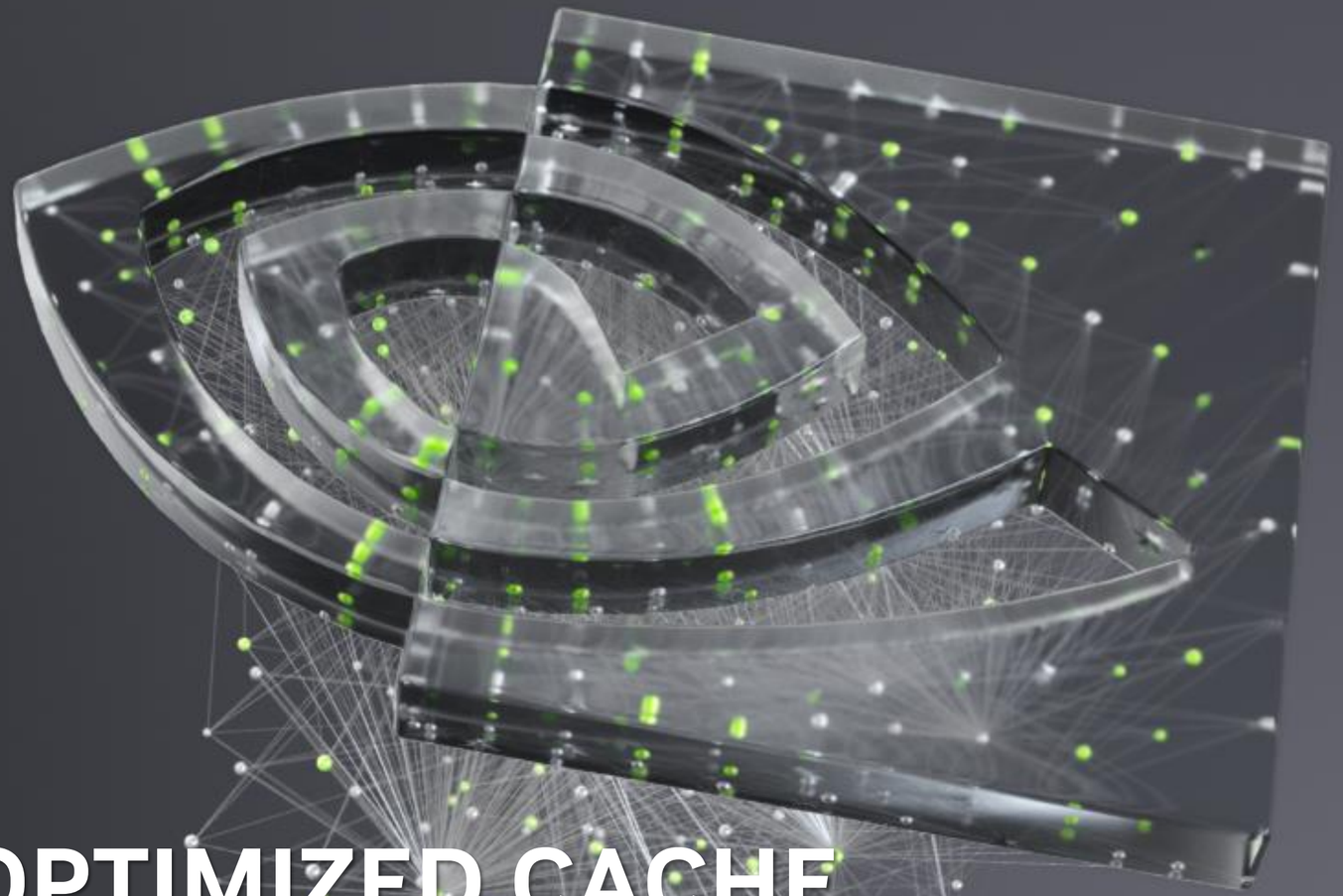


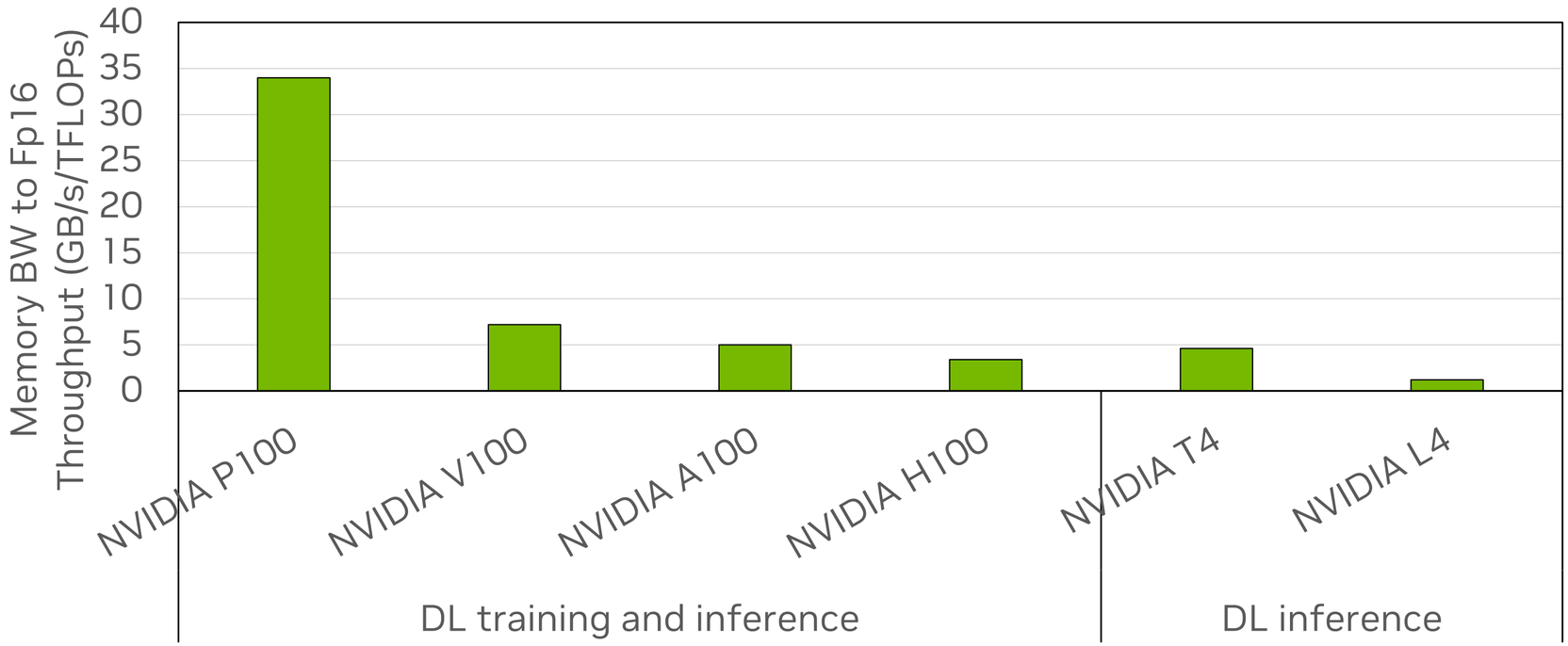
AUTOSCRATCH: ML-OPTIMIZED CACHE MANAGEMENT FOR INFERENCE-ORIENTED GPUS

Yaosheng Fu, Evgeny Bolotin, Aamer Jaleel, Gal Dalal, Shie Mannor, Jacob Subag, Noam Korem, Michael Behar and David Nellans.

