World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(REVIEW ARTICLE)

Check for updates

Optimizing generative AI models for edge deployment: Techniques and best practices

Sai Kalyan Reddy Pentaparthi *

ST Engineering iDirect, Inc., USA.

World Journal of Advanced Research and Reviews, 2025, 26(01), 1485-1492

Publication history: Received on 28 February 2025; revised on 07 April 2025; accepted on 10 April 2025

Article DOI: https://doi.org/10.30574/wjarr.2025.26.1.1161

Abstract

Generative AI models represent a significant advancement in content creation capabilities but face substantial challenges when deployed at the network edge due to inherent resource constraints. This article examines comprehensive optimization strategies for enabling generative AI functionality on edge devices without requiring cloud connectivity. The exponential growth in model size has created a widening gap between computational requirements and the limited resources available in edge environments. Through systematic model compression, architectural redesign, and hardware-software co-optimization, generative models can achieve dramatic efficiency improvements while maintaining acceptable quality thresholds. The compression techniques examined include pruning methodologies that systematically eliminate redundant parameters, quantization approaches that reduce numerical precision, and knowledge distillation methods that transfer capabilities from larger models to compact alternatives. Architectural innovations such as modified attention mechanisms, conditional computation, and neural architecture search further enhance efficiency by fundamentally rethinking model design for resource-constrained environments. The integration of these techniques with hardware-specific optimizations and specialized software frameworks enables practical deployment across diverse application domains. Real-world implementations in speech processing, computer vision, and industrial IoT demonstrate that properly optimized generative models can operate within edge constraints while delivering near-real-time performance and maintaining high-quality outputs. These advancements empower industries to leverage generative AI capabilities in scenarios where privacy concerns, connectivity limitations, or latency requirements make cloud-based processing impractical.

Keywords: Generative AI; Edge Computing; Model Compression; Quantization; Neural Architecture Search; Hardware Acceleration

1. Introduction

Generative artificial intelligence (GenAI) has transformed digital content creation while simultaneously presenting significant deployment challenges at the network edge. The computational demands of state-of-the-art generative models are substantial, with leading transformer-based architectures containing between 175 billion and 540 billion parameters, requiring 350-700GB of memory and 5,000-8,000 watt-hours of energy for a single training run [1]. Edge deployment of these models confronts fundamental resource constraints, as typical edge devices offer only 4-16GB of RAM, 10-50 TOPS of computing capability, and operate within 1-5W power envelopes [2].

The market for edge AI accelerators is responding to these challenges, growing at 38.9% CAGR with projected value reaching $7.68 billion by 2027, reflecting the increasing demand for efficient on-device generative capabilities [1]. Survey data indicates 78.3% of organizations now prioritize edge AI deployment to address latency requirements, with 64.7% citing data privacy regulations as a primary motivation for edge processing solutions [2]. These market forces are driving innovation in model optimization, with 82% of commercial edge AI deployments now utilizing some form of model compression or architectural modification [1].

* Corresponding author: Sai Kalyan Reddy Pentaparthi

Table 1 Edge AI Market and Model Constraints [1, 2]

Year	Edge AI Market (Billion $)	GenAI Memory Requirements (GB)	Edge Device Memory (GB)	Model-Device Memory Gap (Factor)
2023	2.56	350	4	87.5
2024	3.56	420	6	70
2025	4.94	490	8	61.3
2026	6.87	580	12	48.3
2027	7.68	650	16	40.6

Performance benchmarks demonstrate the necessity of these optimizations, as unmodified generative models often exceed practical edge constraints by orders of magnitude. Text generation models like GPT variants require 350-520ms of inference time per token on standard edge CPUs, while image generation using diffusion models demands 75-120 seconds per image at reasonable quality levels [2]. Through strategic optimization, these latencies can be reduced by 85-97%, bringing generation tasks within feasible operational parameters for edge deployment [1].

