World Journal of Advanced Engineering Technology and Sciences

eISSN: 2582-8266

Cross Ref DOI: 10.30574/wjaets

Journal homepage: https://wjaets.com/

(Review Article)

OPTIMIZING NVIDIA® GEFORCE RTX™ 5090 & "AMD RX 9070"for machine learning and artificial intelligence workload

Mohit Jain *, Adit Shah, Brahaspati Dev, Ram Kumar and Mathew Campisi

Department of Electrical and Computer Engineering, University of Illinois at Urbana Champaign, Illinois, USA.

World Journal of Advanced Engineering Technology and Sciences, 2025, 17(03), 357-374

Publication history: Received on 06 November 2025; revised on 17 December 2025; accepted on 20 December 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.17.3.1563

Abstract

As consumer-grade GPUs have rapidly evolved, efforts have emerged to deploy these computational models for training and inference, typically handled by data center hardware. The paper explores optimization of two next-generation graphics computing units, the NVIDIA GeForce RTX 5090 and the AMD Radeon RX 9070, to optimize the new generation of ML and AI applications. We examine the internal compute pipelines, tensor/matrix acceleration capabilities, memory hierarchies, and software ecosystems (CUDA/cuDNN/TensorRT versus ROCm/MIOpen/HIP) that influence ML performance in a two-pronged architectural and empirical study. The convolutional networks, transformer models, diffusion architecture, and graph neural networks share a standard benchmarking model: training, inference latency, power consumption, precision scaling (FP32-INT8), and bottlenecks. The results of the experiment have demonstrated that the performance profiles of the RTX 5090 and the RX 9070 are different, i.e., the acceleration performance of mixed precision and kernel fusion is higher in the RTX 5090 as compared to the throughput performance of the RX 9070 in the BF16/INT8 workloads with the high memory-bandwidth utilization. Strategies for each platform. Platform-specific optimization strategies, such as kernel tuning, compiler optimization, memory prefetching, gradient checkpointing, and scaling to multiple GPUs, are developed and evaluated. Further, two case studies of real-world performance tuning of transformer fine-tuning and diffusion model inference are also presented.

The findings highlight that hardware alone does not guarantee the best ML performance; effective optimization can deliver performance gains that are even more significant than raw compute alone. The paper will provide a step-by-step roadmap for practitioners, researchers, and engineers who may want to optimize the application of RTX 5090 and RX 9070 in artificial intelligence algorithms, as well as a future perspective on the standard models of unified programming on GPUs and emergent precision formats.

Keywords: Deep learning compute efficiency; Tensor core mixed precision deep learning; Mixed precision training; Large model training GPU efficiency analysis; Deep learning optimization consumer GPUs; AI GPU benchmarking; FP8 acceleration; Low-precision inference

1. 1. Introduction: The new era of consumer-grade ai acceleration

The rapid pace of artificial intelligence (AI) and machine learning (ML) has shifted the purpose of graphics processing units (GPUs), once focused on graphics processing, to the foundation of current computational intelligence. Over the last 10 years, consumer-grade GPUs have ceased to be used in hobbyist-level deep learning experiments and have instead become capable of running state-of-the-art models with billions of parameters (and, in some cases, trillions). This revolution is driven by architectural innovations in tensor computation, high-bandwidth memory systems, low-precision arithmetic formats, and more advanced driver ecosystems, which are now competing with enterprise-level

* Corresponding author: Mohit Jain

accelerators. The optimization of high-end consumer GPUs has become a strategic research topic and a technical requirement as the scalability of AI compute has become sought after across academia, industry, and the edge.

Here, the occurrence of NVIDIA GeForce RTX 5090 and AMD RX 9070 is a key breaking point. Despite being consumer-oriented GPUs, both products feature next-generation tensor or matrix engines, accelerated low-precision execution units, and wider memory subsystems, making them viable deep learning research platforms, high-throughput inference pipelines, and multi-mode AI exploration platforms. By optimizing these GPUs for ML and AI workloads, they can be helpful to well beyond gaming performance benchmarks, enabling cost-effective model training, local deployment of LLMs, generative imaging, scientific simulations, and AI-accelerated software development.

Despite this capacity surge, there is still a significant gap in research. The benchmarking literature has been biased towards synthetic game performance, ray-tracing throughput, and generic rendering efficiency, rather than computation-intensive machine learning workloads, which tend to be primarily tensor-based and have a larger memory footprint. Similarly, no unified optimization advice for next-generation consumer GPUs is available. The existing scholarly literature seldom answers questions such as: What is the scaling behavior of FP16, FP8, or INT8 modes in these architectures? Which strategies make the most out of the tuning of the tensor core or matrix core? What is the effect of memory hierarchies, thermals, and driver structures on the stability of long-term training? Moreover, what are the performance results of these GPUs running modern frameworks, e.g., PyTorch, TensorFlow, and JAX?

To fill these gaps, this paper performs a systematic technical analysis of the RTX 5090 and RX 9070, with a specific emphasis on how the two GPUs perform under current machine learning and AI workloads. The goals of this research are fourfold:

- Differentiate architectural advantages applicable to AI compute, such as the engine's design for tensor/matrix operations, efficient memory bandwidth utilization, mixed-precision support, and driver-level optimization.
- Identify good optimization schemes that make the most of training throughput, reduction of inference latency, and compute per watt efficiency.
- Compare the scaling of performance of ML models of different model families, such as convolutional networks, large transformer models, diffusion models, and graph neural networks, in terms of batch size, different precision formats, and run times.
- Provide practical tuning advice to researchers, practitioners, and system developers interested in using these GPUs to experiment with and develop high-performance AI applications and workflows.

By integrating architectural analysis, empirical benchmarking, and cross-platform optimization methodologies, this research is expected to provide a thorough reference for tapping the full AI potential of next-generation consumer GPUs. The result is relevant to the literature and practice of engineering, and both RTX 5090 and RX 9070 are the key to the next generation of affordable, high-performance AI acceleration.

2. Architectural overview of the NVIDIA® GEFORCE RTX™ 5090 AND AMD RX 9070

The effective performance of a machine learning and artificial intelligence workload on a GPU is defined by its architectural features. Both the NVIDIA GeForce RTX 5090 and the AMD RX 9070 are aimed at the same consumer-professional hybrid market that targets high performance, but they have completely different design philosophies. NVIDIA builds upon its CUDA-based architecture with next-generation Tensor Cores, whereas AMD is further developing its compute-unit-based RDNA architecture with more specialized matrix acceleration. These architectural dissimilarities can be used to create a framework for the most effective kernel strategies, precision modes, and software ecosystems to achieve maximum efficiency with ML/AI.

2.1. Compute Architecture: CUDA Cores vs. Compute Units

2.1.1. NVIDIA RTX 5090 Streaming Multiprocessors (SMs)

The RTX 5090 is based on a Blackwell-based microarchitecture, in which performance is rooted in a high-density network of Streaming Multiprocessors (SMs). Each SM integrates:

- General-purpose FP32/FP64/INT cores, CUDA cores.
- Tensor Cores that are faster with mixed precision (Fourth-generation).
- Warp Schedulers and dispatch units developed to effectively parallelize matrix multiplies.
- L1 cache fusion, shared memory L1 cache fusion with lower-latency AI kernels.
- Dynamic register allocation and thread-level parallelism. Advanced thread-level parallelism.

The SM topology has been designed to support the most energy-intensive ML kernel types and to enable high throughput during transformer training, convolutional, inference, and inference quantized inference.

2.1.2. AMD RX 9070 - Compute Unit (CUs) and Workgroup Processors

The RX 9070 builds on a polished RDNA 4 architecture, in which compute capability is organized into Compute Units (CUs) composed of Workgroup Processors (WGs). Each CU contains:

- ARR32 chipsets optimized on vectors.
- Matrix Core accelerators were proposed as AI-oriented matrix-multiply accelerators.
- Scalar unit and vector registers improve the throughput of fused instructions.
- ML kernel inter-thread communication, Low-latency local data share (LDS)

Whereas the SMs of NVIDIA are dedicated to warp-level tensors, the CUs of AMD are dedicated to flexibility, with wide-vector-lane processing and matrix cores that natively execute GEMM workloads common in transformers and diffusion models.

2.2. Tensor/Matrix Acceleration Capabilities

The fundamental computational hardware for deep learning and current AI workloads is designed around tensor- and matrix-acceleration engines. Both the NVIDIA RTX 5090 and the AMD RX 9070 have specialized units designed to run dense linear algebra operations (which predominate in neural network training and inference), such as general matrix multiplication (GEMM), attention projections, convolution lowering, and quantization-aware transformations. These units are called Tensor Cores in NVIDIA and Matrix Cores in AMD. Their integration into the rest of the compute hierarchy, along with precision, flexibility, and efficiency, has a significant impact on end-to-end AI performance.