NVIDIA

Motivation

GPU DRAM bandwidth and power efficiency limit DL workload performance

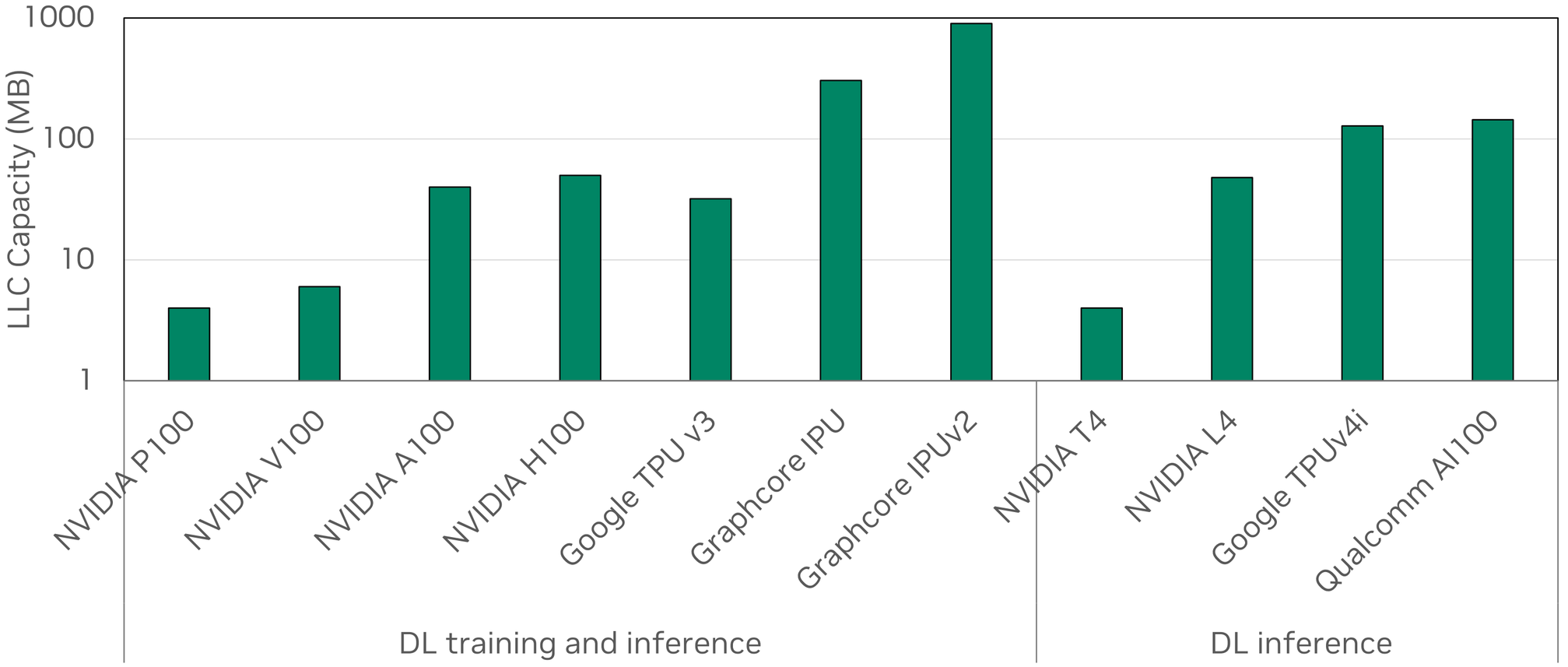


Memory BW to FP16 throughput ratio (GB/s/TFLOPs) on NVIDIA GPU generations.

- Memory bandwidth scales slower than compute capability (FP16) for DL workloads on GPUs.
- Memory power consumption becomes more significant especially in inference-oriented GPUs (could be >40%).

Observation

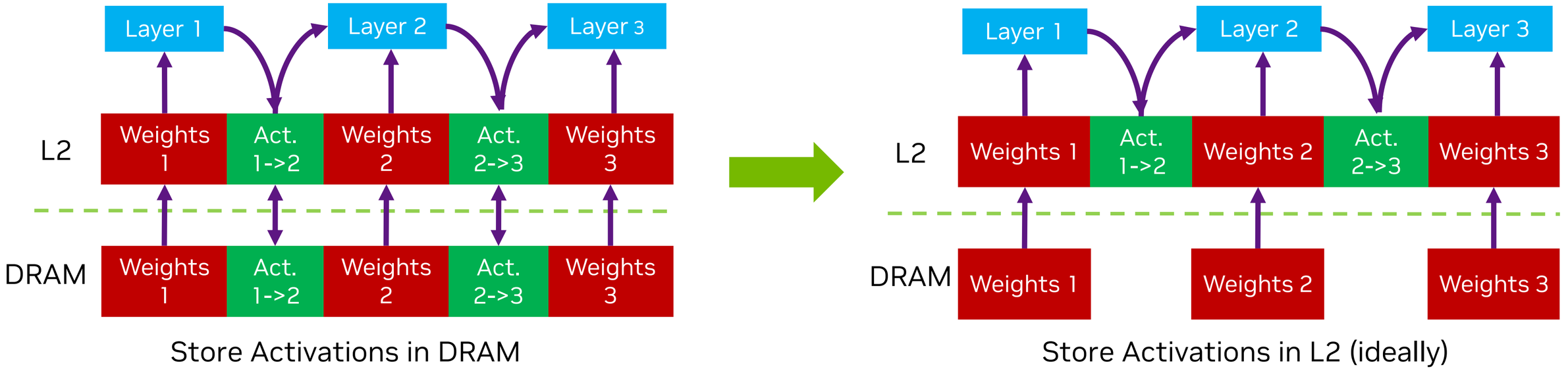
The LLC (last-level cache) capacity on GPUs and DL accelerators keeps increasing



Q: Can we make more efficient use of the LLC to save more DRAM BW and power consumption?

