Review Article

GenAI at the Edge: Comprehensive Survey on Empowering Edge Devices

Mozhgan Navardi[1], Romina Aalishah[1], Yuzhe Fu[2], Yueqian Lin[2], Hai Li[2], Yiran Chen[2], Tinoosh Mohsenin[1]

1. Johns Hopkins University, United States; 2. Duke University, United States

Generative Artificial Intelligence (GenAI) applies models and algorithms such as Large Language Model (LLM) and Foundation Model (FM) to generate new data. GenAI, as a promising approach, enables advanced capabilities in various applications, including text generation and image processing. In current practice, GenAI algorithms run mainly on the cloud server, leading to high latency and raising security concerns. Consequently, these challenges encourage the deployment of GenAI algorithms directly on edge devices. However, the large size of such models and their significant computational resource requirements pose obstacles when deploying them in resource-constrained systems. This survey provides a comprehensive overview of recent proposed techniques that optimize GenAI for efficient deployment on resource-constrained edge devices. For this aim, this work highlights three main categories for bringing GenAI to the edge: software optimization, hardware optimization, and frameworks. The main takeaways for readers of this survey will be a clear roadmap to design, implement, and refine GenAI systems for real-world implementation on edge devices.

Correspondence: papers@team.qeios.com — Qeios will forward to the authors

Introduction

Generative Artificial Intelligence (GenAI) has become a promising solution in text generation, image synthesis, and multimodal content creation. These developments often rely on large-scale models such as Large Language Models (LLMs) that achieve remarkable performance but demand large computational and memory resources. Traditionally, these models run on powerful cloud servers, which introduces latency, dependency on network connectivity, and potential privacy risks. As real-time

 doi.org/10.32388/JEU3U0

applications and data security become ever more critical, there is a growing push to embed GenAI functionalities directly into edge devices[1][2].

However, implementing high-intensive models on the edge presents significant challenges[3][4][5]. Edge devices, including drones[6], and autonomous systems[7] benefit significantly from the GenAI capabilities on devices. For instance, drones can generate real-time terrain analysis in remote areas, Autonomous systems can enhance decision-making through local models. Wearable health monitoring could generate personalized insights from biometric data while ensuring privacy through local data processing. To support these applications, specialized edge hardware such as NVIDIA Jetson, and Qualcomm AI Engine have been developed to handle the computational demands of GenAI while maintaining efficiency.

This situation calls for innovative approaches in software optimization including model compression, Neural Architecture Search (NAS). In parallel, hardware optimization including specialized accelerators, attention optimization, and dedicated frameworks address computational and energy constraints at the edge[8]. These strategies not only reduce model size and inference latency but also address privacy concerns when deploying complex models on edge devices[2]. This paper aims to survey existing methods and provide extensive details on implemented GenAI techniques on edge devices. To the best of our knowledge, there is no dedicated survey on GenAI at the edge. By reviewing state-of-the-art techniques from top-tier conferences and journals, this work offers a roadmap for researchers seeking to apply GenAI in edge. The main category of the paper is organized as follows:

- **Software Optimization:** Discusses key strategies for adapting GenAI models to edge devices, including model compression methods (pruning, quantization, and knowledge distillation), NAS, and open-source GenAI models.
- **Hardware Optimization:** Explores hardware accelerators and attention optimization to highlight how they meet GenAI's computational demands while addressing power and resource constraints on edge devices.
- **Frameworks:** Reviews frameworks to improve inference latency, memory, and overall energy efficiency.