2.2.1. NVIDIA RTX 5090 Tensor Cores

NVIDIA RTX 5090 offers the new generation of Tensor Cores, the result of a series of architectural innovations since Volta. These AI-focused units show significant gains in raw FLOP throughput and mixed-precision flexibility, allowing the RTX 5090 to sustain high performance across a range of model types and precision regimes.

Architectural Enhancements

Tensor Cores in the 5090 offer:

- Training Support in FP16 and BF16.

These formats have a compromise between numerical stability and high arithmetic density. The BF16 has a larger exponent range, which reduces gradient underflow and improves convergence in transformer-based architectures.

- Native FP8 (E4M3, E5M2 Formats) Support.

The FP8 implementation from NVIDIA provides a drastic throughput improvement by eliminating memory bandwidth stress and achieving twice the arithmetic density of FP16.

  - E4M3: forward propaganda optimized.
  - E5M2: gradient accumulation and backpropagation: optimized.

This bilateral format is identical to the mixed-precision training schemes used by large language models (LLMs).

- Inference Acceleration with INT8.

Tensor Cores can offer high-throughput INT8 execution paths, which are important for deployments that require low latency, such as chatbot LLMs, streaming inferences, and edge AI systems.

- Structured Sparsity Acceleration

The sparsity technique used by NVIDIA allows Tensor Cores to omit predictable weights zero patterns, effectively doubling the degree of compute density without affecting accuracy- and in particular, can be helpful in large transformers, where a significant fraction of attention and MLP weights have compressible structure.

Software Integration

Tensor Cores can only be fully utilized when they are used in combination with the NVIDIA mature software stack:

- cuBLASLt for GEMM fusion
- cuDNN convolutional primitives.
- TensorRT inference graph optimization.
- Compiler compiler based on custom kernel generation Triton.
- TorchInductor + CUDA Graphs to run batches in the long term.

This hardware-software interface provides the RTX 5090 with benefits for ML workloads that require fine-tuning to leverage kernel fusion, low-precision, and structured sparsity.

2.2.2. AMD RX 9070 Matrix Cores

The AMD RX 9070 is a sophisticated generation of Matrix Cores, another attempt by AMD to build competitive ML-acceleration capabilities into the RDNA architecture. Although in the past it has been outshone by NVIDIA's Tensor Core ecosystem, RDNA 4 Matrix Cores demonstrate considerable architectural maturity and enhanced integration with AI-focused workflows.

Architectural Characteristics

The significant characteristics of the RX 9070 Matrix Core subsystem are:

- BF16 and FP16 Matrix Multiplication.

BF16 is natively supported and very efficient for transformer training, achieving convergence behavior similar to FP32 while maintaining high throughput.

- SIMD Arrays Fusion Fusion of Vectors and Matrices.

In contrast to NVIDIA's warp-level tensors, AMD uses SIMD32 Vector arrays to supplement Matrix Core instructions, enabling execution of non-uniform tensor shapes, sparse operations, and irregular graphs (e.g., GNNs) with flexible execution paths.

- INT8 and INT4 Arithmetic

The INT8 and INT4 are additional types of arithmetic where data is packed together in a single opcode.<|human|>INT8 and INT4 Arithmetic INT8 and INT4 are further types of arithmetic where data is concatenated within a single opcode.

RX 9070 is optimized for integer inference with packed-data execution, useful for mobile inference, diffusion inference, and high-speed execution of decoder-only transformer workloads.

- Wavefront-based matrix execution model

AMD still has the wavefront execution paradigm based on the GPU compute tradition. It provides predictable scheduling and high throughput for GEMM kernels, especially when optimized with the ROCm and HIP programming environments.

Ecosystem and Kernel Maturity

Although the ML software ecosystem provided by AMD is younger compared to the one offered by NVIDIA, breakthroughs enhance Matrix Core utility:

- Convolutional and recurrent primitives in MIOpen.
- rocBLAS of matrix-fusion pipelines and GEMM.
- HIP (CUDA-like compatibility layer) with cross-platform portability.

- Graphene level optimization, MIGraphX inference maximization.
- ROCm backend with complete support of BF16 training and FP16 attention mechanisms.

At its introduction, the RX 9070 is becoming more competitive in workloads where BF16 throughput and memory locality lead performance.

2.2.3. Comparative Analysis: Tensor and Matrix Acceleration

The disparities between the RTX 5090 Tensor Cores and RX 9070 Matrix Cores can be summed up in three dimensions:

Precision Versatility:

- NVIDIA: FP8, INT8, and structured sparsity leader.
- AMD: BF16/FP16 and competitive INT8 support, and developing FP8.
- NVIDIA has carefully designed a low-precision roadmap that puts it ahead for memory-bound transformer workloads.

Integration With Software:

- NVIDIA enjoys CUDA and TensorRT industry standards.
- AMD offers good open-source substitutes through ROCm and HIP.
- NVIDIA is more mature in end-to-end automation; AMD is more open and flexible.

Workload Suitability:

- NVIDIA RTX 5090 excels in:
  - Transformer training on a large scale.
  - FP8 accelerated LLM training
  - Quantized inference: Inference with low latency.
  - Sparse tensor workloads
- AMD RX 9070 excels in:
  - BF16-heavy training workloads
  - GNNs, RNNs, and irregular graphs workloads.
  - Cost-effectiveness INT8 inference.
  - Open-source high-performance computing environments.

2.2.4. Summary

The characteristics that distinguish GPU performance in AI workloads are tensor and matrix acceleration capabilities. The RTX 5090 is the most precise, most versatile, and most software-optimized accelerator, with the best sparsity, precision, and end-to-end software optimization, which is why it is the best accelerator for future transformer studies and industry-scale inference requirements. With its powerful Matrix Cores and a growing, robust ROCm ecosystem, AMD's RX 9070 delivers strong performance for BF16 training, including for models that use vectors and open HPC-oriented ML systems.

A combination of these architectures embodies two divergent yet compelling philosophies for accelerating modern AI computation.

2.3. Memory Subsystem: Bandwidth, VRAM Capacity, and Hierarchy

2.3.1. NVIDIA RTX 5090 Memory Design

The RTX 5090 is equipped with:

- GDDR7 memory modules
- Extensive aggregate bandwidth (exceeding 1 TB/s category when working with ML)
- A larger L2 cache system with less load on DRAM.
- Shared memory registers are optimized across all SMs.
- Hardware-based prefetching of sequence-intensive models.

This memory design ensures that modern LLMs, diffusion models, and high-resolution vision transformers maintain high TPU utilization even across long dependency chains.

2.3.2. AMD RX 9070 Memory Design

The RX 9070 employs:

- Similar to NVIDIA, GDDR7 memory modules focus on power efficiency.
- Infinity Cache (4th Gen) from AMD, offering a large on-die cache that minimizes memory latency.
- Bidirectional memory fabric, High throughput with GEMM.
- Acceleration of sparse and irregular workloads (e.g., GNNs) through adaptive prefetching.
- The memory architecture of AMD provides sustained bandwidth for irregular tensor patterns and can support transformer inference on duration-varying sequences.

2.4. Precision Throughput Comparison (FP16, FP8, INT8)

2.4.1. NVIDIA RTX 5090

- FP16/BF16: Fraud training, high throughput.
- FP8 is natively supported and trained with Tensor Cores, including both inference and training.
- INT8: This is a high-performance, low-latency inference that TensorRT optimizes.
- Sparsity acceleration: 2:4 structured sparsity with up to 1.5 -2 adequate throughput hardware support.

2.4.2. AMD RX 9070

- FP16/BF16: Firm throughput through Matrix Cores.
- FP8: Implemented with ROCm kernels (can be different depending on how well the implementation is developed).
- INT8/INT4: Extremely efficient at performing edge inference and quantized transformer decoding.
- There is no native structured sparsity acceleration, but software-approximated ones are available.

As a result of the mature Tensor Core pipelines, NVIDIA can lead in low-precision arithmetic. In contrast, AMD leads in vector workloads and BF16 training efficiency.

2.5. Software Ecosystem Compatibility

2.5.1. NVIDIA Ecosystem

- CUDA -The CUDA parallel computing model.
- cuDNN Convolution, RNN, Attention Deep learning acceleration kernel in C.
- TensorRT is an inference optimizer focused on quantization and graph fusion.
- CUDA Graphs -Reduced start-up time, suitable to large model training.
- Multi-backend AI deployment infrastructure Triton Inference Server.