Memory Access Pattern in DL Inference

Repetitive producer-consumer pattern for activations



Ideally, storing activations in L2 can completely eliminate DRAM traffic on activations.

L2 Cache Residency Control

New HW feature for explicit control of data residency in L2

L2 Access Property	Priority of Staying in the L2
Persisting	High
Normal	Medium
Streaming	Low

L2-resident Data

L2 residency control defines 3 types of L2 access properties.

L2 cache residency control is a new HW feature allowing programmers to explicitly control data residency (first introduced in NVIDIA Ampere GPUs).

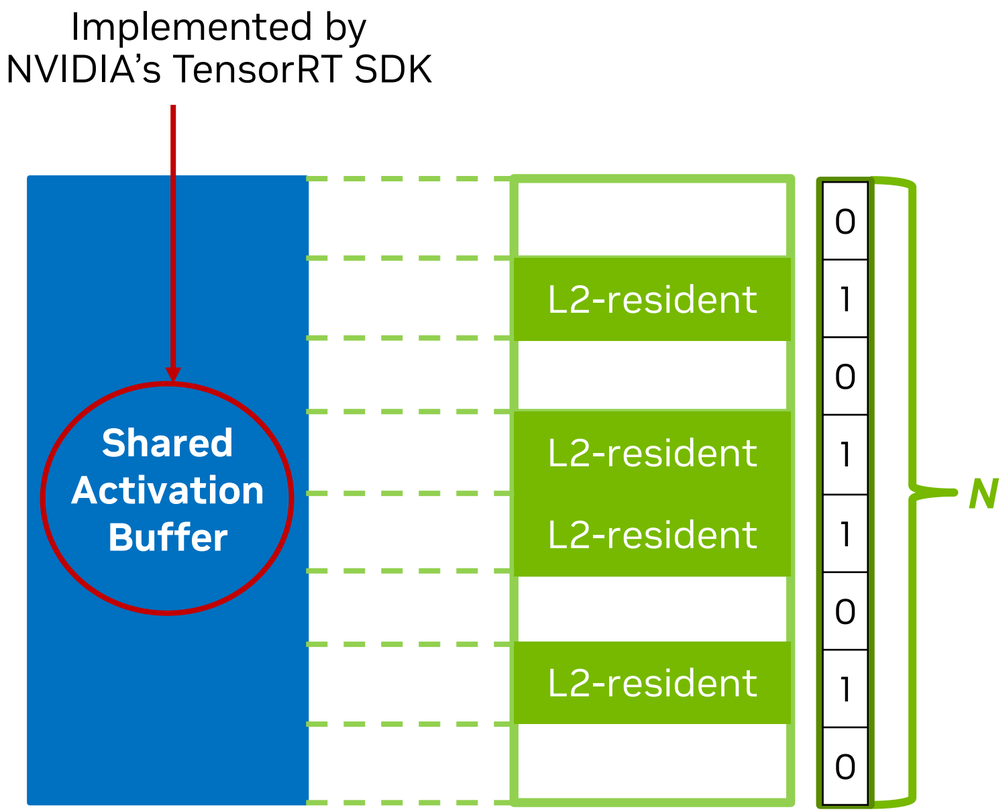
Pros:

- More flexible L2 cache replacement.
- Inline with increasing L2 capacity.
- Well-suited for deep learning workloads.

Cons:

- Requires non-negligible programming effort.
- Difficult to reason about and optimize.
- Sensitive to L2 capacity & memory footprint.

How To Program L2 Residency Control for DL Inference?



L2 residency selections within the shared activation buffer.

An L2 residency configuration can be represented by an N-dimensional binary vector (1MB per bit in our case).

The maximum number of L2-resident positions cannot pass the HW limit.

Optimization options:

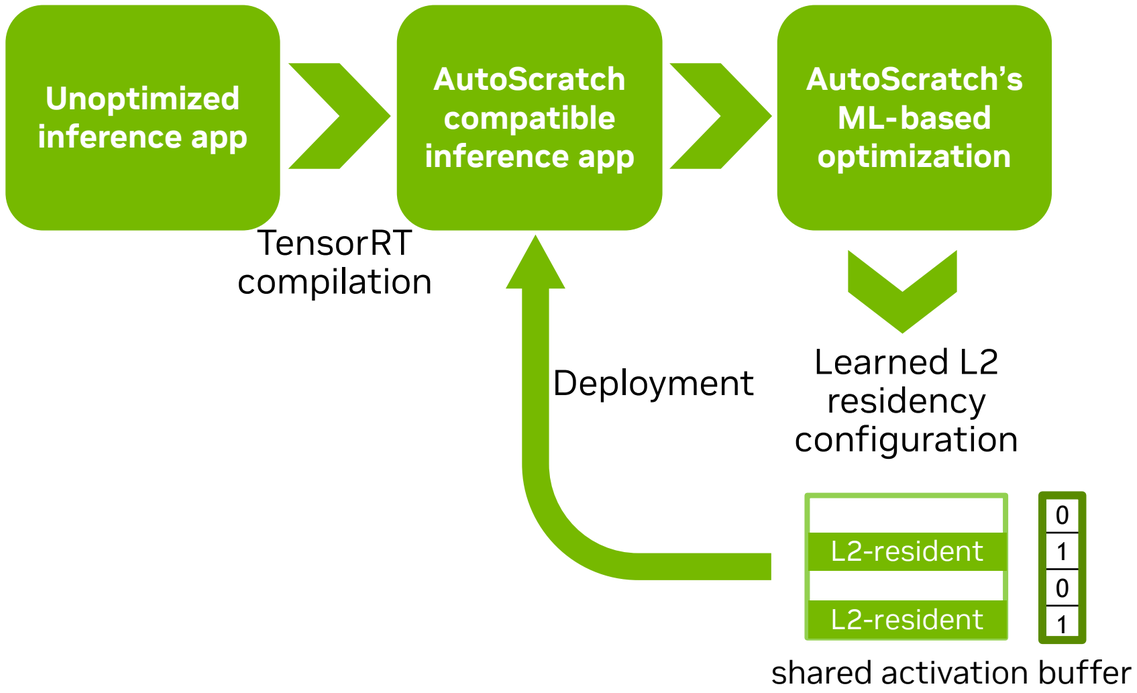
1. Exhaustive search on the activation memory address space.
   - The search space grows exponentially with N.
2. Heuristic-based tuning.
   - Need in-depth investigation on HW & workload characteristics.