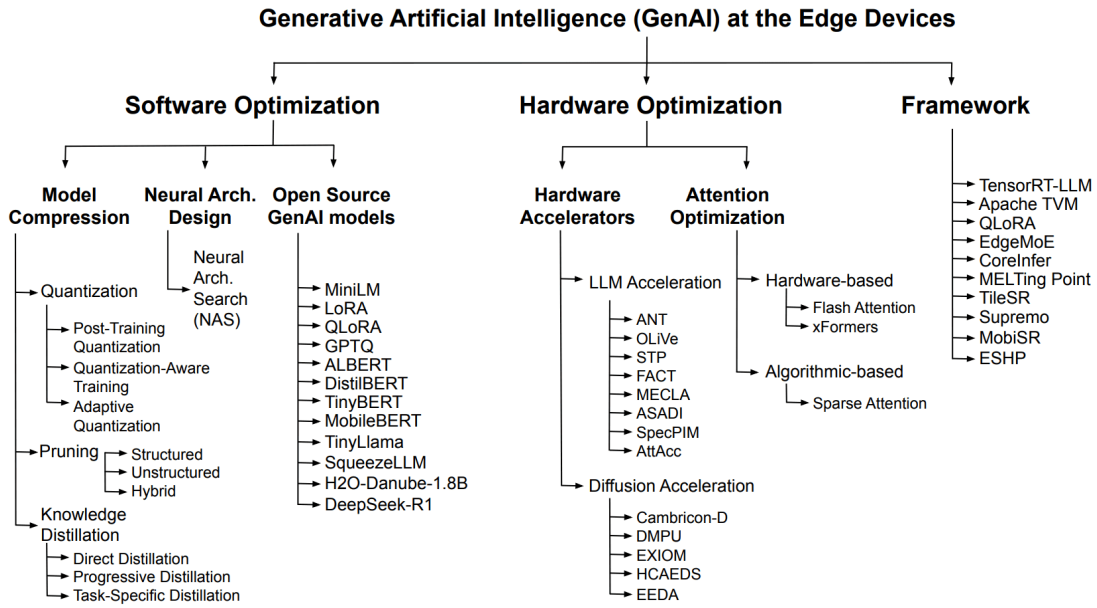

Figure 1. Illustration of the flow of GenAI at the edge

Software Optimization

Model Compression

The rapid advancement of GenAI models, while ushering in unprecedented capabilities, has also given rise to increasingly large model architectures that present significant deployment challenges[9]. Early attempts to address these challenges explored distributed mobile computing systems that could partition model computation across multiple devices[10][11].

This challenge has since prompted extensive research in model compression techniques, which have evolved along three principal directions to enable broader deployment and accessibility. Firstly, quantization techniques have achieved remarkable efficiency through reduced precision representations, particularly through enhanced activation distribution handling and hardware-optimized strategies. Secondly, methodologies for pruning have advanced from rudimentary magnitude-based techniques to sophisticated hardware-aware structured approaches, enabling considerable model reduction while preserving architectural integrity. Thirdly, knowledge distillation has evolved to incorporate progressive frameworks and multi-teacher architectures, showing particular promise in task-specific applications. Contemporary research emphasizes hardware-aware compression strategies and architecture-specific

solutions. While these advancements have enabled the deployment of foundation models with competitive performance metrics, the fundamental challenge persists in optimizing the compression-performance trade-off for edge deployment scenarios.

Quantization. Model quantization has emerged as a critical technique for deploying large-scale GenAI models on resource-constrained edge devices. Quantization approaches are broadly categorized into post-training quantization (PTQ) and quantization-aware training (QAT). PTQ methods like OPTQ[12] and AWQ[13] directly convert trained model parameters to lower precision formats, while QAT approaches such as EdgeQAT[14] incorporate quantization effects during training. PTQ methods are generally preferred due to their computational efficiency, though recent advances in both approaches have enabled effective compression through sophisticated handling of weight and activation distributions. When applied to LLMs, unique challenges emerge from their heavy-tailed weight distribution. Methods like SmoothQuant[15] and OliVe[16] address this through distribution smoothing and outlier handling techniques. Mixed-precision approaches[17] have shown promise by automatically determining optimal bit widths for different model components based on their quantization sensitivity. Recent work like OneBit[18] and BitNet[19] has pushed boundaries by demonstrating viable 1-bit quantization through sophisticated distribution-aware schemes. However, significant challenges remain in maintaining generation quality under extreme compression and developing efficient training methods for quantized LLMs on edge devices[20].

Diffusion models present their own set of quantization challenges, particularly in handling varying activation distributions across diffusion steps. Approaches like Q-DM[21], PTQD[22], and Q-Diffusion[23] tackle the challenge of varying activation distributions across diffusion steps through adaptive calibration and noise-aware quantization. Specialized temporal-aware quantization methods[24] [25] have been developed to handle the unique challenges of the iterative denoising process. Current research focuses on effectively handling dynamic activation ranges and balancing compression ratios with generation quality for edge deployment of diffusion models[26].

Pruning. Model pruning methods can be broadly categorized into structured and unstructured approaches, each with distinct trade-offs between compression efficiency and hardware compatibility. These techniques have shown particular promise in compressing large-scale generative models while maintaining performance for edge deployment. The field of LLM pruning has recently witnessed several novel approaches. Structured pruning methods like LLM-Pruner[27] and edge-optimized

approaches[28] achieve 2× speedup with minimal performance degradation by removing entire structural components.Unstructured approaches like SparseGPT[29] enable up to 60% sparsity in large-scale models, while recent advances in modality-specific pruning techniques have shown promising results across speech, vision, and multimodal domains, with methods like SpeechPrune[30] achieving up to 80% pruning rates while maintaining performance. Hardware-aware methods have become increasingly crucial, as exemplified by Flash-LLM[31], which achieves 3× inference speedup through unstructured sparsity-aware system optimization. Semi-structured pruning methods such as E-Sparse[32] further advance this direction by leveraging N:M sparsity patterns to maintain hardware compatibility while achieving high compression rates on edge devices.

In the context of diffusion models, methods like Diff-Pruning[33] achieve approximately 50% reduction in FLOPs by leveraging Taylor expansion over pruned timesteps while maintaining generative quality. Specialized approaches like LD-Pruner[34] implement task-agnostic pruning strategies for Latent Diffusion Models, while DiP-GO[35] demonstrates 4.4× speedup on Stable Diffusion without requiring retraining. Recent work combines gradient-based pruning for mask matrix continuity[36] with strategic data pruning[37], showing particular promise for edge deployment where both computational efficiency and generation quality are critical[38].

