RTInfer: Real-Time Inference of Multiple DNNs on Edge GPUs

Renjie Li · Tong Sun · Yi Gao · Wei Dong
College of Computer Science, Zhejiang University

ICML International Conference On Machine Learning 2026 · POSTER

Code | Homepage | Emnets Lab

1 Problem

Multi-RT DNN bursts break serial edge-GPU scheduling.

Queue-first systems react late: they trade deadline pressure for accuracy loss and memory pressure.

Workload changed

RT streams arrive together.

Robots, UAVs, and traffic workloads release multiple DNNs with different deadlines and priorities.

Serial catch-up fails

Preemption moves the bottleneck.

A high-priority queue can force shallower exits, but it does not prove that memory layout and loading are feasible.

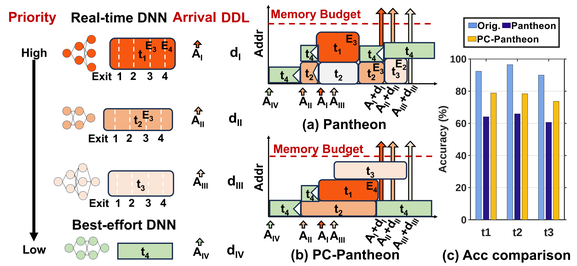
RTInfer goal

Exploit concurrency without emergency shallow exits.

One admission check chooses the variant, places buffers, and schedules Delta chunks before first use.

2 Motivation: Serial SOTA Sacrifices Accuracy

Serial RT queueing is the failure mode behind the problem: later urgent jobs preempt current work, preempted feature state stays in memory, and early exits become the main escape path when deadlines tighten.



Serial execution and PC-Pantheon reduce deadline misses by forcing earlier exits, but pay for feasibility with accuracy.

Why serial execution fails

- Later urgent jobs preempt current work and keep intermediate state alive.
- Early exits reduce queueing, but directly trade accuracy for timeliness.
- Naive pruning relieves memory pressure, yet removes capacity before the runtime knows which jobs will arrive together.

Why this matters The nominal application memory demand already exceeds the effective Jetson budget before BE work, staging buffers, allocator slack, and preempted feature state are added. RTInfer therefore cannot rely on queue order alone; it must prove that all active buffers and transfers fit in the same time–address plan.

Key insight Multi-RT-DNN concurrency can reduce queueing and early-exit dependence, but only when **which variant to run**, **where its buffers live**, and **when missing chunks arrive** are solved as one scheduling decision.

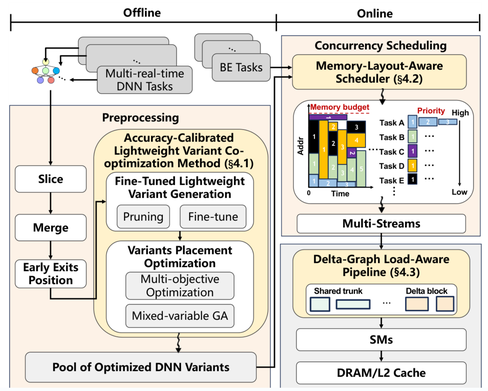
6500	7070	7950
MiB	**MiB**	**MiB**
traffic	UAV	robot

Challenges

- **Memory vs. accuracy.** Concurrent RT inference can exhaust GPU memory, while naive pruning relieves pressure by sacrificing accuracy.
- **Urgency vs. layout.** Scheduling only by temporal urgency can fragment memory and erode concurrent performance.
- **Loading latency.** Host-to-GPU on-demand loading during variant switches can dominate RT latency and miss deadlines.

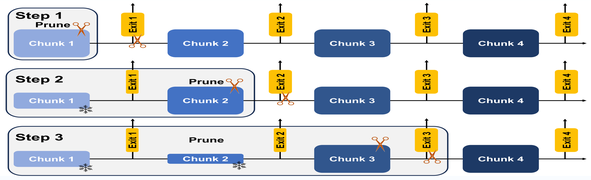
3 Framework: Offline Variants + Online Layout

RTInfer prepares accuracy-safe variants offline, then online packs active tasks in a **time** × **memory** plane and prefetches sparse deltas before first use.



Offline profiling feeds memory-aware scheduling and Delta-Graph loading.

Variant atlas: acc./lat./mem./delta bytes. **Layout:** place time–address rectangles. **Delta:** load chunks before first use.



Technique 1 generates candidates by pruning chunks and choosing exits, then profiles accuracy, latency, and memory.

RTInfer jointly chooses pruning ratio p and early-exit depth e, then fine-tunes promising variants offline rather than waiting for emergency shallow exits at runtime:

$$\min_{E,P} L(E,P) + \lambda_1 M(E,P) + \lambda_2\big(1 - A(E,P)\big)$$

subject to $\Delta a \le \alpha$ and $M \le \beta$.