3. This work: Automatically learn and optimize L2 residency configurations.

- **No need to understand detailed HW & workload characteristics.**
- **Could discover better solutions than heuristics-based ones.**

AutoScratch Framework

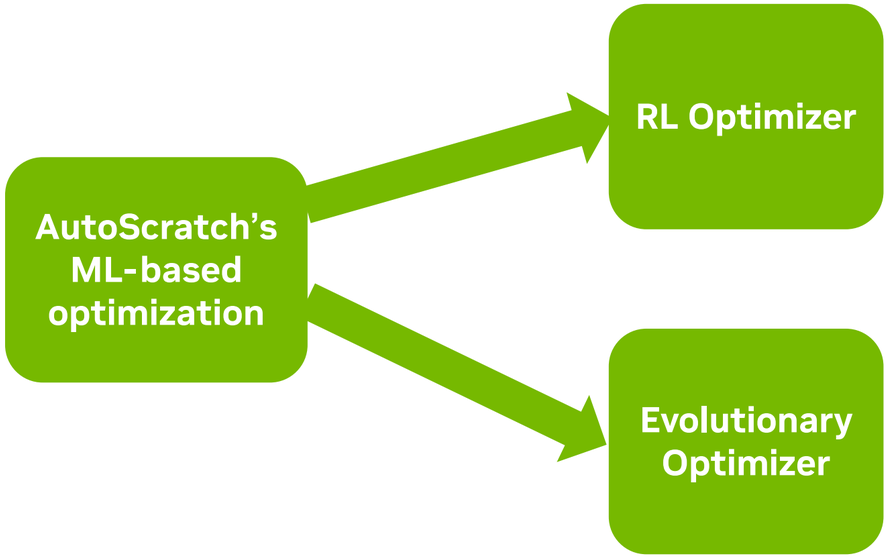
Integration with NVIDIA's TensorRT SDK



The AutoScratch optimization flow.

- Works with arbitrary inference applications without any code modifications.
- Trained once per combination of an application and HW configuration.
- Once trained, the optimal L2 residency control configuration is stored and deployed (no additional training needed).
- The training process can also take place on-the-fly after deployment to hide the training overhead.

Options for ML-based Optimization in AutoScratch



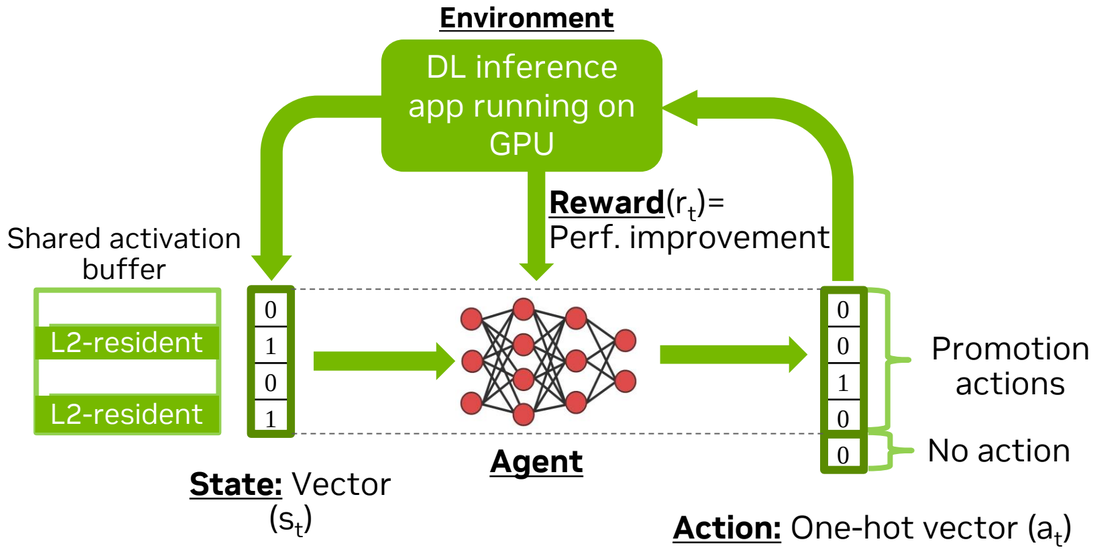
We explore two types of ML-based optimizers in AutoScratch:

- Optimizing with Reinforcement Learning (RL)
  - We leverage off-the-shelf Proximal Policy Optimization (PPO) algorithm [1].
- Optimizing with Evolutionary algorithms (EA)
  - We leverage the regularized evolutionary algorithm [2] originally developed for neural architecture search.

1) Schulman et al. "Proximal policy optimization algorithms", ArXiv, 2017.
2) Real et al. "Regularized Evolution for Image Classifier Architecture Search", AAAI, 2019.

RL Optimizer

Automatic optimization with Reinforcement Learning (RL)



RL-based optimization pipeline.

Environment: A TensorRT-compiled DL inference application with L2 residency control APIs.

Agent: A small MLP network consisting of two hidden layers.

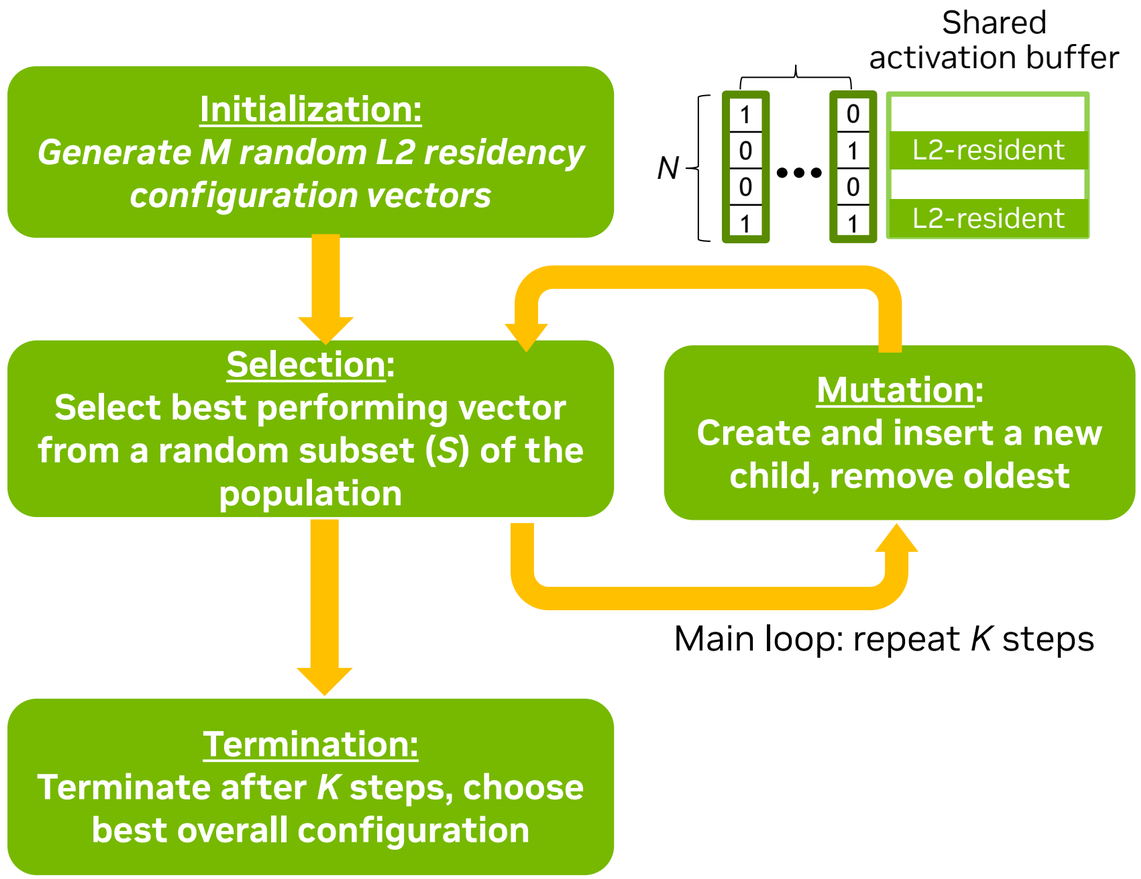
State: Current L2 residency configuration represented by an N-dimensional binary vector.