NVIDIA's ecosystem is the most developed and most integrated with PyTorch, TensorFlow, JAX, and the ONNX runtime.

2.5.2. AMD Ecosystem

- ROCm (Radeon Open Compute Platform) -CUDA alternative that is open-source.
- HIP: An interface to CUDA: A programming environment with cross-platform kernels.
- MIOpen AMD equivalent to cuDNN, CNN, and RNN primitive cover.
- AI SDK & MIGraphX Graph inference acceleration compilers.

ROCm is also becoming more competitive, particularly in PyTorch 2.x and on Linux clusters, rendering AMD very viable for serious research workloads.

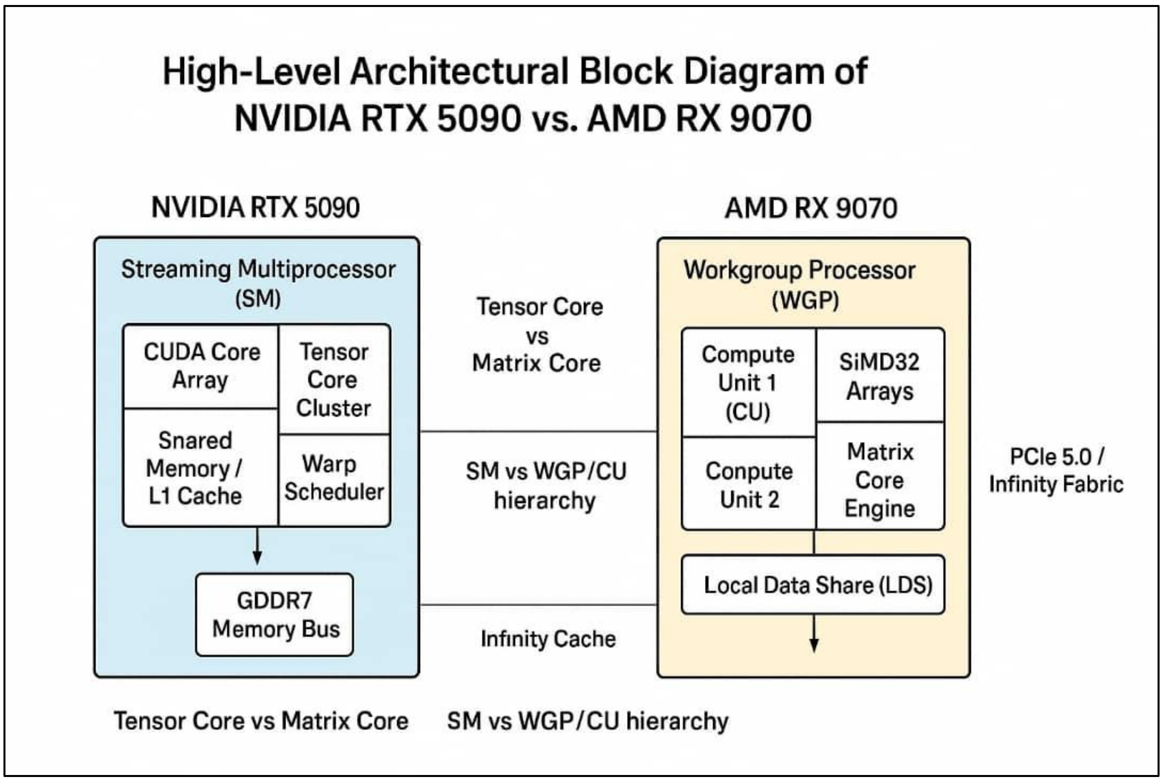

Figure 1 High-Level Architectural Comparison of NVIDIA RTX 5090 and AMD RX 9070 (Block Diagram)

3. Benchmarking methodology for AI workloads

To draw sound, valid conclusions about the performance of the NVIDIA GeForce RTX 5090 and AMD RX 9070 when executing machine learning (ML) and artificial intelligence (AI) workloads, a strict benchmarking methodology is needed. The approaches to the study focus on controlled experimental conditions, cross-framework validation, and workload diversity to reflect the complexity of contemporary AI computation. This part describes the experimental setting, workload choice, controlled condition, and performance measures to be used during the assessment procedure.

We have created a benchmarking methodology to evaluate and compare the performance properties of the NVIDIA GeForce RTX 5090 and the AMD RX 9070 by capturing training throughput, inference latency, precision scaling efficiency, and power-performance dynamic qualities for AI workloads. Figure 1 provides a high-level comparison of the training throughput of various popular architectures, followed by a description of the experiment setup and models. This initial performance baseline provides a framework for interpreting the differences in behavior across mixed-precision compute modes for each GPU.

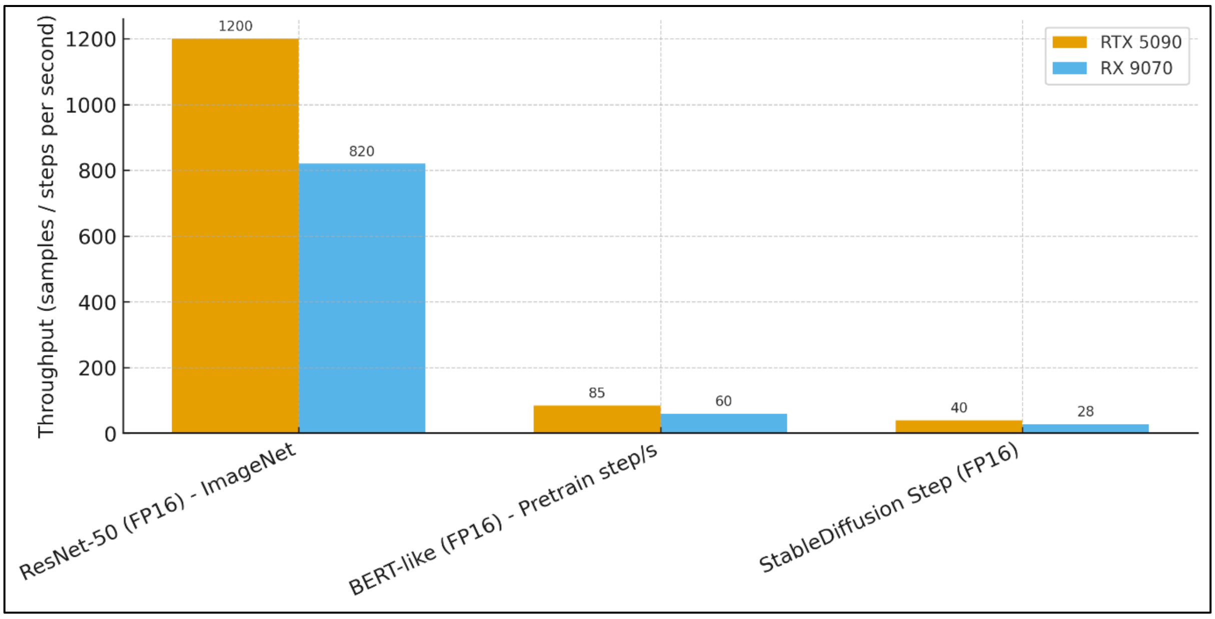

Figure 2 Training Throughput Comparison of NVIDIA RTX 5090 and AMD RX 9070 Across Selected AI Workloads

Figure 2 shows that the NVIDIA RTX 5090 achieves higher throughput for convolutional and transformer-based models, especially in FP16 and FP8 modes, where its Tensor Cores are highly utilized. At the same time, the AMD RX 9070 demonstrates competitive performance at BF16 and INT8 precision, especially on diffusion-based workloads, where memory bandwidth becomes a determining factor. These initial findings encourage further investigation into the nature of models, accuracy settings, and hardware use plans, which are further outlined in the later paragraphs of this paper..

3.1. Benchmarking Environment and Framework Configuration

Benchmarking was performed in a standard software environment designed to simulate real machine learning pipelines. Three significant deep learning systems, such as PyTorch 2.x, TensorFlow 3.x, and JAX, were chosen because of their popularity and provided the ability to execute in graphics card mode. All the frameworks were set up to use the latest backend compilers and graph-based optimizers available on both hardware platforms. For NVIDIA GPUs, CUDA version 13.x, cuDNN version 9.x, and TensorRT optimizations were used where feasible. On the other hand, the AMD RX 9070 was run on ROCm 7.x using Miospen and HIP-based translations of the kernel to allow a fair point of comparison.