The resource-performance tradeoff represents a central challenge in edge GenAI deployment. Cloud-based generative models typically operate with 99.7% accuracy and high-fidelity output quality, whereas edge-optimized variants must balance reduced model capacity against acceptable output degradation [2]. Research indicates that properly optimized models can maintain 92-96% of reference quality while reducing computational requirements by 75-90%, enabling practical edge deployment that meets both resource constraints and user expectations [1]. This article examines proven optimization strategies for deploying generative AI at the edge, focusing on techniques that achieve maximum efficiency while preserving core generation capabilities across diverse application domains and hardware environments.

2. Model Compression Technique

Model compression techniques represent essential strategies for deploying generative AI on edge devices with constrained resources. Recent benchmarks demonstrate that uncompressed transformer-based generative models require 4-16GB of memory and 10-30 GFLOPS of compute power, exceeding typical edge device capabilities by factors of 5-20× [3]. Through strategic compression, these requirements can be substantially reduced while maintaining acceptable generation quality.

2.1. Pruning

Network pruning systematically eliminates redundant parameters based on importance criteria. Magnitude-based pruning, which removes weights below specified thresholds, has demonstrated 67.8% model size reduction with only 2.3% degradation in generation quality across benchmarked generative models [3]. Hardware-aware structured pruning, which removes entire filters or channels, achieves 3.2× speedup on edge NPUs while maintaining 94.6% of baseline performance in image generation tasks [4].

The lottery ticket hypothesis approach has yielded particularly promising results for generative models. Experiments with transformer-based text generators demonstrate that pruned subnetworks containing only 14.7% of original weights maintain 95.3% of full model performance on standard benchmarks, reducing inference time from 752ms to 167ms on edge GPUs [3]. Dynamic pruning further enhances efficiency, with adaptive techniques demonstrating 40.3% lower average computation compared to static approaches for variable-complexity inputs [4].

2.2. Quantization

Precision reduction through quantization substantially decreases memory and computation requirements. Post-training quantization to 8-bit integers reduces model size by 73.5% with only 1.9% quality degradation for generative tasks [3]. More aggressive 4-bit quantization yields 87.2% size reduction but increases quality loss to 4.6%, establishing practical trade-off boundaries for deployment scenarios [4].

Table 2 Model Compression Effectiveness [3, 4]

Compression Technique	Model Size Reduction (%)	Performance Retention (%)	Inference Speedup (×)	Memory Savings (%)
Magnitude Pruning	67.8	97.7	2.1	67.8
Structured Pruning	74.5	94.6	3.2	74.5
Lottery Ticket Pruning	85.3	95.3	4.5	85.3
8-bit Quantization	73.5	98.1	3.4	75
4-bit Quantization	87.2	95.4	5.8	87.2
Knowledge Distillation	90.1	91.2	8.4	90.1
Progressive Distillation	97.2	93.7	11.3	97.2

Quantization-aware training demonstrates superior results, with INT8 models achieving less than 1.3% quality degradation compared to FP32 baselines [3]. Mixed-precision approaches optimize this trade-off further, with detailed benchmarks showing that allocating 16-bit precision to 12.3% of particularly sensitive network components while maintaining 8-bit precision elsewhere results in only 0.7% quality degradation while preserving 69.8% of compression benefits [4].

3. Knowledge Distillation

Knowledge distillation transfers capabilities from larger teacher models to compact student networks. Response-based distillation for generative image models yields 8.4× parameter reduction while maintaining 91.2% of generation quality on established metrics [3]. Feature-based approaches demonstrate even better results for text generation, with 11.3× smaller student models achieving 93.7% of teacher model performance on coherence and relevance metrics [4].

Progressive distillation through multiple intermediate models has proven particularly effective, with benchmarks showing 4.7% higher generation quality compared to direct distillation when reducing GPT-style models from 1.3 billion to 37 million parameters [3]. For real-time edge deployment, these optimized models achieve 28-143ms inference latency per generation step across tested edge devices, making interactive generative applications viable on current hardware [4].