Action: An (N+1)-dimension one-hot vector to indicate which memory location to promote or taking no action.

Reward: Execution time difference before and after an action.

Evolutionary Optimizer

Automatic optimization with Evolutionary Algorithm (EA)



EA-based optimization pipeline.

1) **Initialization**: Generate the initial population of M random L2 residency configurations and put them into a FIFO queue.

2) **Main Loop**: Repeat the following re-generational steps until termination: (max 20,000 steps in our case)
   - Select the best-performing configuration from a random subset of the population.
   - Breed a child configuration through a mutation operation.
   - Insert the child configuration and remove the oldest one.

3) **Termination**: Choose the best one across all configurations.

AutoScratch Evaluation

Silicon based evaluation.

- NVIDIA's L4 GPU with 48MB of L2 capacity, a maximum TDP of 72W and 300GB/s of memory bandwidth.

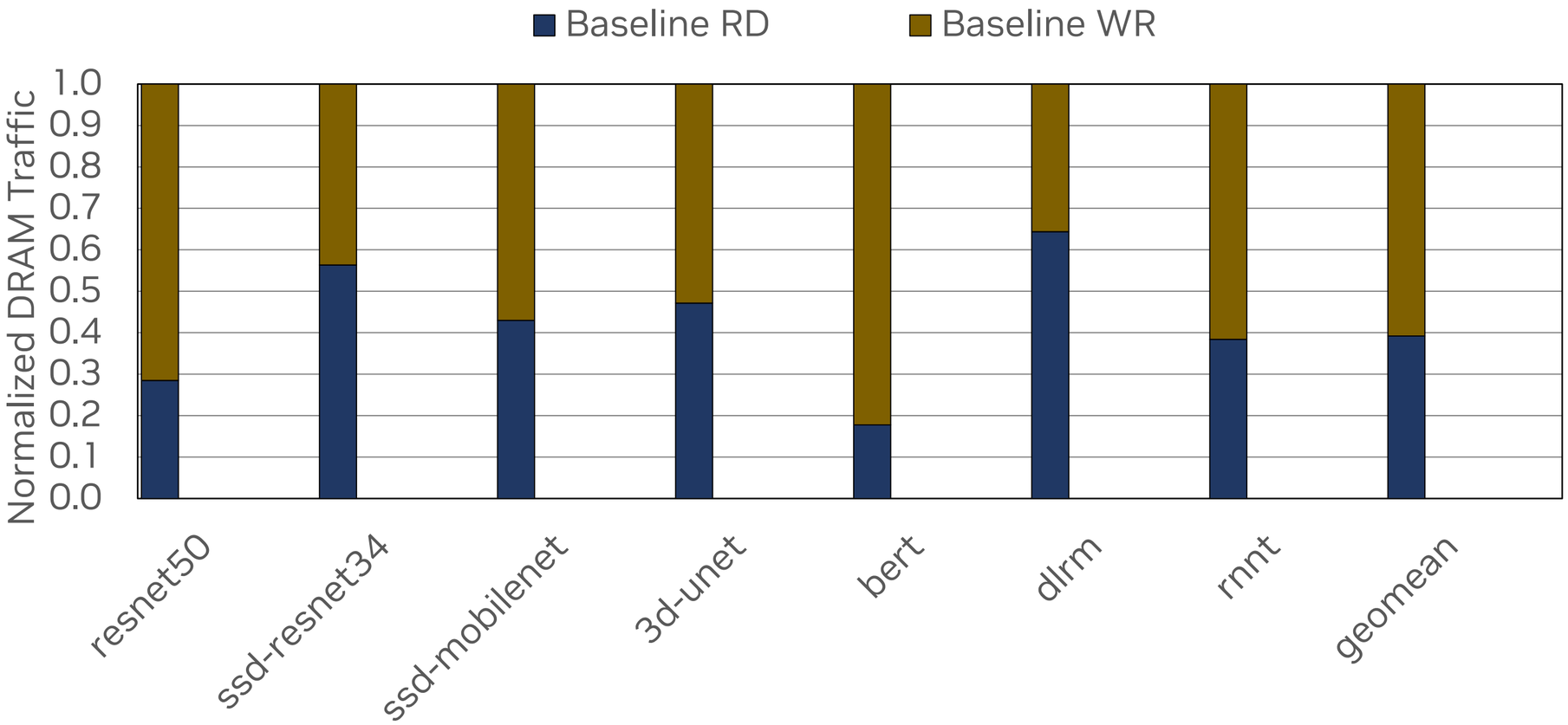
Comparing methods:

- **Baseline:** Default HW-based L2 management scheme with no L2 residency control.
- **AutoScratch-RL (AS-RL):** Using reinforcement learning as the ML optimizer in AutoScratch.
- **AutoScratch-EA (AS-EA):** Using evolutionary algorithm as the ML optimizer in AutoScratch.