Knowledge Distillation. Knowledge Distillation (KD) has emerged as a crucial paradigm for deploying GenAI models on edge devices, with distinct approaches developed for different model architectures to balance model capabilities with computational constraints. The application of KD to language models has led to a variety of approaches. These can be categorized into white-box and black-box methods. White-box KD enables student models to match both final predictions and internal representations when the teacher model is open-source (e.g., LLaMA[39]), while black-box KD works with closed-source models (e.g., GPT-4[40]) through API calls[41]. Notable advances include MiniLLM[42], which introduces a reversed Kullback-Leibler divergence objective to stabilize student updates, and instruction-following distillation approaches that have produced efficient open-source models like Vicuna[43] and Koala[44]. Recent work in instruction-following KD has enabled compact yet capable models through supervised fine-tuning[45], while advanced applications like RLAI feedback[46] demonstrate the potential for model alignment through distillation. Adaptive distillation methods have further enhanced this field by dynamically adjusting the distillation process based on input complexity, allowing student models to focus learning where improvement is most needed[47].

In the domain of diffusion models, KD primarily focuses on accelerating sampling speed to address the challenge of high inference latency. Progressive distillation[48] represents an approach that iteratively halves sampling steps (e.g., from 1000 to 1), enabling efficient edge deployment while maintaining generation quality. Single-step approaches[49] further compress diffusion teachers into one-step generators, although this requires careful balance between efficiency and generation fidelity. Teacher-free acceleration methods like DPM-Solver[50] and consistency models[51] demonstrate effective inference cost reduction without extensive re-training. Recent advances include two-stage approaches[52] for text-conditional models and score distillation sampling[53] for 3D generation, showcasing the versatility of distillation in different applications. Also, generative dataset distillation using models like SDXL-Turbo with class-specific prompts has achieved superior images per class ratios in recent benchmarks[54], offering new possibilities for efficient model training and deployment.

Neural Architecture Design

Efficient neural architecture design has emerged as a critical research direction to address the increasing complexity and resource demands of modern models, particularly for edge devices[55][56]. By automating the generation of network architectures while considering specific hardware and constraints, computational overhead, required memory, and power consumption have been improved, while maintaining model performance.

Neural Architecture Search (NAS). Neural Architecture Search (NAS)[57][56] serves as a powerful framework to automate the design of optimal model topologies with strict latency, memory, or power budgets. By systematically exploring a predefined search space such as varying layer depth, width, or connection patterns. NAS algorithms can discover specialized architectures that outperform traditional solutions. In[57], they have proposed the first NAS using reinforcement learning (RL) to determine optimal Recurrent Neural Network (RNN) parameters. Subsequently, this idea was extended to Convonotional Neural Network (CNNs) in[58], where the authors integrated a Sequential Model-Based Optimization (SMBO) approach with a reinforcement mechanism for cell-based searches to find the best configuration.

In the context of GenAI, where large models often dominate in tasks such as text generation or image synthesis, NAS-driven architectures present a promising route to achieve efficiency. There are a limited number of work on NAS in the field of transformers[59]. FL-NAS[60] have proposed an approach which

leverages LLM to find high-performance DNNs for resource-constrained systems. Moreover, work in[61] proposed a LLM-based methodology for NAS technique in Edge devices. Puzzle[62] proposed an LLM optimized for inference using NAS under hardware constraints, achieving a 2.17x inference throughput speedup.

Open-Source GenAI models

The recent advancements in reasoning capabilities of models such as DeepSeek-R1[63] emphasize the power of open research development. DeepSeek-R1[63] has profited significantly from open-source tools like PyTorch and Meta's Llama[39]. One of the key contributions to the advancement in GenAI is open-source innovations, specifically for edge scenarios in which the resources are limited. In these cases, smaller model sizes and less latency besides not losing performance are the main considerations. Therefore, researchers explored various compression methods, leading to models like DistilBERT[64], TinyBERT[65], ALBERT[66], MobileBERT[67], MiniLM[68], and MiniLMv2[69] each using techniques such as knowledge distillation, parameter sharing, or factorization to make large models smaller while maintaining strong performance.

Beyond these compression-based strategies that are already covered in the previous sections, novelties in architecture further improved efficiency. Reformer[70] introduced locality-sensitive hashing for attention and reversible residual layers, enabling near-linear complexity for longer sequences. Meanwhile, GPT-NeoX-20B[71], LLaMA[39], and LLaMA2[72] showed how LLMs could be developed and released collaboratively, making it easier for edge-focused adaptations. Even smaller-scale of these projects such as TinyLlama[73] and H2O-Danube-1.8B[74] now offer compact language models tailored to edge constraints, continuing the trend of collaborative research. Similarly, research on instruction tuning[75], which trains models to handle various tasks by exposing them to different instructions, reinforced the importance of building flexible and open-source foundations for further innovation.