3.1. Architectural Optimizations for Edge Deployment

Architectural redesign represents a critical strategy for enabling generative AI deployment on edge devices. Standard generative architectures require significant adaptation to meet the strict constraints of edge computing environments, where memory limitations and computational efficiency are paramount concerns.

3.2. Efficient Model Architectures

Modified attention mechanisms provide substantial efficiency improvements in transformer-based generative models. Linear attention variants reduce the computational complexity from $O(n^2)$ to $O(n)$, decreasing memory requirements by 76.3% for sequence processing on resource-constrained devices [5]. Benchmarks across five different edge platforms demonstrate that these optimized attention mechanisms reduce inference latency by 68.4-81.7% while processing 3.2× longer sequences within the same memory footprint [5].

Depth-wise separable convolutions in generative image models reduce FLOPs by 87.6% compared to standard convolutions, with only 2.7% degradation in generation quality metrics [6]. This architectural modification reduces energy consumption by 79.3% on tested edge NPUs, extending battery life from 4.2 to 18.7 hours during continuous operation [5].

Table 3 Architectural Optimization Impact [5, 6]

Architectural Technique	Compute Reduction (%)	Memory Reduction (%)	Latency Improvement (%)	Energy Savings (%)
Linear Attention	75	76.3	75.1	72.4
Depth-wise Separable Conv	87.6	72.8	76.5	79.3
Progressive Generation	82.4	67.5	74.3	78.5
Mixture of Experts	72.8	64.7	76.8	68.3
Early-exit Mechanisms	57.7	48.5	52.9	54.6
Adaptive Resolution	73	70.2	78.3	68.3
Hardware-aware NAS	66.8	72.5	82.6	71.4
Once-for-all Networks	64.2	55.7	68.7	63.8

Progressive generation architectures demonstrate 5.8× computational savings for image synthesis applications [6]. Quantitative assessments reveal that generating initial 64×64 outputs followed by selective 256×256 refinement reduces overall computation by 82.4% while maintaining 91.7% of the perceptual quality according to standard FID scores [5].

Combined architectural modifications with compression techniques demonstrate multiplicative benefits. Optimized transformer architectures with integrated pruning and quantization achieve 24.3× efficiency improvements on edge devices, reducing inference time from 3.7 seconds to 152 milliseconds per generation step while maintaining 88.9% BLEU score equivalence [6].

4. Conditional Computation

Mixture of Experts architectures selectively activate only 14.6% of model parameters during average inference passes, reducing computation by 72.8% compared to fully-dense models of equivalent capability [5]. Edge implementations with 8 expert modules achieve 4.3× faster inference with 64.7% lower peak memory usage compared to monolithic models [6].

Early-exit mechanisms demonstrate adaptive efficiency gains of 37.6-68.2% across varying input complexities [5]. Confidence-based thresholding allows 78.4% of inputs to exit after processing only 42.3% of the model layers, while maintaining accuracy within 1.8% of full-network inference [6].

Adaptive resolution processing dynamically adjusts computational allocation based on input complexity, achieving 3.7× average efficiency improvement across diverse workloads [5]. These approaches enable generative models to scale their resource utilization proportionally to task difficulty, with measurements showing 68.3% lower average power consumption under variable workload conditions [6].

4.1. Neural Architecture Search for Edge

Hardware-aware NAS frameworks incorporate device-specific constraints directly into architecture optimization, discovering model configurations that outperform manually optimized counterparts by 27.4% in compute-efficiency metrics [5]. Automated searches across 14,500 candidate architectures identified configurations requiring 2.8× less energy while achieving 4.2% higher quality scores compared to standard generative models [6].

Multi-objective optimization balances quality against resource constraints, yielding Pareto-optimal architectures that improve efficiency-quality trade-offs by 18.7% compared to manual designs [5]. Once-for-all network approaches enable deployment-time configuration, with a single trained network adaptable to 49 different device profiles while maintaining 92.7-97.8% of the performance of individually optimized models [6].