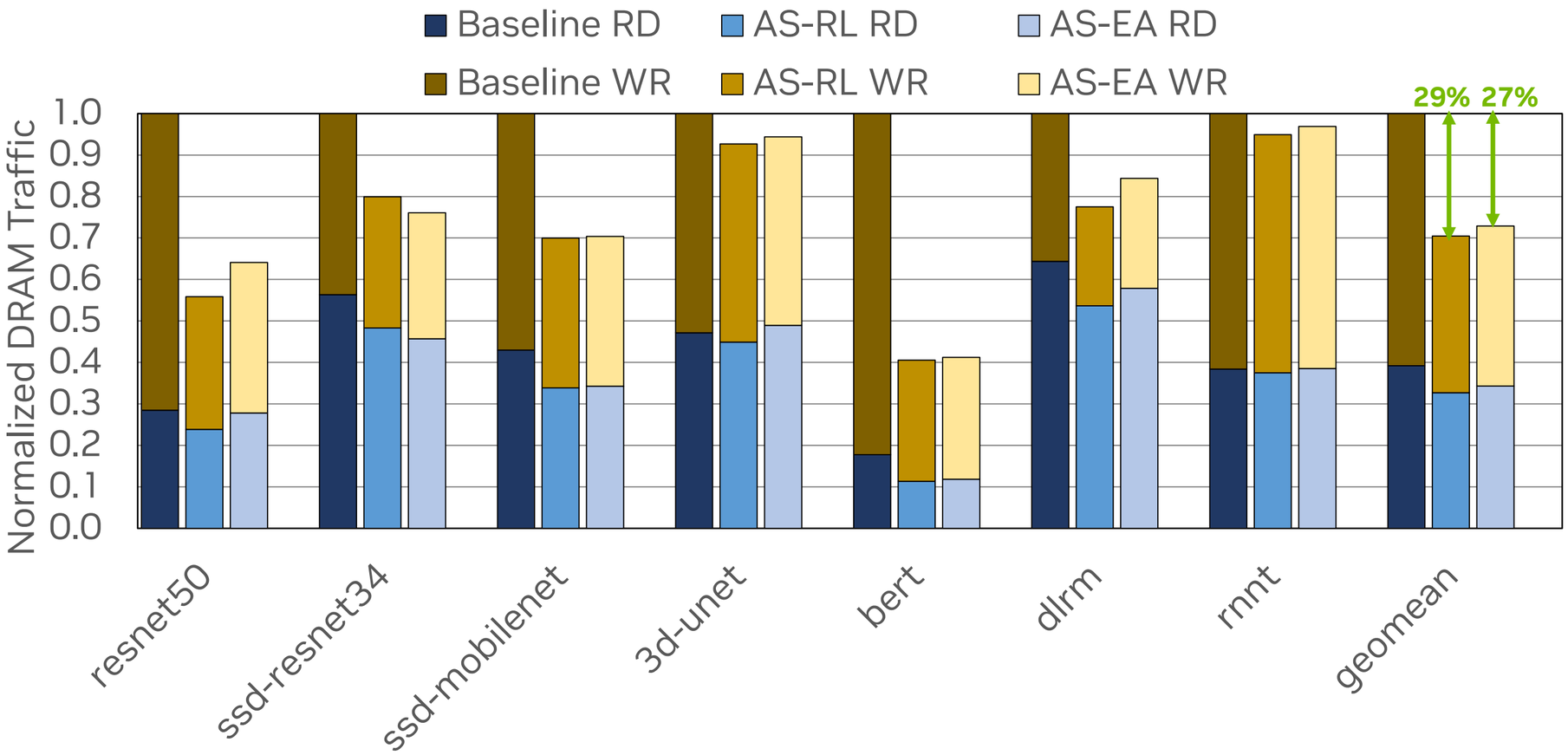
Evaluating on MLPerf inference applications.

Application	Precision	Batch size	Activation buffer size (MB)
resnet50	int8	32	63
ssd-resnet34	int8	6	104
ssd-mobilenet	int8	64	140
3d-unet	int8	1	278
bert	int8	32	81
dlrm	int8	51200	106
rnnt	fp16	2048	4175

DRAM Traffic Reduction



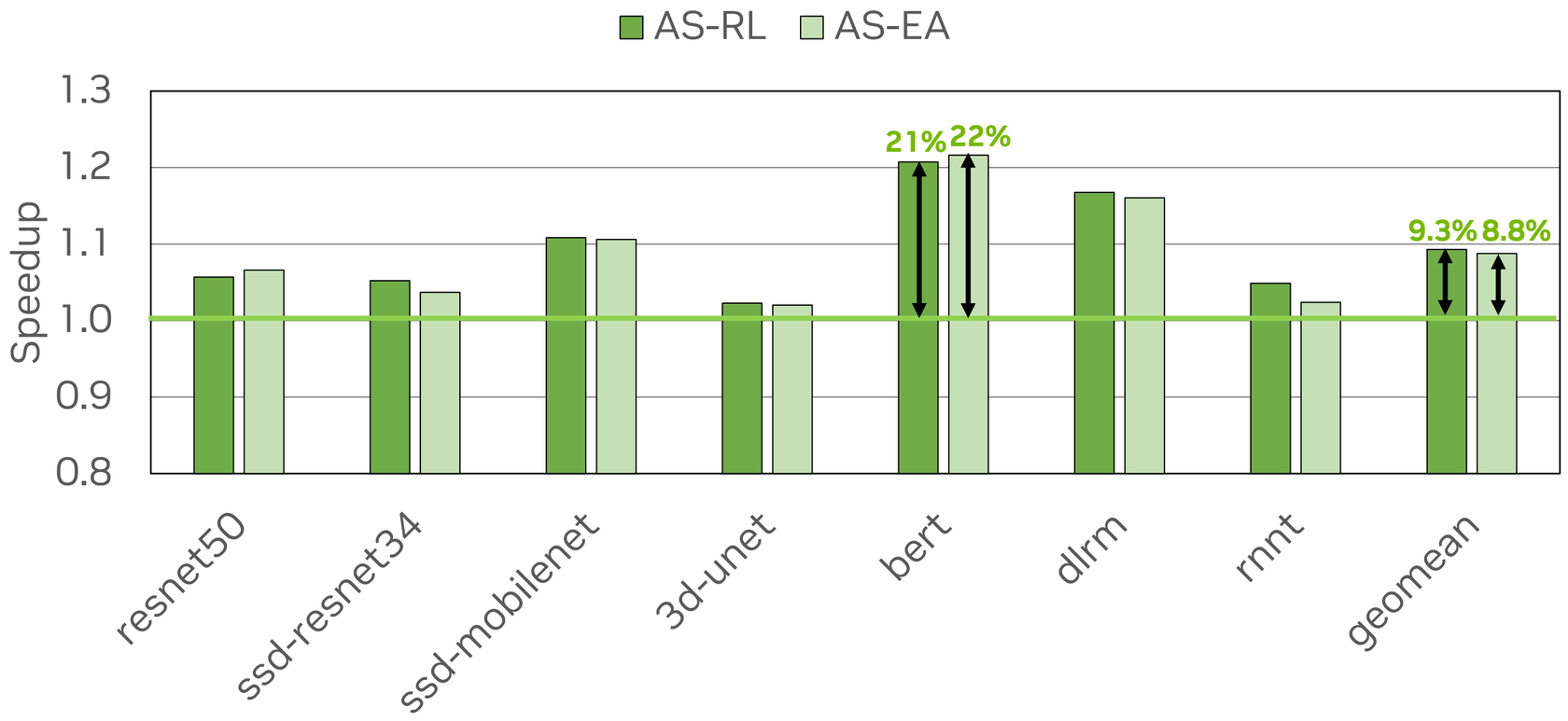
DRAM Traffic Reduction



- Both **AS-RL** and **AS-EA** result in significant (29% and 27%) DRAM traffic reduction.
- DRAM WR traffic reductions are more significant because AutoScratch effectively prevent activations from being written back to DRAM.