Researchers have further built on open releases to develop conversational systems, including Alpaca[76], Koala[77], and Vicuna[43], each developed by fine-tuning LLaMA[39] on curated datasets, all demonstrating competitive performance against models like ChatGPT and Bard. These models have also served as benchmarks for edge-focused projects such as SqueezeLLM[78], which introduces a post-training quantization framework to compress LLMs for more efficient inference, focusing on reducing memory bandwidth, outperforming methods like GPTQ[79], AWQ[13], and SpQR[80]. In parallel, techniques

like LoRA (Low-Rank Adaptation)[81] have reduced the cost of fine-tuning large models, accelerating domain-specific deployments. Later, QLoRA[82] tried to fine-tune a large model on a single GPU by reducing memory usage by quantizing the quantization constants and using this technique. Taken together, several open-source LLMs have been developed, and some of them are compressed to reduce their size and improve efficiency. These include MPT-7B[83], which implements a 7B-parameter architecture designed for commercial applications; DLite[84], which scales from 124M to 1.5B parameters; and RedPajama-INCITE[85], which spans 3B to 7B parameters. Open-source models and innovations can be valuable for resource-constraint applications, and be fine-tuned for specific tasks to improve their performance.

Hardware Optimization

Hardware Accelerators

Accelerator	Year	Platform	Technology	Networks	Sparsity/Quantization	Peak Energy Efficiency (TOPS/W)
EXION[86]	2025	ASIC simulator	14nm	SD/DiT	✓/ ✓@INT12	11.53
HCAEDS[87]	2024	CIM tapeout	28nm	SD	- / ✓@INT10/BF16	74.34
DMPU[88]	2024	ASIC tapeout	22nm	DDPM	✓/ -	52.01
EEDA[89]	2024	ASIC tapeout	28nm	SD	- / ✓@HYP8	4.96
Cambricon-D[90]	2024	ASIC simulator	7nm	SD	✓/ ✓@INT3/FP16	13.34
AttAcc[91]	2024	CIM simulator	7nm	LLaMA/GPT-3	- / -	2.67×DGX A100
SpecPIM[92]	2024	CIM simulator	-	LLaMA/OPT	- / -	6.7×A100
ASADI[93]	2024	CIM simulator	28nm	GPT-2/BERT	✓/ -	-
MECLA[94]	2024	ASIC simulator	28nm	LLaMA/BERT	- / ✓@INT8	7.09
STP[95]	2023	ASIC tapeout	28nm	BERT	- / ✓@FP4	18.1
OliVe[16]	2023	ASIC simulator	22nm	GPT-2/OPT/BERT	- / ✓@Adaptive 4bit	4×GOBO[96]
FACT[97]	2023	ASIC simulator	28nm	BERT	✓/ ✓@INT8	4.39

doi.org/10.32388/JEU3U0

Table 1. Hardware Accelerator for GenAI

Hardware accelerators are typically designed through the software and hardware co-design for specific networks. Algorithmically, data sparsity is enhanced by pruning, and model compression, such as quantization, reduces network size. On the hardware side, specific architectures are designed to bypass sparse or redundant computations, increase data reuse, and minimize data movement, thus enabling energy-efficient acceleration on edge devices. Generative AI (GenAI) includes GAN, LLM, and Diffusion models. While extensive hardware work has focused on optimizing GAN models[98][99][100], recent trends have shifted toward LLM and Diffusion models, driving further hardware research in GenAI. This section reviews recent efforts in optimizing hardware accelerator for LLM and Diffusion networks, with representative works summarized in Table 1.

LLM Acceleration LLM models have diverse distributions at the tensor or channel levels, numerous studies leverage customized data types to accommodate this challenge. For example, ANT[101] introduces a novel data type and employs an adaptive mechanism to determine the most appropriate type for each tensor from a predefined set. Expanding on ANT, OliVe[16] proposes an outlier-victim pair approach, which provides a more precise representation of outlier distributions in LLM models. Both ANT and OliVe incorporate specialized decoders and multiply-accumulate (MAC) units to optimize their arithmetic computation processes for LLMs. Some studies focus on reducing redundant computations in LLM models to improve the energy efficiency during inference. STP[95] proposes a computation-skipping strategy and dynamic data path reconfiguration based on entropy, achieving high energy efficiency with minimal accuracy loss. Furthermore, it has been observed that linear projections contribute significantly to the memory footprint and latency in LLM models. FACT[97] introduces an eager prediction method with a leading-one detector and log-based inner-product estimation, reducing computations in both attention and linear projections. MECLA[94] surpasses FACT by decomposing large matrices into smaller sub-matrices to minimize off-chip memory access and re-associating data on-chip for better reuse.