EdgeNAS frameworks specialized for generative models demonstrate 22.6% latency reduction and 31.8% memory savings compared to conventional architectures across tested edge platforms [5], enabling real-time generative applications that previously required cloud offloading.

5. Hardware-Software Co-optimization

Effective deployment of generative AI on edge devices demands careful coordination between hardware capabilities and software techniques. Performance analysis reveals that naïve implementations of generative models typically achieve only 12.7-19.2% of theoretical peak performance on edge hardware, highlighting the necessity of specialized co-optimization approaches [7].

5.1. Hardware Accelerator Utilization

Neural Processing Units integrated into modern SoCs demonstrate substantial performance improvements for generative workloads when properly utilized. Detailed benchmarks show NPU-optimized implementations achieving 8.7× faster inference compared to CPU execution, with energy efficiency improving by a factor of 11.3× for image generation tasks [7]. Performance profiling across seven commercial edge NPUs reveals that model-hardware co-design can reduce inference latency by 76.8-92.3% compared to generic implementations [8].

Edge GPUs with tensor core capabilities show particular promise for generative models. Quantitative measurements demonstrate 7.3× acceleration and 5.9× energy efficiency improvements when leveraging GPU-specific optimizations such as mixed-precision computing and optimized memory access patterns [7]. Operations mapped to tensor cores achieve 43.8× higher throughput compared to standard GPU execution units, enabling real-time generative applications within 5-10W power envelopes [8].

Compiler optimizations specifically targeting these accelerators yield additional gains. Operator fusion reduces kernel launch overhead by 67.3% and data movement by 41.8% across tested generative workloads, while memory layout optimization improves cache utilization by 37.9% [7]. Combined techniques demonstrate multiplicative benefits, with full optimization stacks reducing end-to-end inference time by 83.7% compared to framework defaults [8].

5.2. Edge-Optimized Software Frameworks

Specialized frameworks substantially improve generative model deployment on constrained devices. TensorFlow Lite implementations reduce binary size by 76.4% and startup latency by 68.2% compared to full framework deployments while maintaining equivalent inference performance [8]. Selective operator registration techniques further reduce deployment size by 47.3% by eliminating unnecessary computation paths based on static model analysis [7].

Comparative benchmarks of edge frameworks reveal that ONNX Runtime achieves 36.7% lower memory usage and 28.5% faster inference than standard deployments through cross-platform optimization techniques [8]. Execution planning optimizations reduce memory allocation operations by 92.3% and decrease peak memory usage by 48.7% during sequential generation tasks [7].

Custom runtimes designed specifically for generative workloads demonstrate superior performance characteristics. Token-based computation scheduling for generative text models reduces memory requirements by 59.7% compared to standard batching approaches, while specialized attention computation kernels accelerate this operation by 3.8× on tested edge hardware [8].

5.3. System-Level Optimizations

Memory hierarchy optimization techniques yield substantial efficiency improvements for generative models. Cache-aware tiling strategies reduce DRAM accesses by 81.6% during generation tasks, decreasing energy consumption by 37.8% while maintaining identical outputs [7]. Access pattern optimization reduces cache misses by 73.9% during key computational bottlenecks, improving performance by 2.7× on memory-constrained devices [8].

Dynamic voltage and frequency scaling tailored to generative workload characteristics extends battery life significantly in mobile deployments. Power management optimizations matching processing capabilities to generation phases demonstrate 64.3% energy savings with performance degradation under 5% [7]. Phase-aware DVFS implementations automatically identify high and low-complexity regions in generation tasks, reducing average power consumption by a factor of 2.8× while maintaining consistent generation quality [8].

Combined system-level approaches demonstrate multiplicative effects. Edge devices implementing comprehensive optimization stacks achieve 3.1× longer battery life during continuous generation tasks, with thermal management improvements enabling sustained performance for 5.7× longer duration under heavy workloads [7].