Performance Speedup

Measured on NVIDIA's L4 GPU Silicon



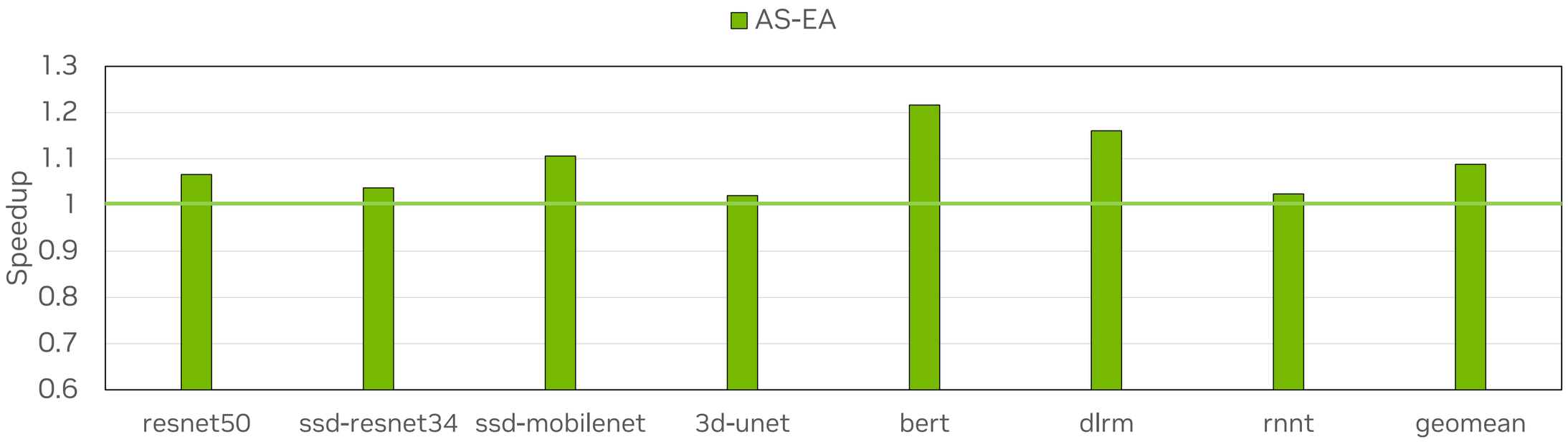
- Both **AS-RL** and **AS-EA** result in good (9.3% and 8.8%) performance speedup, up to 21% and 22% for *bert*.
- MLP-based workloads (*bert* and *dlrm*) benefit more from AutoScratch than Conv-based workloads because MLP operators are more memory-bound on GPUs.

The Cost of AutoScratch Training

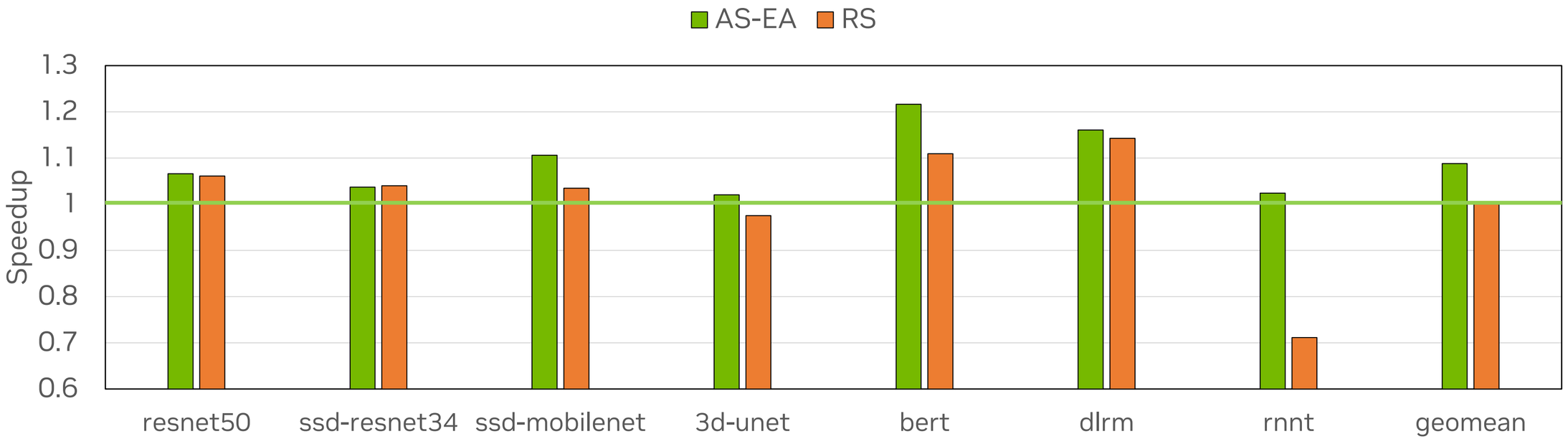
Workload	AS-RL (minutes)	AS-EA (minutes)	Time Reduction
resnet50	241	6.8	35x
ssd-resnet34	421	5.8	72x
ssd-mbnet	260	8.0	33x
3d-unet	390	4.4	88x
bert	765	10.2	75x
dlrm	238	5.4	44x
rnnt	227	2.5	90x
geomean	330	5.7	58x

- **AS-EA** results in 58x reduction in training time compared to **AS-RL**.
- The training time of **AS-EA** is comparable to TensorRT compilation time, making it practical for deployment.