Recently, Computing-in-Memory (CIM) becomes a prominent approach for LLM acceleration. CIM accelerators offer significant energy efficiency gains, particularly for general matrix-matrix multiplication (GEMM) operations. Existing studies typically leverage CIM architectures to accelerate the attention mechanism, while relying on CPUs or GPUs to handle other operations. ASADI[93] introduces a

sparse attention paradigm based on diagonal compression (DIA) format, enabling highly parallel computation on CIM processors. SpecPIM[92] accelerates speculative inference in LLM by optimizing resource allocation in CIM-enabled heterogeneous systems, while AttAcc[91] accelerates batched LLM inference on CIM/NPU heterogeneous systems. Given these developments, it is expected that CIM-based accelerators for LLM models will become more prevalent in the future.

Diffusion Acceleration Diffusion networks have made significant progress recently in various GenAI tasks, with different network architecture from LLM models. These networks generate images or videos through multiple iterations of denoising operations, with highly similar images in consecutive iterations. Consequently, hardware optimizations often leverage inter- and intra-iteration similarity to accelerate Diffusion networks, typically through differential computing and skipping redundant computations.

Cambricon-D[90] introduces an approximate ReLU in the Stable Diffusion (SD) network, enabling differential computing for nonlinear functions and addressing the memory overhead associated with full-precision nonlinear calculations in traditional differential computing architectures. DMPU[88] observes that many pixels exhibit minimal changes between consecutive time steps in Diffusion models, and thus proposes a semantic-segment sparse convolution along with a trivial attention exponent inheritance method to skip redundant computations in both the convolution and attention mechanisms, significantly enhancing the energy efficiency. EXION[86] presents an FFN-Reuse algorithm that can be applied across iterations, along with an improved eager prediction method for predicting attention scores, which reduces redundant computations and boosts throughput. HCAEDS[87] is the first heterogeneous CIM chip designed for Diffusion models, incorporating a Sign-Magnitude radix-8 Booth CIM macro for integer data and a four-operand exponent CIM macro for floating-point data, achieving a high energy efficiency.

Numerous GenAI hardware studies[90][89][102][103] have observed that nonlinear functions (such as softmax, GeLU, etc.) can introduce significant latency overhead during the hardware acceleration. These studies optimize nonlinear functions to enhance overall throughput. Additionally, some studies[104][105][106][107] have focused specifically on optimizing nonlinear functions and have designed specialized hardware to facilitate network inference. All of these studies indicate a potential research trend on optimizing nonlinear functions in GenAI networks. Combined with techniques such as eliminating redundant computations and data compression, these approaches can enhance hardware acceleration and improve energy efficiency for GenAI systems.

Attention Optimization

Transformers have become the backbone of many GenAI models, but their multi-head self-attention mechanism can dominate runtime and memory usage. Therefore, researchers have explored a range of strategies to optimize attention on *hardware* and *algorithmic* levels.

Hardware-based. FlashAttention[108] reorders attention operations to reduce the number of reads and writes between GPU high bandwidth memory (HBM) and on-chip static RAM (SRAM) by splitting queries, keys, and values into smaller blocks, recomputing attention on-chip during the backward pass, and fusing multiple GPU kernels into one. Built on this, FlashAttention-2[109] takes the foundation of memory efficiency and adds better parallelism and work distribution to further increase speed and GPU utilization, especially for longer sequences. Then, FlashAttention-3[110] introduces asynchrony and low-precision computation to further optimize the attention mechanism for modern GPU architectures, which allows for even higher performance and efficiency, along with reduced error for low-precision (FP8) computing. Besides these, xFormers[111], a PyTorch-based library, provides a collection of optimized attention and Transformer blocks, including custom GPU kernels and memory-efficient attention implementations.

Algorithmic-based. Work on sparse attention reduces the quadratic complexity of self-attention by ignoring parts of the input that do not affect the result significantly. Child et al.[112] pioneered this approach by limiting attention to strided patterns using sparse factorizations of the attention matrix to reduce computation cost while maintaining performance on sequence models. Subsequent techniques like Longformer[113] by using a combination of sliding window local attention and task-motivated global attention, Big Bird[114] by combining random, windowed, and global attention to create a sparse attention mechanism, and Linformer[115] by decomposing attention with linear projections to achieve linear complexity introduced various structured sparsity patterns. Meanwhile, Choromanski et al. [116] developed performer, which uses random feature maps to approximate the softmax function, reducing its time complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.

Frameworks

Deploying GenAI models on edge devices might bring challenges because of limited computational power, memory, and latency requirements. To address these constraints, researchers have explored various techniques that simplify computations at both the graph and operator levels. By fusing kernels,

reducing redundant operations or parameters, and customizing algorithms to the hardware, these methods enable fast inference for tasks such as large language modeling, super-resolution, and more.

NVIDIA TensorRT and Apache TVM are pioneered compiler-based optimizations by combining graph-level fusion and quantization with lower latency. Likewise, Google's EdgeTPU and Coral stacks enable rapid deployment of compressed models through low-power hardware and software stack. TensorRT-LLM[117] is also a specialized toolkit for accelerating LLM inference on GPUs, including optimized CUDA kernels for attention computations, inflight batching, and quantization.