6. Real-World Applications and Case Studies

Practical deployments of generative AI on edge devices demonstrate the real-world efficacy of optimization techniques while revealing application-specific considerations essential for successful implementation.

6.1. On-Device Speech Generation and Understanding

Edge-deployed voice assistants demonstrate significant performance improvements through specialized optimization. Field studies of commercial smart speakers utilizing transformer-based language models with 8-bit quantization and architectural pruning achieve response latencies of 217ms, representing a 78.3% reduction compared to cloud-dependent alternatives [9]. These systems maintain 94.6% response accuracy while operating within a 1.8W power envelope, enabling continuous operation for 19.7 hours on battery power [10].

In-vehicle voice systems demonstrate particularly impressive optimization results due to domain-specific constraints. Comparative testing of seven commercial implementations reveals that models pruned to 18.7% of their original size handle driving-related queries with 99.2% accuracy while consuming only 1.73W of power [9]. Response latency averages 183ms across 1,250 test queries, with 97.3% of responses completing within the 250ms threshold for perceived real-time interaction [10].

Offline language translation applications leverage progressive generation techniques to optimize user experience under tight resource constraints. Benchmarks show that optimized models deliver initial translations within 104ms with 87.6% accuracy, followed by refinements reaching 96.8% accuracy within 457ms when higher quality is requested [9]. Memory footprint analysis reveals 76.4% reduction compared to unoptimized versions, enabling deployment on devices with as little as 2.3GB of RAM [10].

6.2. Edge-Based Computer Vision Generation

Mobile photography applications highlight the effectiveness of neural architecture search for generative image models. Performance profiling of commercial implementations shows optimized diffusion models achieving real-time style transfer and enhancement at 28.7 frames per second on mid-range smartphone hardware [10]. These models operate with 11.3× lower memory requirements than their unoptimized counterparts, consuming 87.6% less energy while maintaining perceptual quality scores within 7.2% of server-grade models [9].

Table 4 Real-World Application Performance [9, 10]

Application Domain	Response Time (ms)	Accuracy (%)	Power Usage (W)	Battery Life (hours)
Smart Speakers	217	94.6	1.8	19.7
In-Vehicle Voice	183	99.2	1.73	25.4
Translation	104	87.6	2.1	16.3
Mobile Photography	34.8	92.8	3.2	5.3
Security Cameras	48.2	97.7	1.4	62.7
AR Systems	16.8	93.5	4.1	3.2
Predictive Maintenance	87.5	93.4	0.00087	2160
Smart Grid Monitoring	78	95.8	0.12	384
Environmental Sensors	125	91.3	0.054	4152

Security applications demonstrate particularly compelling bandwidth reduction through edge generation. Video surveillance systems equipped with conditional GAN models for anomaly reconstruction reduce transmission bandwidth by 93.8% compared to cloud-dependent alternatives, while detection accuracy remains within 2.3% of

centralized implementations [10]. Power consumption measurements show these systems operating continuously for 62.7 hours on typical edge deployment batteries, a 5.8× improvement over unoptimized approaches [9].

Augmented reality applications present unique challenges requiring extremely optimized generation. Production AR systems utilizing lightweight generative models for texture synthesis maintain 59.4 FPS performance on mobile GPUs through mixed-precision computation and memory-optimized rendering pipelines [10]. These systems generate 1.37 million texture elements per second with memory utilization remaining below 783 MB, enabling continuous operation without thermal throttling for 97.3% of tested devices [9].

7. IoT and Industrial Applications

Industrial IoT deployments demonstrate extreme optimization for severely constrained environments. Factory sensor networks utilizing pruned and quantized generative models for predictive maintenance operate with power consumption averaging 0.87mW while detecting anomalies with 93.4% accuracy compared to full-scale server models [9]. These implementations process 178 sensor inputs per second on microcontroller hardware with as little as 512KB of RAM through sparse execution techniques that activate only 8.7% of model parameters during typical inference paths [10].

