracy within 1%, making the evaluation thorough and credible.
- Apparate introduces an innovative approach to early exits by continuing computation after exit points, which enables continuous accuracy monitoring while preserving batch processing benefits. This clever design choice addresses the fundamental tension between latency and throughput in ML serving systems without requiring modifications to existing infrastructure.
- The system is designed for practical deployment with automatic model preparation, requiring no developer expertise and remaining compatible with other optimization techniques like model compression and batching. The two-level adaptation strategy (threshold tuning and ramp adjustment) effectively manages the complexity of runtime optimization.

3 Three Negative Comments

- While Apparate achieves impressive latency improvements, it sacrifices potential compute savings by continuing computation after exits. The paper could have explored methods to selectively apply this approach based on workload characteristics or system load, potentially offering better resource efficiency.
- The evaluation, though extensive for classification tasks, lacks deeper analysis of very large language models and multi-tenant scenarios. Additionally, there is limited discussion of energy consumption impact and failure recovery mechanisms, which are crucial considerations for production deployment.
- The system's two-level adaptation strategy, while effective, introduces significant complexity with multiple parameters to tune (window sizes, check intervals, thresholds). The paper could benefit from a more detailed analysis of parameter sensitivity and guidelines for automatic parameter selection in different deployment scenarios.

4 Questions

- Parameter Selection and Sensitivity

  How are optimal window sizes (16 samples) and check intervals (128 samples) determined? What is the sensitivity of performance to these parameters? Could these parameters be automatically tuned based on workload characteristics?

- Resource Management in Multi-GPU Settings

  How does Apparate handle distributed inference across multiple GPUs? What is the communication overhead between GPUs? How are ramp placements optimized for distributed scenarios?