Comparing Against Other Optimization Methods

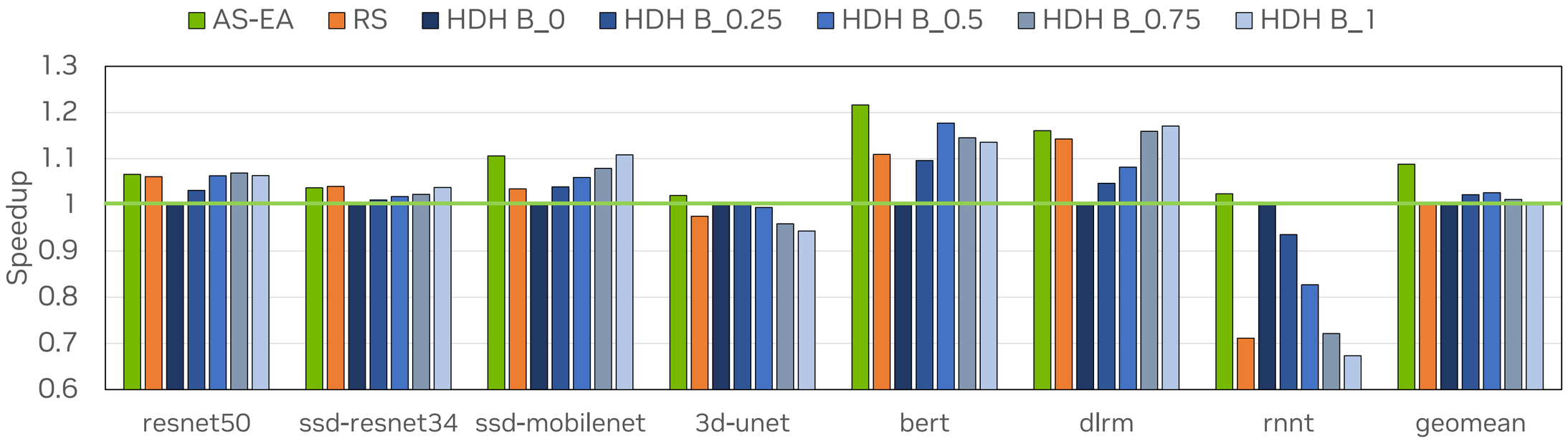


Comparing Against Other Optimization Methods



- Comparing method:
  - **Random Search (RS):** Picking the best L2 residency configuration within a random set.
- The geomean speedup of **RS** is close to 0 compared to the baseline, making it a useless solution.

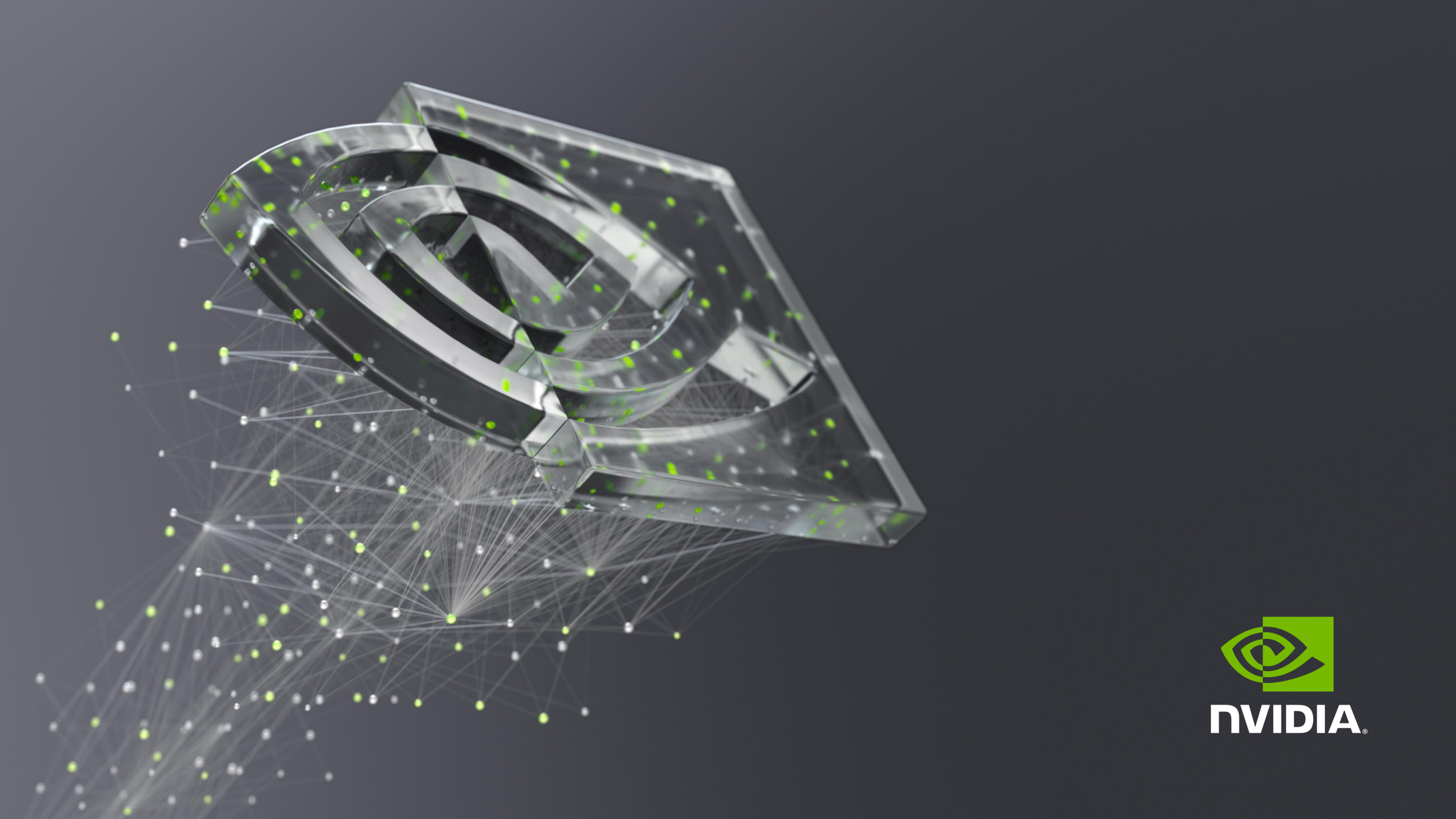
Comparing Against Other Optimization Methods



- Comparing method:
  - **Human-Designed Heuristic (HDH):** Designed by GPU experts at NVIDIA and integrated into the latest TensorRT SDK. An L2 residency budget ranging from 0 to 1 needs to be provided by the user.
- **HDH** can perform well, but additional parameter tuning on the L2 residency budget is required.

Conclusions

- Applying ML for cache management is promising for improving performance and energy efficiency.
- AutoScratch achieves a DRAM traffic reduction of **29%** and performance speedup of **9%** for MLPerf inference workloads on an NVIDIA's L4 GPU.
- AutoScratch with Evolutionary Optimizer is **58x** faster than AutoScratch with RL Optimizer while providing similar performance, making it a suitable choice for practical deployment.


NVIDIA