Beyond these compilers, researchers have developed frameworks customized for various GenAI workloads. For instance, Yi et al. proposed EdgeMoE[118], an engine specifically optimized for Mixture-of-Experts (MoE) language models. By using expert-wise bitwidth adaptation, it supports models with a large number of parameters on edge devices to reduce inference times substantially. Wang et al. introduced CoreInfer[119], achieving over 10$\times$ speedup compared to the Huggingface implementation through semantic-based sparse activation that identifies, fixes, and maintains stable neuron activation patterns at the sentence level. Laskaridis et al. introduced MELTing point[120], a mobile benchmarking suite designed to evaluate LLM performance, focusing on energy usage and memory footprints, across smartphones and Jetson platforms. TinyChatEngine[121] is also, an on-device LLM/VLM Inference Library that uses compression techniques to limit memory budgets while maintaining interactive response times on edge hardware. Furthermore, Nikoghosyan et al. showed that applying TensorRT to Transformer-based models on NVIDIA Jetson Xavier yields over 60% latency reduction with negligible accuracy loss[122].

In addition to language models, solutions target Super-Resolution (SR) and other vision-based generators. Chen et al. introduced TileSR[123], which splits ultra-high-resolution images into tiles and selects the ones with the highest upscaling difficulty; these tiles are processed in parallel across multiple devices, reducing latency by up to 82% and improving the image quality up to 10% compared to other alternatives such as Supremo[124] and MobiSR[125]. Wang et al.[126] proposed ESHP, which combines a difficulty predictor with deep reinforcement learning to distribute SR tasks among CPUs, GPUs, and NPUs, speeding up SR processing without modifying the original architecture of the given SR model. Zhao et al. demonstrated a full-stack SR acceleration framework for embedded GPU devices, which outperformed standard TensorRT baselines in speed due to dictionary compression and operations optimization[127].

FPGAs also provide a promising platform for runtime acceleration. Li et al. proposed a lookup-table (LUT)–based SR pipeline making sharper images while using much less energy without losing image quality[128]. Other research has combined FFT-based processing with efficient multipliers[129], designed heterogeneous CNN-SNN architectures[130], or combined FPGA and GPU via PCIe to achieve real-time SR in microscopic imaging[131]. For video-specific scenarios, Kim et al. employed pipeline and memory optimizations to reach 60 fps on 4K UHD content[132], while Sun et al. developed RNN compression techniques to manage temporal correlations[133]. On larger multi-core systems Georgis et al. attained speedups over CPU-only baselines via parallelization[134], and Liu et al. achieved real-time 4K SR on edge FPGAs through a DSP-enhanced caching scheme[135]. Finally, several system-level revisions help further reduce overhead. Fan et al.[136] leveraged codec-side data to skip redundant decoding in video SR, improved performance by up to 9.4×. Deformable 3D convolutional networks, essential in video tasks, were accelerated through tile decoupling and memory optimization by Zhang et al.[137]. Even resource-limited devices like the Raspberry Pi can support real-time SR: Osorno-Ortiz et al. integrated 2D-DWT with parallel interpolation to handle HD images in a short time[138].

Conclusion and Future Work

This work proposed a comprehensive survey regarding deploying Generative AI (GenAI) on edge devices. It presents a promising path toward reducing latency, enhancing data privacy, and enabling real-time capabilities in various applications. This survey has showcased the critical roles of software optimization, hardware specialization, and on-device inference frameworks in overcoming the resource constraints typical of embedded systems. Despite these advancements, significant challenges persist especially regarding model personalization, and security across distributed edge nodes. By effectively addressing these challenges and combining these techniques with ongoing optimizations in model design and hardware acceleration, researchers and practitioners can pave the way for even more efficient, scalable, and privacy-preserving GenAI solutions at the edge.

References

1. ^ *Nezami Z, et al. (2024). "Generative AI on the Edge: Architecture and Performance Evaluation". arXiv preprint arXiv:2411.17712. Available from: arXiv:2411.17712.*