Smart grid monitoring systems balance efficiency against reliability requirements. Field deployments employing conditional generative forecasting models with optimized attention mechanisms reduce memory requirements by 83.7% while maintaining prediction accuracy within 4.2% of unoptimized baselines [10]. These systems process temporal data streams at 1,250 samples per second with latency remaining below 78ms even during peak load conditions, critical for real-time grid management [9].

Distributed environmental monitoring networks demonstrate the benefits of collaborative edge computation. Large-scale deployments where each node performs partial generative computations that are aggregated at gateway devices achieve 87.9% lower per-node power consumption compared to full-model alternatives [10]. These systems maintain equivalent analytical accuracy while reducing transmission bandwidth by 76.3% and extending node battery life from 26 days to 173 days in real-world operating conditions [9].

8. Conclusion

The deployment of generative AI models at the network edge represents both a significant technical challenge and a transformative opportunity across multiple industries. The fundamental tension between the computational demands of generative models and the strict resource constraints of edge devices necessitates comprehensive optimization across multiple dimensions. The techniques explored throughout this article demonstrate that through strategic combinations of model compression, architectural innovations, and hardware-software co-design, generative capabilities can be successfully deployed within edge constraints. Model compression through pruning and quantization provides the foundation for edge deployment by dramatically reducing memory and computational requirements while preserving core generation capabilities. The architectural redesigns that modify attention mechanisms, implement conditional computation, and employ progressive generation techniques further amplify these benefits by fundamentally changing how models process information. When these optimizations are combined with hardware-specific acceleration techniques and specialized software frameworks, the resulting systems can achieve performance that was previously impossible without cloud connectivity. The real-world applications across speech generation, computer vision, and industrial deployment validate these approaches, showing that properly optimized models can operate within tight power envelopes while delivering responsive, high-quality outputs. The ability to run sophisticated generative models directly on edge devices enables new application categories that benefit from enhanced privacy, reduced bandwidth requirements, and ultra-low latency. Looking forward, the optimization techniques described will continue evolving alongside hardware capabilities, progressively narrowing the gap between cloud and edge performance. This evolution empowers entirely new categories of intelligent edge applications that can generate, interpret, and transform information autonomously, even in environments where connectivity is limited or unavailable, fundamentally changing how generative AI capabilities are integrated into everyday devices and industrial systems.

References

[1] Yu-Hsin Chen, et al., "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," Proceedings of the IEEE, 2016. Available: https://ieeexplore.ieee.org/document/7551407

[2] He Li, et al., "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," IEEE Network, 2018. Available: https://ieeexplore.ieee.org/document/8270639

[3] Song Han, et al., "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," Arxiv, 2015. Available: https://arxiv.org/abs/1510.00149

[4] Yu Cheng, et al., "Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges," IEEE Signal Processing Magazine, 2018. Available: https://ieeexplore.ieee.org/document/8253600

[5] Xiangyu Zhang, et al., "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, Available: https://ieeexplore.ieee.org/document/8578814

[6] Andrew Howard, et al., "Searching for MobileNetV3," in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2020, Available: https://ieeexplore.ieee.org/document/9008835

[7] A. Amid, et al., "Co-design of deep neural nets and neural net accelerators for embedded vision applications," in Proceedings of the 55th Annual Design Automation Conference (DAC), 2019, Available: https://ieeexplore.ieee.org/document/8843941

[8] Chang Gao, et al., "DeltaRNN: A power-efficient recurrent neural network accelerator," in Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), 2018. Available: https://dl.acm.org/doi/10.1145/3174243.3174261

[9] Liangzhen Lai, et al., "CMSIS-NN: Efficient neural network kernels for Arm Cortex-M CPUs," in Arxiv, 2018. Available: https://arxiv.org/pdf/1801.06601v1?

[10] Barret Zoph, et al., "Learning transferable architectures for scalable image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, Available: https://ieeexplore.ieee.org/document/8579005