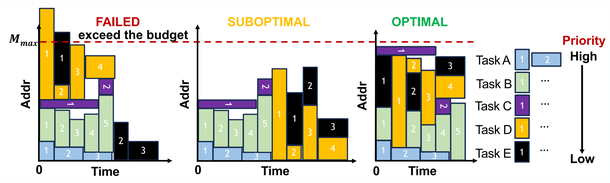
The offline pass removes dominated prune/exit choices, records first-use layers and Delta bytes, and leaves runtime with a compact Pareto pool rather than a combinatorial search.

Pipeline summary Offline, RTInfer keeps Pareto-safe variants with profiled accuracy, latency, memory, and delta bytes; online scheduling admits one only when execution, placement, and first-use transfer constraints pass together.

4 Online Scheduling: Layout + Delta Loading

Once a burst arrives, RTInfer preserves urgency ordering but checks whether the chosen variants actually fit in memory over time:

$$[t_i^{start}, t_i^{end}) \times [a_i, a_i + size_i)$$



Technique 2 searches feasible time–address placements: enough total memory is not enough if rectangles collide.

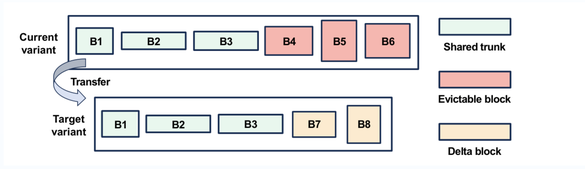
Placement contract. Accept a candidate only if execution meets the deadline, active buffers do not collide, and missing chunks arrive before first use.

Feasible placement search

- Model every live buffer as a rectangle in the time–address plane.
- Buffers overlapping in time cannot overlap in GPU address space.
- Priority orders urgency; layout decides whether jobs can coexist.

Variant switching is scheduled too. RTInfer stores a shared trunk plus sparse deltas, then prefetches missing chunks before their first use:

$$t_c^{load\_end} \le s_l^{first-use}$$



Shared trunk blocks stay resident; variant switches move only sparse delta chunks.

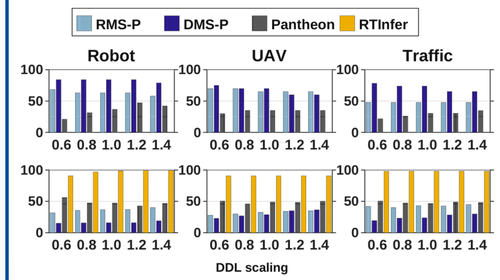
- **Offline:** compile each variant into a resident shared trunk plus sparse 1–4 MiB deltas.
- **Online:** prefetch only missing chunks, and require each transfer to finish before its first use.
- **Fallback:** if loading or address placement fails, step down to the next lighter variant.

Admission loop Each re-plan preserves EDF-style urgency, tries the strongest feasible variant, places buffers, and enqueues only Delta chunks whose first use is still in the future; if any check fails, RTInfer steps down to a lighter variant. Arrivals, completions, and memory releases trigger the next short event-driven re-plan.

5 Results

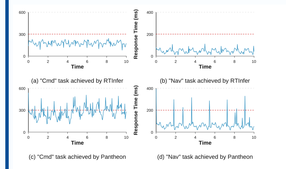
Evaluation setup Jetson Xavier NX, 6144 MiB budget, 15 RT streams; κ sweeps pressure; baselines: RMS-P, DMS-P, and Pantheon.

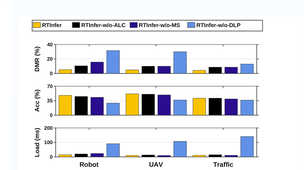
Across vision, classification, and generation workloads, RTInfer prioritizes deadline feasibility first and spends remaining slack on accuracy-preserving variants rather than paying for timeliness through repeated shallow exits.

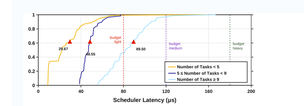


Across deadline scaling factors $\kappa \in \{0.6, 0.8, 1.0, 1.2, 1.4\}$, RTInfer keeps DMR at 0% while preserving accuracy.

Under tight deadlines $\kappa = 0.6$–0.8, RMS-P and DMS-P average **61.2%** and **77.6%** DMR; Pantheon has **27.6%**. RTInfer maintains **0% DMR** with **94.2%** average accuracy. At $\kappa = 1.0$, it still keeps **0% DMR** and reaches **95.8%** average accuracy.







Response traces, GPU utilization, ablations, and scheduler-latency CDF summarize RTInfer's runtime behavior and overhead.

Result reading Feasible concurrency keeps response traces below deadlines, raises GPU use under multiple tasks, and keeps scheduling overhead in the microsecond range; ablations show the gains come from feasibility-aware layout and Delta loading rather than from relaxing deadlines.

6 Takeaways

CONCURRENCY-FIRST. Overlap bursts instead of serial catch-up.

FEASIBILITY CHECKED. Deadline, address layout, and Delta loading pass together.

ACCURACY KEPT. Offline variants avoid emergency shallow exits.

EDGE-READY. 0% DMR on Jetson NX; microsecond scheduling.

Open source: `https://github.com/ELE-Jay/RTInfer`

Contact: gaoyi@zju.edu.cn

EMNETS · ZHEJIANG