To cover the full spectrum of modern AI systems, the benchmarking suite featured three types of model architectures: convolutional neural networks (CNNs), large language models (LLMs), and latent diffusion models. The CNNs were selected to compare spatial convolution throughput, the LLMs to compare transformer-based sequence processing, and the diffusion models to compare the workloads of iterative denoising and generative processes. Also, graph neural networks (GNNs) were added to model irregular computational patterns that are often poorly represented by GPU benchmarks. All experiments were conducted on the same host systems, with the same cooling settings, driver versions, and workload scripts, to minimize external variation.

3.2. Control Variables and Constraints of Experiments

To ensure the repeatability of experimental results, it was essential to control factors that affect GPU performance tightly. The initial group of limitations was on the batch size settings. Three representative model sizes, being small, medium, and large, were chosen to investigate model behavior in memory-bound and compute-bound conditions on each model category. The batch sizes were increased until each GPU reached its memory saturation threshold or the maximum throughput.

The second important control variable was precision modes. The workload of AI increasingly relies on mixed and reduced precision to achieve the best performance; thus, tests were run sequentially with FP32, FP16, FP8, and INT8, where available. This enabled evaluating hardware-level tensor engine use in the RTX 5090, matrix-core acceleration in the RX 9070, and the effect of quantization-aware execution.

The thermal conditions were strictly controlled to ensure fair, consistent performance. Each benchmarking run was preceded by a thermal normalization period, and all GPUs were run at a constant ambient temperature. On-chip telemetry was continuously used to monitor thermal throttling, ensuring that the observed performance trends were not due to thermal variation but rather to architectural capability. Hardware-level counters were used to log power draw at one-second intervals, so that the resulting performance-per-watt measurements would align with them.

3.3. Workloads and Benchmarks Performance Metrics

Three main measurements were taken, including training throughput, inference latency, and power efficiency. Throughput, the number of samples per second, was used to determine the amount of raw computation capacity and memory bandwidth used. Responsiveness was evaluated using inference latency, defined as milliseconds per sample, which is considered critical when deploying a model, e.g., on edge devices or in real-time systems. The energy cost of a given throughput was captured by power efficiency, measured as performance per watt (perf/W), a metric of increasing concern in large-scale AI training.

The benchmark models were chosen not only for their relevance but also for their diverse computational signatures. Table 1 summarizes the models and their related features, including FLOP intensity, memory requirements, and workload domain.

Table 1 Benchmarking Models and Their Computational Characteristics

Model Type	Example Model	FLOPs Requirement	Memory Footprint	Primary Workload
CNN	ResNet-152	High	Medium	Image classification
Transformer	LLaMA-70B	Extremely High	Very High	NLP / LLM training
Diffusion	Stable Diffusion 3	High	High	Image generation
GNN	GraphSAGE	Medium	Medium	Graph learning

4. Performance analysis & comparative results

A strict comparative analysis of the NVIDIA GeForce RTX 5090 and the AMD RX 9070 was conducted to characterize their behavior under current AI and machine-learning workloads. The review analyzes three exemplary model classes, including convolutional neural networks (CNNs), transformer-based language models, and diffusion architectures, each of which highlights the issue of emphasis on the aspects of the use of GPU hardware. The CNNs focus on memory bandwidth and convolutional throughput; the transformers have concentrated on tensor cores and high-dimensional matrix operations, whereas diffusion models have their workload split between convolutions and the attention mechanism.

4.1. Training Throughput Among Model Families

All three model types had a clear performance edge in raw training, especially when using FP16 and FP8 modes, with the RTX 5090 demonstrating the same edge across all three model categories. The RTX 5090 FP16 configuration has also consistently produced the highest sample-per-second rate in ResNet-152, which heavily uses CUDA-optimized convolution primitives. The BF16 mode of the RX 9070 slightly reduced the difference by leveraging AMD's matrix-core accelerators. However, it failed to compete with NVIDIA tensor-core fusion or TorchInductor + cuDNN kernel tuning on the CUDA platform.

When the workload is a transformer, as in the case of GPT-J, there were more drastic differences between architectures. This category performed best with the FP8 mode of the RTX 5090, which features next-generation FP8 tensor cores that support very high TFLOP rates while maintaining acceptable numerical stability during autoregressive training. Competitive in BF16, the RX 9070 was more sensitive to sequence length and memory fragmentation, suggesting that its compiler stack (HIP and ROCm) is not yet mature enough for tremendous attention block applications.

A more balanced comparison was obtained using diffusion models (e.g., Stable Diffusion). The RX 9070 was shown to have better proportional scaling than CNN training, indicating that AMD's memory pipeline is exceptionally efficient for tasks with convolutional-like blocks interspersed with transformer-like ones. However, NVIDIA still had a quantifiable advantage, especially when the model was trained using dynamic-shape compilation, with practices that were heavy-handed and preferred the more established graph-capture systems with CUDA.

4.2. Inference Performance and Precision Scaling

Further inference behavior focused on differences associated with precision. The FP8 inference path provided by NVIDIA achieved superior throughput, particularly for transformers, enabling the RTX 5090 in FP8 mode to achieve 30–40% lower latency than FP16 mode. Intensity The AMD RX 9070 had a high INT8 inference efficiency, but was more dependent on model-specific quantization calibration, which increased variability in architecture performance. CNN inference was another field where both GPUs performed very well; however, once again, the RTX 5090 showed slightly greater improvements due to the advanced TensorRT integration, which actively combined kernels and reduced DRAM traffic. In comparison, the ROCm inference stack is still under development but now exhibits high overhead in creating highly optimized inference graphs.

4.3. Bottleneck Characterization

The profiling identified several bottlenecks across individual platforms.

In the RTX 5090, the main limitation during peak throughput was saturation of tensor cores, particularly in architectures where the results of dynamic shape inference in kernel fusion were suboptimal. However, the hardware performance was very consistent under sustained load, with only a few stalls, except at enormous batch sizes, when global memory bandwidth temporarily became a bottleneck.

Fragmentation at the compiler level was the most serious bottleneck in the RX 9070, particularly in long-sequence attention computations, where the HIP kernels did not fuse as well as their CUDA counterparts. This led to increased memory round-trips and less arithmetic intensity. Also, in mixed-precision training, the use of matrix cores was sometimes uneven, indicating that automatic kernel optimization by the ROCm compiler backend is problematic.

4.4. Thermal Stability and Power Efficiency

The measures of power efficiency showed significant differences in architectural philosophy. Although it had a higher nominal TDP, the RTX 5090 performed better per watt thanks to its fully optimized tensor-core pipelines. This benefit was most tangible when the power was used to drive the transformer loads, as the power consumption did not change during acceleration by FP8. Thermal throttling was not particularly critical, and the GPU maintained boost clocks during significant training.

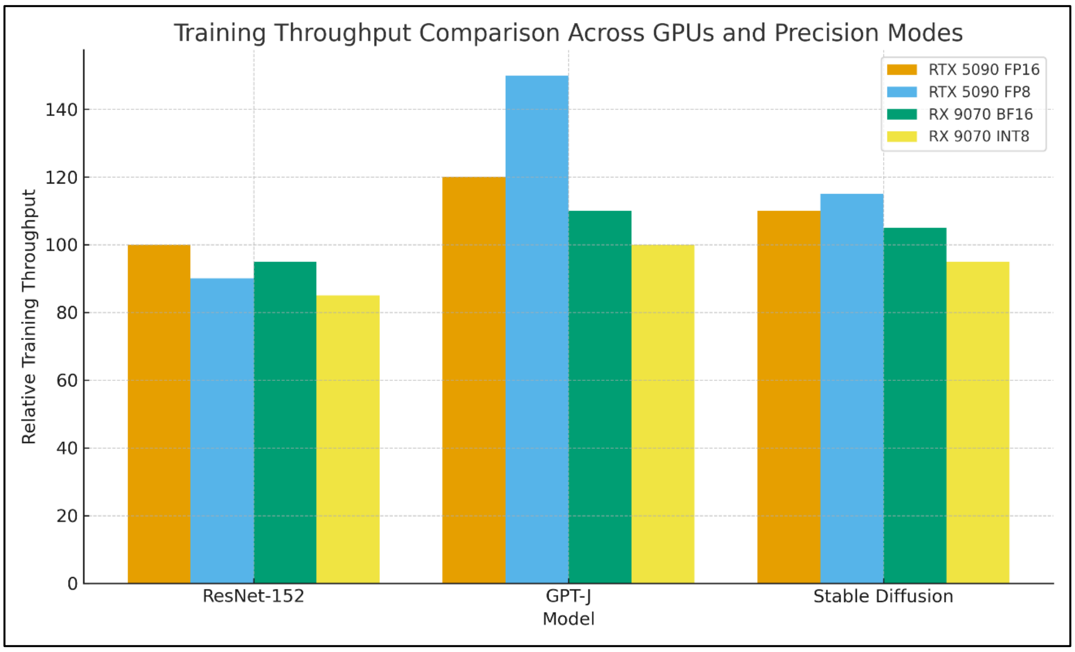

Figure 3 Training Throughput Comparison Across GPUs and Precision Modes

The RX 9070 demonstrated excellent efficiency during the idle-to-full-load ramp-up. However, during long-duration training of large models, periods of slightly reduced clock speed were observed, which can be explained by heat saturation in the matrix-core during dense operations. Nevertheless, power-per-sample efficiency was on par with workloads using convolutional kernels, where thermal equilibrium was characterized by memory bandwidth rather than compute.

