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

Yinwei Dai[1,2] Rui Pan[1,2] Anand Iyer[2] Kai Li[1] Ravi Netravali[1]

[1]Princeton University
[2]Georgia Institute of Technology

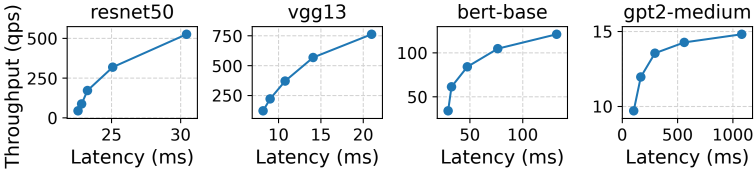
Presenter: Shiwei Zhang

Introduction

- Machine Learning (ML) inference has become a staple for request handling in interactive applications such as traffic analytics, chatbots, and web services
- Existing platforms impose harsh tradeoffs between **throughput** (cost) and **latency**.
- This paper explores **early exits (EE)** to resolve this tension.

Background

Throughput-Latency Tradeoff

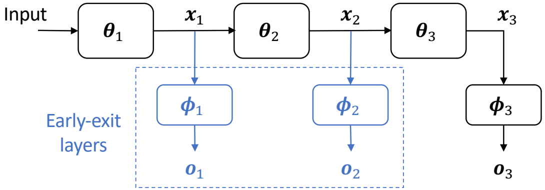


Latency for an input is minimized by scheduling inference as soon as the request arrives with batch size of 1.

Throughput is maximized by creating large batches using a queuing system which directly inflates request latencies.

Early-Exit Models

EE inserts exit points (also called **ramps**) into the model to conditionally produce results without running some of the layers.



For example, an object detection model do not need to run all layers to decide that there is nothing on an empty video frame.

Exiting decisions are made by comparing the entropy in the predicted result to a **threshold**.

Challenges

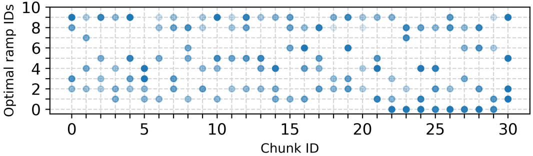
C1: Latency and Resource Overheads

- The ramps take up GPU memory (6.56% for BERT-Base).
- The additional exiting decisions increase latency of “hard” requests (up to 22%).

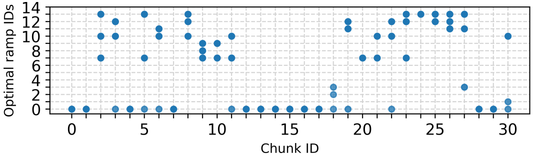
C2: Frequent and Costly Adaptation

The optimal configurations (active ramps and their thresholds) changes frequently.

Strategy\Workload	CV	NLP
Initial Only	84.5% (74.3%)	86.8% (73.6%)
Uniformly Sampled	90.3% (64.2%)	87.7% (69.4%)
Continual Tuning	98.6% (43.5%)	98.3% (26.6%)



(a) BERT-base; Amazon reviews



(b) ResNet50; random corpus video

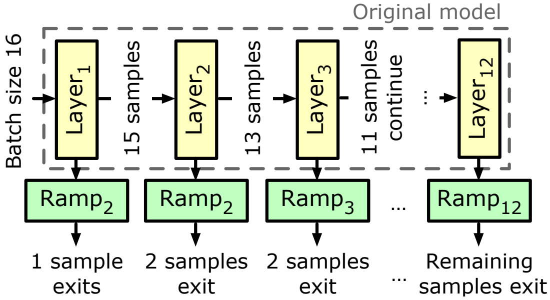
C3: Lack of Accuracy Feedback

In production scenarios, serving optimizations that deliver accuracy reductions of more than 1-2% are generally considered unacceptable. However, current EE proposals suffer from accuracy drops up to 23.9%.

Once deployed, EE models do not provide indication of accuracy drops. When an exit is taken, the corresponding input does not pass throuput the remaining model layers, and the original model's prediction is never revealed.

C4: Incompatibility with batching

When inputs exit at ramps, the batch size for *already scheduled tasks* shrinks, leading to resource underutilization for the rest of the execution.



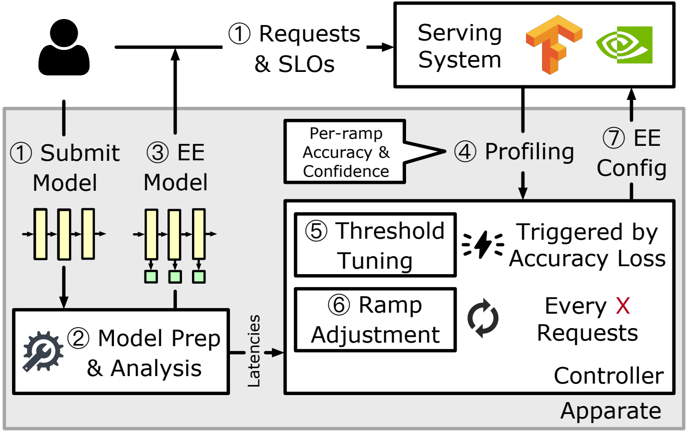
Design

Key Design

Apparate focuses solely on latency savings by **allowing results to exit**, with **inputs still running to completion**.

