Accelerating Inpainting: Benchmarking Fine-Tuning Performance

Jooho Kim[1], Sunyoung Park[2] and Pranav Handa[3]

[1] *Institute for Disaster Resilient Texas (IDRT), Texas A&M University, College Station, TX*
[2] *Department of Computer Science and Engineering, Texas A&M University, College Station, TX*
[3] *Department of Computer Science and Engineering, Texas A&M University, College Station, TX*

Introduction

Diffusion-based inpainting models such as **Stable Diffusion** have shown strong potential for restoring occluded or damaged visual information in post-disaster imagery. However, fine-tuning these models remains computationally intensive, posing challenges for researchers aiming to deploy them at scale. Understanding how different accelerator architectures handle these workloads is essential for improving both efficiency and accessibility in large-scale geospatial analysis.

This study benchmarks the fine-tuning performance of a diffusion-based inpainting model across several modern AI accelerators, including NVIDIA A40, A100, and H200. We focus on evaluating throughput, runtime efficiency, and memory utilization to identify optimal hardware–software configurations for inpainting fine-tuning.

Future research will expand this work to explore model convergence behavior, reconstruction quality, and energy efficiency—key factors for scaling generative models in real-world disaster analytics and large supercomputing environments.

Method

Dataset: ~6,500 geolocated street-view building images

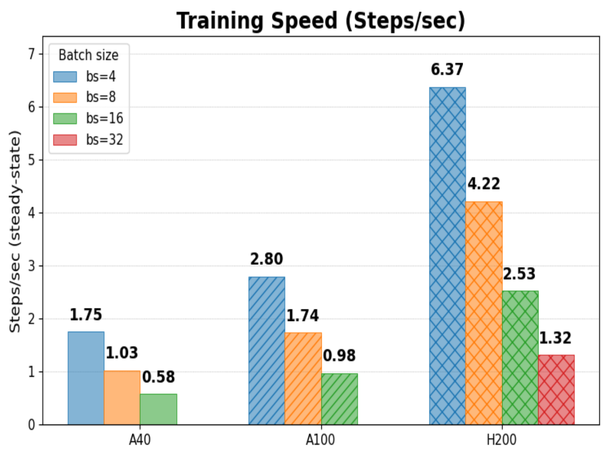
Model: UNet (from Stable Diffusion v1.5) fine-tuned for location-aware inpainting

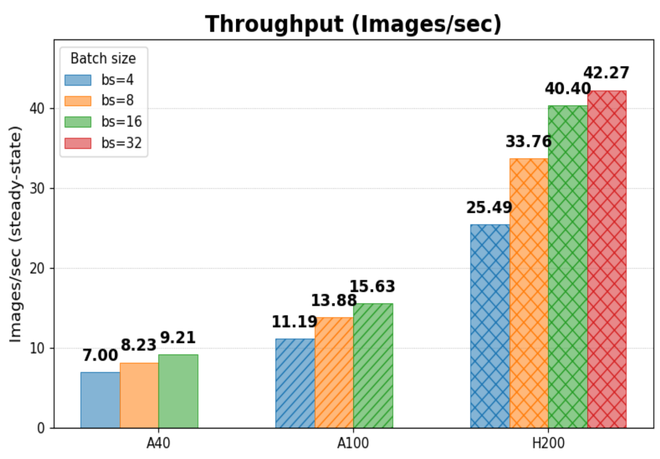
Cluster: NVIDIA A40/A100 GPU on the GRACE supercomputer at Texas A&M High Performance Research Computing (HPRC), and NVIDIA H200 GPU on the Delta supercomputer at the National Center for Supercomputing Applications (NCSA) via the NSF ACCESS program.

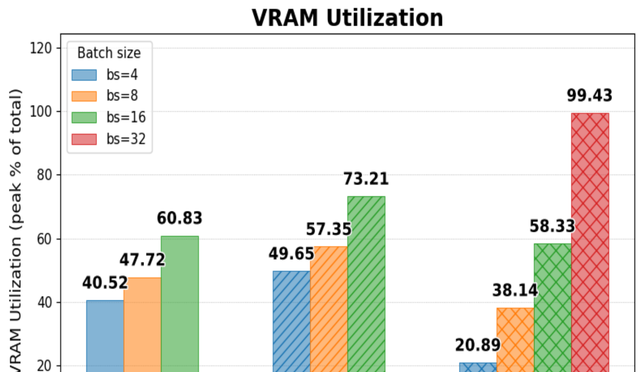
GPU architecture

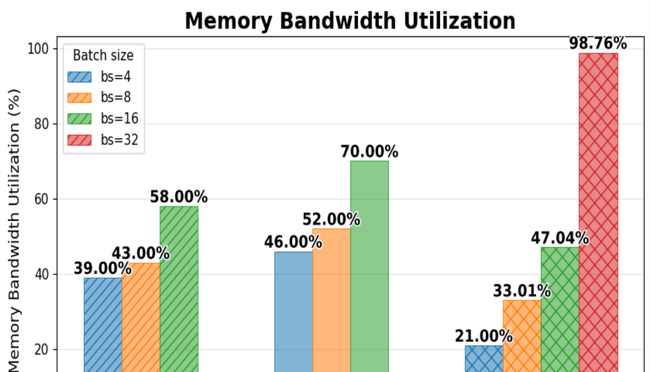
	NVIDIA A40	NVIDIA A100	NVIDIA H200
GPU Memory	48 GB GDDR6	40 GB HBM2	141 GB HBM3e
Memory Bandwidth	696 GB/s	1555 GB/s	4.8 TB/s
BFLOAT16 Tensor Core	149 TFLOPS	312 TFLOPS	362 TFLOPS
Architecture	Ampere	Ampere	Hopper