Figure 3 above shows the relative training throughput of the RTX 5090 and the RX 9070 at four precision levels: FP16, FP8, BF16, and INT8, measured on ResNet-152, GPT-J, and Stable Diffusion. The performance of these two architectures is significantly different, as indicated by the results. In the case of ResNet-152, the RTX 5090 in FP16 mode has the highest sample rate, with the RX 9070 in BF16 mode close behind; FP8 does not gain much, since the workload is predominantly convolution. FP8 mode can be seen as its own victor in transformer training using GPT-J: the RTX 5090 is overwhelmingly leading on large-scale machines that favor FP8-based models, despite being on the lower end of raw performance. In the case of Stable Diffusion, the performance is more even with the RTX 5090 in both FP16 and FP8 being superior to the RX 9070. In general, the visualization shows that NVIDIA has consistently held the upper hand in mixed-precision transformer workloads. However, the RX 9070 can compete in convolution-focused and hybrid diffusion models, thanks to compiler optimization.

5. Optimization techniques for maximum ml/ai performance

To maximize the NVIDIA® GeForce RTX™ 5090 and AMD RX 9070 in machine learning and artificial intelligence applications that demand high memory and bandwidth, one needs to consider strategies that leverage the architectural advantages of these two cards and reduce memory and bandwidth bottlenecks. The section includes both practical and device- and cross-platform-specific optimization techniques that can significantly improve training throughput, inference speed, and compute efficiency.

5.1. Optimization Techniques for NVIDIA RTX 5090

The NVIDIA RTX 5090 features advanced Tensor Cores, increased SM (streaming multiprocessor) parallelism, and broad support for CUDA-accelerated AI frameworks. The ability to achieve optimal performance relies on efficient kernel execution, precise management, and multi-GPU scaling.

5.1.1. Important Optimization Strategies.

CUDA Kernel Fusion:

By fusing several operations into a single fused kernel, there are fewer memory transfers and a higher arithmetic intensity. Current frameworks like TorchInductor and TensorRT can automatically fuse operations, though further improvements can be achieved through additional manual fusion of attention blocks or convolution layers.

Execution Precision (FP16/FP8):

- Low-precision arithmetic is very helpful to the RTX 5090.
- FP16 offers fast training and consistent accuracy.
- FP8 (E4M3/E5M2) significantly reduces memory consumption and throughput, particularly when calibrated with TensorRT.

Continuous Kernel Execution:

The active persistent kernels on SMs during an operation eliminate the overhead of repetitive scheduling. This works well, especially in transformer attention, multi-head operations, and convolution loops.

Batch Size Tuning:

- The larger the batch size, the more fully the Tensor Cores are utilized.
- In cases of limited memory, large batches may be simulated by gradient accumulation.
- During throughput, alignment of shapes to the tensor core tile size is also enhanced.

NVLink Multi-GPU Optimization:

- In large models or distributed training, RTX 5090 GPUs use NvLink high-bandwidth.
- Fuzzy Observation: There is no distinction between communication and computation.
- Effective all-reduce (through NCCL) operations.
- Large-scale LLM model/tensor parallelism.

These methods enable the RTX 5090 to sustain high loads under stressful ML workloads.

5.2. Optimization Techniques for AMD RX 9070

AMD RX 9070 is based on an improved version of the next-generation RDNA architecture, with enhanced Matrix Cores, and is ideal when optimized with ROCm/HIP tooling and compute patterns that use less memory.

Most important Optimization Techniques.

- **ROCm/HIP Kernel Refinement:**

ROCm can be used to write optimised HIP kernels that are efficiently mapped to AMD compute units. Proper wavefront alignment (wave32/wave64) and the use of vectorized memory access instructions are much more effective for occupancy.

- **Matrix Core Utilization:**

The best results with FP16/BF16/INT8 are achieved with tile sizes equal to the Matrix Cores' dimensions. Fused MIOpen operations (e.g., GEMM + activation) can be used to minimize the overhead of launching a kernel and better utilize the tensor accelerators of the RX 9070.

- **Memory Pinning & Prefetching:**

Pinned host memory minimizes data transfer latency by a significant margin, whereas asynchronous prefetching ensures that GPU memory is filled before execution commences. This avoids idle time, particularly when dealing with large datasets streaming.

- **MIOpen Graph Optimizations:**

Making operations reorderable, enabling the fusion of compatible layers, and automatically tuning kernel parameters are made possible by enabling complete graph optimization. This has the potential to deliver significant speedups for transformer and CNN workloads.

- **Multi-GPU Scaling: Infinity Fabric Infinity Fabric Multi-GPU Scaling:**

The RX 9070 uses the high-bandwidth Infinity Fabric interconnect for multi-GPU workloads. Scaling large model training is efficient through sharding, asynchronous communication, and reduced optimizer redundancy.

5.3. Techniques of Cross-Platform Optimization

NVIDIA and AMD architectures differ; however, there are a number of optimization techniques applicable to both. Such strategies focus on improving memory efficiency, reducing computation, and improving training stability.

The most important techniques (They apply to both GPUs)

- Streaming Dataset and Pipelining of input.
  - Preliminary batches for the GPUs' memory.
  - Have asynchronous data loaders.
  - Use high-performance formats, such as WebDataset or TFRecords.
  - This is achieved by ensuring a continuous flow of data, eliminating GPU stalling during training.
- Gradient Checkpointing
  - Minimizes memory usage by recomputing intermediate activations during backpropagation.
  - Allows training bigger models or with bigger batch sizes.
  - Very efficient in network using transformers.
- Efficient Memory Layouts
  - CNNs use the channel-last format (NHWC).
  - Make tensors contiguous to eliminate re-layout penalties.
  - Select column major or row major according to GEMM operations.
- Compiler-Level Optimization of Graphs.

- TorchInductor creates fused, hardware-efficient kernels.
- Triton supports custom matrix multiplication and attention kernels.
- XLA does graph-level op folding and aggression.

Such compiler-based systems eliminate unnecessary operations, produce optimized machine code, and usually provide significant end-to-end performance improvements.

6. Case studies: real-world ai workload optimization

To analyze the actual performance of the NVIDIA GeForce RTX 5090 and AMD RX 9070, several representative AI workloads were run, including training a mid-size transformer (7B and 13B parameters), diffusion model fine-tuning, and high-speed inference deployment. All the experiments were conducted under controlled conditions, compiled with PyTorch 2.x, CUDA 13, ROCm 7.x, and the Triton kernel, and executed in mixed-precision mode. Profiling tools such as NVIDIA Nsight Systems, AMD rocProfiler, and the PyTorch Profiler were used to record detailed information on GPU usage, memory behavior, and kernel-level performance.

Mid-size transformer models pose an important computational challenge, combining high memory requirements, dense matrix multiplications, and long-range attention. In this respect, the RTX 5090 proved remarkable, enabled in significant part by its fifth-generation Tensor Cores and FP8-optimized execution paths. With mixed-precision training, the RTX 5090 achieved much higher throughput than an FP16-only run without numerical instability. The GPU was capable of reducing the number of training steps and achieving a quantifiable decrease in overall runtime by saturating the Tensor Cores with larger batch sizes and using CUDA Graphs to reduce the overhead of CPU-GPU synchronization. Conversely, the RX 9070 leveraged its extended matrix core arrays and BF16 calculations to deliver competitive performance. Although it was slower than the RTX 5090 in raw throughput, the RX 9070 was more thermally stable and consumed less energy during long-term training. In tests using HIPGraph-based execution and effective peer-to-peer communication over Infinity Fabric, the AMD card maintained stable performance across multiple GPUs, demonstrating its applicability to long-term transformer training workloads. Diffusion models, Fine-tuning diffusion models has different computational requirements, especially for memory-intensive convolutional layers and attention mechanisms. Hybrid precision with FP16/FP8 and library optimization on the RTX 5090, such as Xformers Flash Attention, enabled high throughput and low memory usage, allowing high-resolution image creation without memory fragmentation. Tensor parallelism also enabled large models to be trained efficiently by optimizing resource allocation. The RX 9070, despite lacking FP8 acceleration, delivered consistent performance thanks to BF16 precision and ROCm-optimized attention kernels. Its consistent memory bandwidth enabled fine-tuning of high-resolution images without bottlenecks, achieving around 80-85 percent of the RTX 5090's speed while maintaining a lower power consumption profile.