2. ^a, ^b*Navardi M, et al. MetaTinyML: End-to-End Metareasoning Framework for TinyML Platforms. IEEE Embedded Systems Letters. **16**(4): 393–396. 2024.*
3. ^*Pourmehrani H, et al. FAT-RABBIT: Fault-Aware Training towards Robustness Against Bit-flip Based Attacks in Deep Neural Networks. In: 2024 IEEE International Test Conference (ITC). IEEE; 2024. p. 106–110.*
4. ^*Kallakuri U, et al. Resource-Aware Saliency-Guided Differentiable Pruning for Deep Neural Networks. In: Proceedings of the Great Lakes Symposium on VLSI 2024. 2024. p. 694-699.*
5. ^*Humes E, et al. Squeezed Edge YOLO: Onboard Object Detection on Edge Devices. arXiv preprint arXiv:2312.11716. 2023.*
6. ^*Navardi M, et al. MLAE2: Metareasoning for latency-aware energy-efficient autonomous nano-drones. In: 2023 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE; 2023. p. 1-5.*
7. ^*Manjunath T, et al. Reprohrl: Towards multi-goal navigation in the real world using hierarchical agents. On 37th AAAI Conference on Artificial Intelligence. In: The 1st Reinforcement Learning Ready for Production workshop; 2023.*
8. ^*Ali AH, et al. (2024). "Energy-Aware FPGA Implementation of Spiking Neural Network with LIF Neurons". arXiv preprint arXiv:2411.01628.*
9. ^*Guo C, et al. A Survey: Collaborative Hardware and Software Design in the Era of Large Language Models. arXiv preprint arXiv:2410.07265. 2024.*
10. ^*Mao J, et al. MoDNN: Local distributed mobile computing system for Deep Neural Network. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017. 2017. p. 1396-1401. doi:10.23919/DATE.2017.7927211.*
11. ^*Mao J, et al. MeDNN: A distributed mobile system with enhanced partition and deployment for large-scale DNNs. In: 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE; 2017. p. 751–756.*
12. ^*Frantar E, et al. (2023). "OPTQ: Accurate Quantization for Generative Pre-trained Transformers." In The Eleventh International Conference on Learning Representations. Available from: https://openreview.net/forum?id=tcbBPnfwxS.*
13. ^a, ^b*Lin J, et al. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. In: Gibbons P, Pekhimenko G, De Sa C, editors. Proceedings of Machine Learning and Systems. 2024; **6**:87-100. Available from: https://proceedings.mlsys.org/paper_files/paper/2024/file/42a452cbafa9dd64e9ba4aa95cc1ef21-Paper-Conference.pdf.*

14. ^Shen X, et al. (2024). "EdgeQAT: Entropy and Distribution Guided Quantization-Aware Training for the Acceleration of Lightweight LLMs on the Edge". arXiv preprint arXiv:2402.10787.
15. ^Xiao G, et al. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. In: Proceedings of the 40th International Conference on Machine Learning; 2023.
16. a, b, cGuo C, Tang J, Hu W, Leng J, Zhang C, Yang F, Liu Y, Guo M, Zhu Y (2023). "Olive: Accelerating large language models via hardware-friendly outlier-victim pair quantization." Proceedings of the 50th Annual International Symposium on Computer Architecture. 1–15.
17. ^Chen Z, et al. Channel-wise mixed-precision quantization for large language models. arXiv preprint arXiv:2410.13056. 2024.
18. ^Xu Y, et al. OneBit: Towards Extremely Low-bit Large Language Models. arXiv preprint arXiv:2402.11295. 2024.
19. ^Wang H, et al. (2023). "Bitnet: Scaling 1-bit transformers for large language models." arXiv. arXiv:2305.10403.
20. ^Egiazarian V, et al. Extreme Compression of Large Language Models via Additive Quantization. arXiv 2024. Available from: arXiv:2401.06118.
21. ^Li Y, et al. (2023). "Q-DM: An Efficient Low-bit Quantized Diffusion Model". In: Advances in Neural Information Processing Systems, 76680–76691.
22. ^He Y, et al. (2023). "PTQD: Accurate Post-Training Quantization for Diffusion Models." In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, editors. Advances in Neural Information Processing Systems. Curran Associates, Inc.; 2023. **36**:13237-13249. Available from: https://proceedings.neurips.cc/paper_files/paper/2023/file/2aab8a76c7e761b66eccaca0927787de-Paper-Conference.pdf.
23. ^Li X, et al. Q-Diffusion: Quantizing Diffusion Models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). 2023 Oct; p. 17535-17545.
24. ^So J, et al. Temporal Dynamic Quantization for Diffusion Models. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, editors. Advances in Neural Information Processing Systems. Curran Associates, Inc.; 2023. p. 48686-48698. Available from: https://proceedings.neurips.cc/paper_files/paper/2023/file/983591c3e9a0dc94a99134b3238bbe52-Paper-Conference.pdf.
25. ^Huang Y, et al. TFMQ-DM: Temporal Feature Maintenance Quantization for Diffusion Models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2024. p. 7362–7371.
26. ^Yao Y, et al. Timestep-Aware Correction for Quantized Diffusion Models. In: European Conference on Computer Vision (ECCV); 2024.

27. ^Ma X, et al. LLM-Pruner: On the Structural Pruning of Large Language Models. In: Advances in Neural Information Processing Systems; 2023.

28. ^Khiabani YS, et al. Optimizing Small Language Models for In-Vehicle Function-Calling. arXiv preprint arXiv:2501.02342. 2025.

29. ^Frantar E, Alistarh D (2023). "SparseGPT". arXiv. arXiv:2307.00026.

30. ^Lin Y, et al. SpeechPrune: Context-aware Token Pruning for Speech Information Retrieval. arXiv preprint arXiv:2412.12009. 2024.

31. ^Xia H, et al. Flash-LLM: Enabling Cost-Effective and Highly-Efficient Large Generative Model Inference with Unstructured Sparsity. 2023. arXiv:2308.04528.

32. ^Li Y, et al. E-sparse: Boosting the large language model inference through entropy-based n: M sparsity. 2023. arXiv:2310.12843.