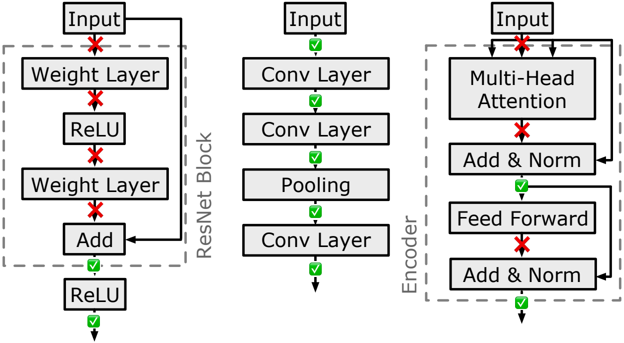
- Foregoing true exiting eliminates batch size changes during inference (C4).
- Running to completion grants Apparate with direct and continual feedback on accuracy (C3).
- This feedback provides the requisite information for continual adaptation of EE configurations (C1, C2).

System Architecture



Ramp Injection

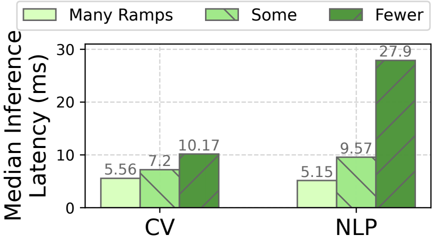
Apparate considers layers that are *cut vertices* as EE candidates.



(a) ResNet (b) VGG (c) BERT

Ramp Architectures

Apparate chooses many lightweight ramps instead of fewer expensive ramps.



Threshold Tuning

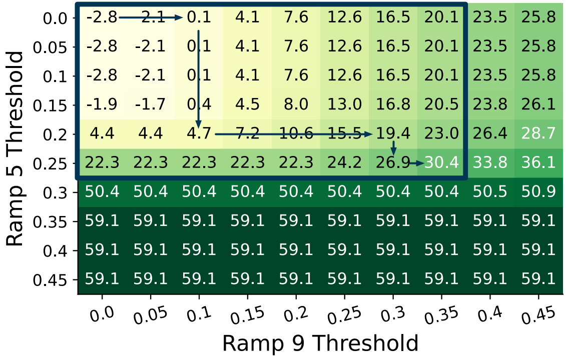
Triggering: Threshold tuning is triggered any time a window's accuracy falls below the user-specified accuracy constraint.

Evaluation: Apparate identifies the earliest ramp whose top prediction now has an error rate below its threshold and compare its result with the original model output.

Greedy Search: Based on the fact that *higher thresholds result in monotonic decreases in accuracy and latency*, Apparate designs a hill climbing algorithm to decide the thresholds for active ramps.

Greedy Search

Starting with thresholds of 0 for all active ramps, Apparate choose a direction with largest **additional latency saving per unit of additional accuracy loss**. Apparate follows a **multiplicative increase, multiplicative decrease** policy on step size to balance search speed and granularity.



Ramp Adjustment

Apparate periodically adjusts the active ramps, which ultimately provides bounds on potential latency savings.

Unlike threshold tuning which runs reactively, ramp adjustment runs periodically. This is because threshold directly affects accuracy and must react promptly, while ramp adjustment only affects latency and is pure optimization. In addition, ramp adjustment requires deployments to evaluate the impact of new ramps.

Removing Active Ramps

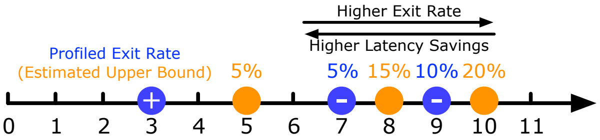
Apparate defines the utility of ramp R $=$ `savings` $-$ `overheads`, where `savings` is the sum of raw latency that exiting inputs avoided by using R, and `overheads` is the sum of raw latency that R added to inputs that did not exit.

If any negative utility values exist, Apparate applies a fast threshold tuning round to see if ramp utilities become positive without harming overall latency savings. If not, Apparate deactivates all negative-utility ramps.

Adding New Ramps

Intuition: later ramps exhibit higher exit rates than earlier ones.

The upperbound exit rate is calculated as the sum of profiled exit rates for the following deactivated ramp and any earlier deactivations.



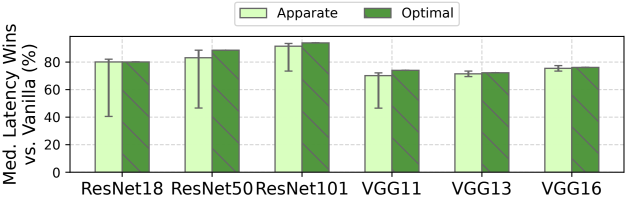
The ramp with the highest utility score is selected for trail.