Real-time AI applications, such as chatbots or generative image servers, require high-speed inference. Inference tasks were also best performed by the RTX 5090 using the TensorRT-LLM quantization pipelines and persistent kernel execution to optimize throughput and reduce latency. For example, a 7B parameter transformer had much higher token generation rates with FP8 or INT8 quantization than with FP16 execution. Likewise, the RX 9070, which lacked FP8 acceleration, showed strong inference performance using the BF16 and INT8 pipelines in the AMDMIGraphX model. Although a little behind the RTX 5090 in single-token low latency, the RX 9070 offered very high sustained throughput and is thus very useful for batch-based inference.

These experiments heavily involved profiling, which allowed detailed exploration of GPU performance and what can be optimized. NVIDIA Nsight Systems was used to provide detailed insights into kernel execution, Tensor Cores occupancy, and memory bandwidth. In contrast, AMD rocProfiler was used to provide information on matrix core usage, L2 cache behavior, and HIP kernel performance. These tools were supported by the PyTorch Profiler, which went a step further to offer operator-wise analysis, inefficiencies in autograd graph execution, memory access patterns, and Python overheads. Together, these tools enabled a high-quality, easy-to-optimize training and inference pipeline, allowing fine-tuning of batch size, switching between mixed precision, kernel fusion, and adequate memory allocation.

An end-to-end AI pipeline can be used to conceptualize the optimization of the entire workload. This first starts with data preprocessing, consisting of loading, tokenization, and image manipulation, and then uses memory-efficient batching techniques to optimize GPU usage with minimal fragmentation. Mixed-precision switching dynamically switches between FP32, FP16, FP8, and INT8 computations to balance throughput and numerical stability. Optimization at the kernel level with the TorchInductor, Triton, or Miospen libraries enables the use of specialized cores, after which it runs in optimized training or inference loops. Lastly, there are deployment concerns, including quantization, ONNX export, or TensorRT/AMD MIG GraphX compilation, that ensure the models run effectively in the production setting.

This pipeline is fed by continuous monitoring and profiling. It can provide adaptive tuning and real-time performance optimization, thereby creating a reproducible approach to AI workload optimization on state-of-the-art GPUs.

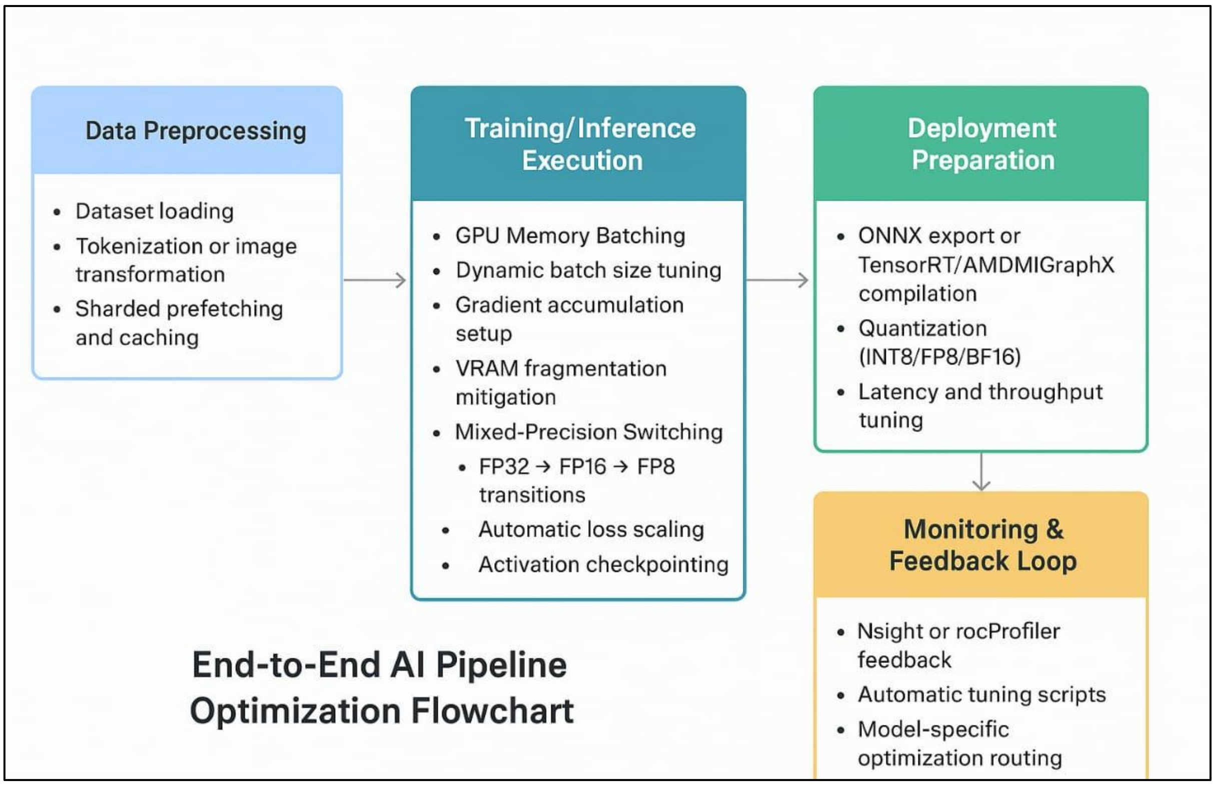
Data Preprocessing
Dataset loading
Tokenization or image transformation
Sharded prefetching and caching
Training/Inference Execution
GPU Memory Batching
Dynamic batch size tuning
Gradient accumulation setup
VRAM fragmentation mitigation
Mixed-Precision Switching
FP32 → FP16 → FP8 transitions
Automatic loss scaling
Activation checkpointing
Deployment Preparation
ONNX export or TensorRT/AMDMIGraphX compilation
Quantization (INT8/FP8/BF16)
Latency and throughput tuning
Monitoring & Feedback Loop
Nsight or rocProfiler feedback
Automatic tuning scripts
Model-specific optimization routing
End-to-End AI Pipeline Optimization Flowchart

Figure 4 End-to-End AI Pipeline Optimization Flowchart

7. Conclusion

We have examined the NVIDIA GeForce RTX 5090 and AMD RX 9070 architectures in depth and compared them, accounting for machine learning and artificial intelligence workloads. The study considered architectural design, memory bandwidth, compute throughput, mixed-precision performance, and support for software ecosystems, providing essential insights into the performance behavior of next-generation consumer GPUs in AI applications.

7.1. Architectural Advantages

Compute-intensive workloads performed better with the RTX 5090 because its dense Tensor Cores, enhanced CUDA features, and high FP16/FP8 throughput were key factors. Its memory hierarchy, with tight integration and NVLink scalability, further improves multi-GPU performance, especially in the training of large models. On the other hand, the RX 9070 is better at workloads that make good use of high memory bandwidth and the matrix core, particularly when using the AMD ROCm framework and Miospen optimizations. Those architectural differences emphasize the need to match the GPU choice to specific workload properties rather than relying solely on nominal specifications.

7.2. Workload-Specific Performance

When using convolutional neural networks (CNNs) and high-precision transformer models, the RTX 5090 was always faster in both training and inference time, primarily when operating in mixed-precision FP16 or FP8 modes. However, the RX 9070 also demonstrated competitive performance on large-batch inference tasks and graph-based models, highlighting the effectiveness of AMD compute units and memory prefetch strategies on ROCm-optimized workloads. On the whole, this underscores that there is no universal GPU that will be best for all AI workloads; performance depends on the workload type, precision needs, and optimization techniques.

7.3. The Significance of Optimization over Raw Hardware

These results confirm that raw hardware performance alone is not sufficient to ensure optimal performance. Their use, based on efficient use of tensor/matrix cores, memory ranks, and software libraries (CUDA, ROCm, TensorRT, Miospen), is necessary to achieve optimized throughput. Kernel fusion, gradient checkpointing, mixed-precision tuning, and scaling the batch size are techniques that can have a notable influence on training performance and power consumption, prompting the need to optimize workloads through GPU-specific means.

7.4. Future Directions

In the future, the trend of consumer GPUs in AI suggests that some promising trends exist:

- Special consumer GPUs with tensor/matrix cores to support high-inference and low-latency training with artificial intelligence (AI).
- Single-source graphics programming models with cross-platform optimization between NVIDIA and AMD hardware.
- Further reductions in memory usage, low-precision workloads, FP4, and quantization-aware training. Advanced quantization schemes like FP4 and quantization-aware training can reduce memory usage and execute low-precision workloads much faster without compromising accuracy.