Evaluation Results





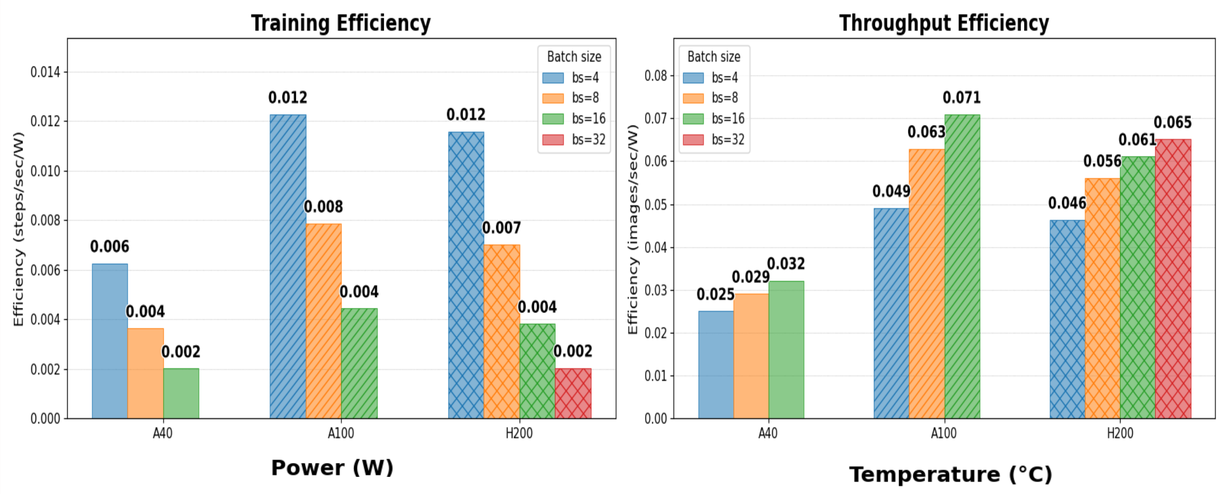




	A40	A100	H200
4	98.87	90.27	94.96
8	99.78	92.59	98.87
16	99.92	99.34	99.25
32	-	-	93.69

	A40	A100	H200
4	4.12	2.57	1.13
8	3.50	2.07	0.85
16	3.13	1.84	0.71
32	-	-	0.68

Efficiency and Thermal Metrics



	A40	A100	H200
4	280	228	550
8	283	283	602
16	286	286	661
32	-	-	649

	A40	A100	H200
4	66.4	57.5	49.4
8	66.3	58.4	53.0
16	51.4	57.7	58.5
32	-	-	50.9

Measured via NVML API (steady-state average)

Discussion

- A100 is 1.6 - 1.7 x faster than A40
- **H200** is **2.3–2.6× faster than A100** and **3.6–4.4× faster than A40**
- **A40** : at bs=16, VRAM = 60%, BW = 58%, **primarily compute-limited**
- **A100** : at bs = 16, VRAM = 73%, BW = 70%, **primarily compute-limited**
- **H200** : at bs = 32, VRAM = 99%, BW = 99%, while GPU util = 94%, **memory bandwidth/capacity limited**

Input-Pipeline Optimizations on H200

Run	Step /Sec	Img /Sec	Time (h)	GPU util(%)	Mem BW util (%)	VRAM util (%)	Power (W)	Temp (°C)	Steps/ sec/W	Images/ sec/W
H200 bs16 (baseline)	2.52	40.4	0.713	99.25	47.03	58.33	661	58.51	0.0038	0.0611
H200 bs16 (optimized)	2.65	42.4	0.679	99.29	35.02	41.2	630	57.57	0.0042	0.0673
Δ (%)	+5%	+5%	-4.8%	+0.04%	**-25.55%**	**-29.4%**	-4.67%	-1.6%	**+10.2%**	**+10.2%**

- Optimized the input pipeline and memory transfers by using a pinned-memory DataLoader with more workers and prefetching, non-blocking host-to-device copies, and channels_last tensors.
- The optimized run is **~5% faster (steps/sec), ~5% higher throughput, and ~10% more energy-efficient**, while drawing **~31 W less** on average

Conclusion/Future Direction

Optimal Hardware-Software Configuration: For this workload, A40 and A100 are most efficient at batch size around 16 where GPU utilization is near 99 percent without saturating memory. On H200, batch sizes in the range of 16 to 24 are preferable, while batch size 32 saturates memory bandwidth and capacity and lowers training speed.

Future Direction : We will further optimize the H200 setup by profiling the pipeline with Nsight Systems/Compute to pinpoint dataloader and memory stalls, then applying targeted fixes (caching, prefetching, data-format tweaks).

References/Acknowledgements

Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing.

This work used the Delta system at the National Center for Supercomputing Applications through allocation CIV250023 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

Jooho.kim@tamu.edu