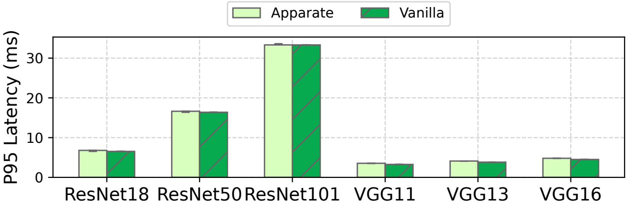
Evaluation

Setup

- **Models**: 10 models across 4 families, covering CV and NLP.
- **Workloads**: CV workloads comprise real-time object classification on 8 one-hour videos. NLP workloads use Amazon product reviews and IMDB movie reviews.
- **Parameters**: SLO is 2x of each model's inference time with batch size 1. Accuracy constraint is 1%. Ramp overhead budget is 2% impact on worst-case latency.
- **Hardware**: Single machine with one NVIDIA A6000 and 32-core AMD CPU.
- **Platforms**: TensorFlow-Serving and Clockwork (using PyTorch).

Comparison with Baselines



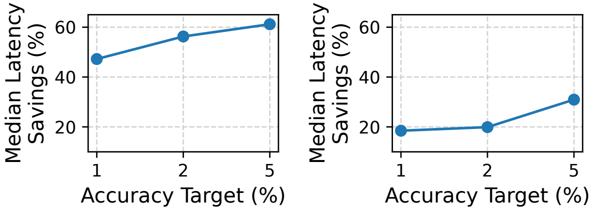


Comparison with Existing EE Strategies

Existing EE approaches yield unacceptable accuracy drops while having longer tail latencies as compared to Apparate.

	Avg Acc	Median Wins	P95 Wins
Apparate (ResNet50)	99.0-99.2%	40.9-88.6%	-1.6-0.0%
BranchyNet	85.8-99.8%	-11.0-88.3%	-11.0%
BranchyNet+	76.1-99.9%	-11.0-88.3%	-11.0%
BranchyNet-opt	99.0-99.7%	-11.0-74.5%	-11.0%
Apparate (BERT-base)	99.1-99.3%	9.1-24.1%	0.7-1.8%
DeeBERT	91.7-97.1%	13.2-36.1%	-1.3-6.4%
DeeBERT+	82.2-90.3%	31.7-36.1%	5.9-6.4%
DeeBERT-opt	99.0%	9.8-36.1%	-1.4-6.4%

Microbenchmarks - Parameter Sensitivity



(a) ResNet50 **(b) GPT2**

Ramp Budget	ResNet50	GPT2
2%	48.9%	18.5%
5%	49.6%	22.2%
10%	50.4%	24.9%

Microbenchmarks - Ramp Architectures

When using DeeBERT's more expensive ramps, Apparate performs **4% worse** since the costlier ramps constrain Apparate's runtime adaptation, i.e., **fewer active ramps** at a time.

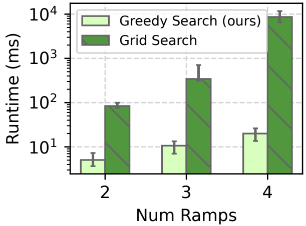
Microbenchmarks - Impact of Serving Platform

The (median, p95) latency over vanilla models are similar on the two platforms.

System\Workload	**ResNet50**	**GPT2**
Clockwork	(20.2, 37.8)	(689.2, 779.4)
TF-Serve	(24.5, 37.8)	(709.3, 793.1)

Microbenchmarks - Profiling Apparate

Ramp adjustment rounds take 0.5ms. Additional CPU-GPU communication take 0.5ms, where 0.4ms comes from PCIe latencies.



Microbenchmarks - Ablation Study

Disabling ramp adjustment results in 20.8-33.4% lower median latency wins.

Conclusion

Conclusion

Strength:

- Coherent design around a novel idea (running EE models to completion).
- Automatic and non-intrusive system (no modification on model definition and serving platform).

Limitation:

- The paper motivates with the throughput-latency tradeoff, but the proposed solution is accuracy-latency tradeoff.
- Limited to time-related tasks.
- Early returning results for some samples may not always be meaningful. e.g., during LLM decoding.

Takeaways

- EE may be promising in cutting the cost of large models serving.
- Workload-specific optimization can be very powerful. e.g., vLLM's design for beam search.
- An idea: combine with pipeline parallelism?

Thank you!

Training and Deployment

Apparate prohibits early exits during initial training, ensuring that ramps are trained independently. This is because Apparate will adaptively change the active ramps at runtime.

For initial deployment, Apparate evently space the ramps based on the budget and GPU memory. To avoid accuracy dips, all ramps begin with thresholds of 0, i.e., no exiting.