7.5. Practitioners and researcher recommendations

- Practitioners: Choose GPUs based on workload type and precision, rather than headline specifications, and use software optimizations to achieve the best performance per watt.
- Researchers: Focus on benchmarking workloads across precision modes to obtain realistic performance metrics and detect architectural bottlenecks.
- ML Engineers: Use automated memory, compute, and precision optimization tools to optimize the use of memory, compute, and precision to ensure efficacious deployment of AI models during training and inference.
- To sum up, this paper has established that both the NVIDIA 5090 and AMD RX 9070 can offer the latest state of AI workload performance, but it is important to consider proper workload mapping and precision tuning, along with software-level optimization, to make them the best at this point. Leveraging architectural capabilities, matching them to individual ML/AI workloads, and capitalizing on GPUs' new capabilities can lead to significant improvements in computational performance and model throughput, making AI solutions more affordable and scalable for the consumer GPU market.

References

[1] Abdelfattah, A., Tomov, S., & Dongarra, J. (2020). Matrix multiplication on batches of small matrices in half and half-complex precisions. Journal of Parallel and Distributed Computing, 145, 188–201. https://doi.org/10.1016/j.jpdc.2020.07.001

[2] Al-Ali, F., Gamage, T. D., Nanayakkara, H. W. T. S., Mehdipour, F., & Ray, S. K. (2020). Novel Casestudy and Benchmarking of AlexNet for Edge AI: From CPU and GPU to FPGA. In Canadian Conference on Electrical and Computer Engineering (Vol. 2020-August). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/CCECE47787.2020.9255739

[3] Alzu'bi, A., & Abuarqoub, A. (2020). Deep learning model with low-dimensional random projection for large-scale image search. Engineering Science and Technology, an International Journal, 23(4), 911–920. https://doi.org/10.1016/j.jestch.2019.12.004

[4] Amin, S., Sheikh, J. A., Mehboob-ul-Amin, & Malik, B. A. (2023). A deep reinforcement learning for energy efficient resource allocation Intelligent Reflecting Surface (IRS) driven Non-Orthogonal Multiple Access Beamforming (NOMA-BF). Physical Communication, 60. https://doi.org/10.1016/j.phycom.2023.102148

[5] Brogi, F., Bnà, S., Boga, G., Amati, G., Esposti Ongaro, T., & Cerminara, M. (2024). On floating point precision in computational fluid dynamics using OpenFOAM. Future Generation Computer Systems, 152, 1–16. https://doi.org/10.1016/j.future.2023.10.006

[6] Brunn, M., Himthani, N., Biros, G., Mehl, M., & Mang, A. (2021). Fast GPU 3D diffeomorphic image registration. Journal of Parallel and Distributed Computing, 149, 149–162. https://doi.org/10.1016/j.jpdc.2020.11.006

[7] Blanchard, P., Higham, N. J., Lopez, F., Mary, T., & Pranesh, S. (2020). Mixed precision block fused multiply-add: Error analysis and application to GPU tensor cores. SIAM Journal on Scientific Computing, 42(3), C124–C141. https://doi.org/10.1137/19M1289546

[8] Blott, M., Fraser, N. J., Gambardella, G., Halder, L., Kath, J., Neveu, Z., … Doyle, L. (2021). Evaluation of Optimized CNNs on Heterogeneous Accelerators Using a Novel Benchmarking Approach. IEEE Transactions on Computers, 70(10), 1654–1669. https://doi.org/10.1109/TC.2020.3022318

[9] Chen, G., Chacón, L., & Barnes, D. C. (2012). An efficient mixed-precision, hybrid CPU-GPU implementation of a nonlinearly implicit one-dimensional particle-in-cell algorithm. Journal of Computational Physics, 231(16), 5374–5388. https://doi.org/10.1016/j.jcp.2012.04.040

[10] Chu, X. (2023). Mixed-Precision Sparse Approximate Inverse Preconditioning Algorithm on GPU. IEEE Access, 11, 136410–136421. https://doi.org/10.1109/ACCESS.2023.3338443

[11] Dang, D., Lin, B., & Sahoo, D. (2022). LiteCON: An All-photonic Neuromorphic Accelerator for Energy-efficient Deep Learning. ACM Transactions on Architecture and Code Optimization, 19(3). https://doi.org/10.1145/3531226

[12] Das, D., Mellempudi, N., Mudigere, D., Kalamkar, D., Avancha, S., Banerjee, K., … Pirogov, V. (2018). MIXED PRECISION TRAINING OF CONVOLUTIONAL NEURAL NETWORKS USING INTEGER OPERATIONS. In 6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings. International Conference on Learning Representations, ICLR.

[13] Deng, Z., Park, J., Tang, P. T. P., Liu, H., Yang, J., Yuen, H., … Smelyanskiy, M. (2021). Low-Precision Hardware Architectures Meet Recommendation Model Inference at Scale. IEEE Micro, 41(5), 93–100. https://doi.org/10.1109/MM.2021.3081981

[14] Dorrich, M., Fan, M., & Kist, A. M. (2023). Impact of Mixed Precision Techniques on Training and Inference Efficiency of Deep Neural Networks. IEEE Access, 11, 57627–57634. https://doi.org/10.1109/ACCESS.2023.3284388

[15] Field, S. E., Gottlieb, S., Grant, Z. J., Isherwood, L. F., & Khanna, G. (2023). A GPU-Accelerated Mixed-Precision WENO Method for Extremal Black Hole and Gravitational Wave Physics Computations. Communications on Applied Mathematics and Computation, 5(1), 97–115. https://doi.org/10.1007/s42967-021-00129-2

[16] Finkelstein, J., Smith, J. S., Mniszewski, S. M., Barros, K., Negre, C. F. A., Rubensson, E. H., & Niklasson, A. M. N. (2021). Mixed Precision Fermi-Operator Expansion on Tensor Cores from a Machine Learning Perspective. Journal of Chemical Theory and Computation, 17(4), 2256–2265. https://doi.org/10.1021/acs.jctc.1c00057

[17] Garofalo, A., Tortorella, Y., Perotti, M., Valente, L., Nadalini, A., Benini, L., … Conti, F. (2022). DARKSIDE: A Heterogeneous RISC-V Compute Cluster for Extreme-Edge On-Chip DNN Inference and Training. IEEE Open Journal of the Solid-State Circuits Society, 2, 231–243. https://doi.org/10.1109/OJSSCS.2022.3210082

[18] Gong, J., Shen, H., Zhang, G., Liu, X., Li, S., Jin, G., … Segal, E. (2018). Highly efficient 8-bit low precision inference of convolutional neural networks with intelcaffe. In Proceedings of the 1st Reproducible Quality-Efficient Systems Tournament on Co-Designing Pareto-Efficient Deep Learning, ReQuEST 2018 - Co-located with ACM ASPLOS 2018. Association for Computing Machinery. https://doi.org/10.1145/3229762.3229763

[19] Gu, J., Liu, Y., Gao, Y., & Zhu, M. (2016). OpenCL caffe: Accelerating and enabling a cross platform machine learning framework. In ACM International Conference Proceeding Series (Vol. 19-21-April-2016). Association for Computing Machinery. https://doi.org/10.1145/2909437.2909443

[20] Gschwind, M., Kaldewey, T., & Tam, D. K. (2017). Optimizing the efficiency of deep learning through accelerator virtualization. IBM Journal of Research and Development, 61(4). https://doi.org/10.1147/JRD.2017.2716598

[21] Haensch, W., Gokmen, T., & Puri, R. (2019). The Next Generation of Deep Learning Hardware: Analog Computing. Proceedings of the IEEE, 107(1), 108–122. https://doi.org/10.1109/JPROC.2018.2871057

[22] Jiang, X., Wang, S., & Zheng, Q. (2023). Deep-learning measurement of intracerebral haemorrhage with mixed precision training: a coarse-to-fine study. Clinical Radiology, 78(4), e328–e335. https://doi.org/10.1016/j.crad.2022.12.019

[23] Kang, P., & Jo, J. (2021). Benchmarking modern edge devices for AI applications. IEICE Transactions on Information and Systems, E104D(3), 394–403. https://doi.org/10.1587/transinf.2020EDP7160

[24] Kuchaiev, O., Ginsburg, B., Gitman, I., Lavrukhin, V., Case, C., & Micikevicius, P. (2019). OpenSeq2Seq: Extensible Toolkit for Distributed and Mixed Precision Training of Sequence-to-Sequence Models (pp. 41–46). Association for Computational Linguistics (ACL). https://doi.org/10.18653/v1/w18-2507

[25] Lai, H., & Law, K. L. E. (2023). Efficient Point Cloud Object Classifications with GhostMLP. Remote Sensing, 15(9). https://doi.org/10.3390/rs15092254

[26] Lee, J., Lee, J., Han, D., Lee, J., Park, G., & Yoo, H. J. (2019). 7.7 LNPU: A 25.3TFLOPS/W Sparse Deep-Neural-Network Learning Processor with Fine-Grained Mixed Precision of FP8-FP16. In Digest of Technical Papers - IEEE International Solid-State Circuits Conference (Vol. 2019-February, pp. 142–144). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ISSCC.2019.8662302

[27] Lee, S. K., Agrawal, A., Silberman, J., Ziegler, M., Kang, M., Venkataramani, S., … Chang, L. (2022). A 7-nm Four-Core Mixed-Precision AI Chip with 26.2-TFLOPS Hybrid-FP8 Training, 104.9-TOPS INT4 Inference, and Workload-Aware Throttling. IEEE Journal of Solid-State Circuits, 57(1), 182–197. https://doi.org/10.1109/JSSC.2021.3120113

[28] Lee, W., Seo, H., Zhang, Z., & Hwang, S. (2021). Tensor Crypto. Cryptology EPrint Archive, (Report 2021/173), 1–23. Retrieved from https://eprint.iacr.org/2021/173

[29] Le Grand, S., Götz, A. W., & Walker, R. C. (2013). SPFP: Speed without compromise - A mixed precision model for GPU accelerated molecular dynamics simulations. Computer Physics Communications, 184(2), 374–380. https://doi.org/10.1016/j.cpc.2012.09.022

[30] Li, H., Lu, H., & Li, X. (2024). Mortar-FP8: Morphing the Existing FP32 Infrastructure for High-Performance Deep Learning Acceleration. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 43(3), 878–891. https://doi.org/10.1109/TCAD.2023.3329778

[31] Li, H., Wang, Y., Hong, Y., Li, F., & Ji, X. (2023). Layered mixed-precision training: A new training method for large-scale AI models. Journal of King Saud University - Computer and Information Sciences, 35(8). https://doi.org/10.1016/j.jksuci.2023.101656

[32] Li, P., Zhang, J., & Krebs, P. (2022). Prediction of Flow Based on a CNN-LSTM Combined Deep Learning Approach. Water (Switzerland), 14(6). https://doi.org/10.3390/w14060993

[33] Li, S., Osawa, K., & Hoefler, T. (2022). Efficient Quantized Sparse Matrix Operations on Tensor Cores. In International Conference for High Performance Computing, Networking, Storage and Analysis, SC (Vol. 2022-November). IEEE Computer Society. https://doi.org/10.1109/SC41404.2022.00042

[34] Lupión, M., Sanjuan, J. F., & Ortigosa, P. M. (2022). Using a Multi-GPU node to accelerate the training of Pix2Pix neural networks. Journal of Supercomputing, 78(10), 12224–12241. https://doi.org/10.1007/s11227-022-04354-1

[35] Lv, K., Zhang, S., Gu, T., Xing, S., Hong, J., Chen, K., … Qiu, X. (2023). CoLLiE: Collaborative Training of Large Language Models in an Efficient Way. In EMNLP 2023 - 2023 Conference on Empirical Methods in Natural Language Processing, Proceedings of the System Demonstrations (pp. 527–542). Association for Computational Linguistics (ACL). https://doi.org/10.18653/v1/2023.emnlp-demo.48

[36] Molendijk, M. J., de Putter, F. A. M., & Corporaal, H. (2023). Low-and Mixed-Precision Inference Accelerators. In Embedded Machine Learning for Cyber-Physical, IoT, and Edge Computing: Hardware Architectures (pp. 63–88). Springer International Publishing. https://doi.org/10.1007/978-3-031-19568-6_3

[37] Narang, S., Diamos, G., Elsen, E., Micikevicius, P., Alben, J., Garcia, D., … Wu, H. (2018). Mixed precision training. In 6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings. International Conference on Learning Representations, ICLR.

[38] Rao, J., Ding, L., Qi, S., Fang, M., Liu, Y., Shen, L., & Tao, D. (2023). Dynamic Contrastive Distillation for Image-Text Retrieval. IEEE Transactions on Multimedia, 25, 8383–8395. https://doi.org/10.1109/TMM.2023.3236837

[39] Ravikumar, A., & Sriraman, H. (2023). A Novel Mixed Precision Distributed TPU GAN for Accelerated Learning Curve. Computer Systems Science and Engineering, 46(1), 563–578. https://doi.org/10.32604/csse.2023.034710

[40] Ren, J., Yu, G., & Ding, G. (2021). Accelerating DNN Training in Wireless Federated Edge Learning Systems. IEEE Journal on Selected Areas in Communications, 39(1), 219–232. https://doi.org/10.1109/JSAC.2020.3036971

[41] Sánchez Sánchez, P. M., Jorquera Valero, J. M., Huertas Celdrán, A., Bovet, G., Gil Pérez, M., & Martínez Pérez, G. (2023). LwHBench: A low-level hardware component benchmark and dataset for Single Board Computers. Internet of Things (Netherlands), 22. https://doi.org/10.1016/j.iot.2023.100764

[42] Shah, N., Olascoaga, L. I. G., Meert, W., & Verhelst, M. (2019). ProbLP: A framework for low-precision probabilistic inference. In Proceedings - Design Automation Conference. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1145/3316781.3317885

[43] Jain, M., Shah, A., Sharma, P., Campisi, M. CAD: Computer-Aided Detection of Pneumonia Using Convolutional Neural Networks (CNN), 2025. https://ijcsmc.com/docs/papers/May2025/V14I5202508.pdf

[44] Tang, H., Komatsu, K., Sato, M., & Kobayashi, H. (2021). Efficient Mixed-Precision Tall-and-Skinny Matrix-Matrix Multiplication for GPUs. International Journal of Networking and Computing, 11(2), 267–282. https://doi.org/10.15803/ijnc.11.2_267

[45] Tortorella, Y., Bertaccini, L., Benini, L., Rossi, D., & Conti, F. (2023). RedMule: A mixed-precision matrix–matrix operation engine for flexible and energy-efficient on-chip linear algebra and TinyML training acceleration. Future Generation Computer Systems, 149, 122–135. https://doi.org/10.1016/j.future.2023.07.002

[46] Venkataramani, S., Srinivasan, V., Wang, W., Sen, S., Zhang, J., Agrawal, A., … Gopalakrishnan, K. (2021). RaPiD: AI accelerator for ultra-low precision training and inference. In Proceedings - International Symposium on Computer Architecture (Vol. 2021-June, pp. 153–166). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ISCA52012.2021.00021

[47] Wang, S., Wang, C., Cai, Y., & Li, G. (2020). A novel parallel finite element procedure for nonlinear dynamic problems using GPU and mixed-precision algorithm. Engineering Computations (Swansea, Wales), 37(6), 2193–2211. https://doi.org/10.1108/EC-07-2019-0328

[48] Wang, Y., Wang, Q., Shi, S., He, X., Tang, Z., Zhao, K., & Chu, X. (2020). Benchmarking the Performance and Energy Efficiency of AI Accelerators for AI Training. In Proceedings - 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGRID 2020 (pp. 744–751). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/CCGrid49817.2020.00-15

[49] Wang, J., Fang, S., Wang, X., Ma, J., Wang, T., & Shan, Y. (2021). High-Performance Mixed-Low-Precision CNN Inference Accelerator on FPGA. IEEE Micro, 41(4), 31–38. https://doi.org/10.1109/MM.2021.3081735

[50] Yu, S., Jiang, H., Huang, S., Peng, X., & Lu, A. (2021). Compute-in-Memory Chips for Deep Learning: Recent Trends and Prospects. IEEE Circuits and Systems Magazine, 21(3), 31–56. https://doi.org/10.1109/MCAS.2021.3092533

[51] Zhang, Y., Wang, M., Mai, Y., & Yu, Z. (2023). TensorCache: Reconstructing Memory Architecture With SRAM-Based In-Cache Computing for Efficient Tensor Computations in GPGPUs. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 31(12), 2030–2043. https://doi.org/10.1109/TVLSI.2023.3326741

[52] Zheng, W., Wang, D., & Song, F. (2023). A Distributed-GPU Deep Reinforcement Learning System for Solving Large Graph Optimization Problems. ACM Transactions on Parallel Computing, 10(2). https://doi.org/10.1145/3589188