33. ^Fang G, et al. (2023). "Structural pruning for diffusion models". In: Advances in Neural Information Processing Systems.

34. ^Castells T, et al. LD-Pruner: Efficient Pruning of Latent Diffusion Models using Task-Agnostic Insights. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. 2024:821-830.

35. ^Zhu H, et al. DiP-GO: A Diffusion Pruner via Few-step Gradient Optimization. arXiv preprint arXiv:2410.16942. 2024. Available from: https://arxiv.org/abs/2410.16942.

36. ^Wan B, et al. Pruning for Sparse Diffusion Models based on Gradient Flow. arXiv. 2025. Available from: arXiv:2501.01101.

37. ^Briq R, et al. Data Pruning in Generative Diffusion Models. arXiv [cs.LG]. 2024. Available from: https://arxiv.org/abs/2411.12523.

38. ^Yan C, et al. Hybrid SD: Edge-Cloud Collaborative Inference for Stable Diffusion Models. arXiv. 2024. Available from: arXiv:2410.02453.

39. a, b, c, dTouvron H, et al. (2023). "LLaMA: Open and Efficient Foundation Language Models". arXiv preprint arXiv:2302.13971. Available from: https://arxiv.org/abs/2302.13971.

40. ^**OpenAI** (2024). "GPT-4 Technical Report". arXiv. arXiv:2303.08774.

41. ^Liu C, et al. Evolving Knowledge Distillation with Large Language Models and Active Learning. In: Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024). 2024. p. 6717-6731. ELRA and ICCL.

42. ^Gu Y, et al. Mini{LLM}: Knowledge Distillation of Large Language Models. In: The Twelfth International C onference on Learning Representations; 2024. Available from: https://openreview.net/forum?id=5h0qf7IBZ Z.

43. a, bChiang WL, Li Z, Lin Z, Sheng Y, Wu Z, Zhang H, Zheng L, Zhuang S, Zhuang Y, Gonzalez JE, Stoica I, Xing EP (2023). "Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality". https://lmsys. org/blog/2023-03-30-vicuna/.

44. ^Geng X, et al. Koala: A Dialogue Model for Academic Research. 2023.

45. ^Wu M, et al. Lamini: Large Language Model Approaches for Generating Assistive Programs. Proceedings o f the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers). 2024:944-964.

46. ^Lee H, et al. (2023). "RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback". ar Xiv. arXiv:2309.00267.

47. ^Liang Z, et al. Dynamic Self-adaptive Multiscale Distillation from Pre-trained Multimodal Large Model for Efficient Cross-modal Representation Learning. arXiv preprint arXiv:2404.10838. 2024.

48. ^Salimans T, Ho J (2022). "Progressive Distillation for Fast Sampling of Diffusion Models". In: International Conference on Learning Representations. Available from: https://openreview.net/forum?id=TIdIXIpzhoI.

49. ^Luhman E, Luhman T (2021). "Knowledge distillation in iterative generative models for improved samplin g speed". arXiv preprint arXiv:2101.02388. Available from: https://arxiv.org/abs/2101.02388.

50. ^Lu C, et al. DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. I n: Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K, Oh A, editors. Advances in Neural Information Proce ssing Systems. Curran Associates, Inc.; 2022. p. 5775-5787. Available from: https://proceedings.neurips.cc/pa per_files/paper/2022/file/260a14acce2a89dad36adc8eefe7c59e-Paper-Conference.pdf.

51. ^Song Y, et al. Consistency Models. In: Proceedings of the 40th International Conference on Machine Learni ng. 2023. p. 32211–32252.

52. ^Meng C, et al. On distillation of guided diffusion models. In: Proceedings of the IEEE/CVF Conference on Co mputer Vision and Pattern Recognition. 2023. p. 14297–14306.

53. ^Poole B, et al. DreamFusion: Text-to-3D using 2D Diffusion. In: The Eleventh International Conference on L earning Representations; 2023. Available from: https://openreview.net/forum?id=FjNys5c7VyY.

54. ^Su D, et al. Generative Dataset Distillation Based on Diffusion Model. In: Proceedings of the European Conf erence on Computer Vision (ECCV), Workshop; 2024.

55. ^Howard AG, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861. 2017. Available from: https://arxiv.org/abs/1704.04861.

56. a, bElsken T, et al. (2019). "Neural architecture search: A survey". Journal of Machine Learning Research.

57. a, bZoph B (2016). "Neural architecture search with reinforcement learning". arXiv preprint arXiv:1611.01578. Available from: https://arxiv.org/abs/1611.01578.

58. ^Zoph B, et al. (2018). "Learning transferable architectures for scalable image recognition." In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8697–8710.

59. ^Liu Z, et al. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. arXiv preprint arXiv:2402.14905. 2024. Available from: https://arxiv.org/abs/2402.14905.

60. ^Qin R, et al. FL-NAS: Towards Fairness of NAS for Resource Constrained Devices via Large Language Models. In: Proceedings of the 29th Asia and South Pacific Design Automation Conference, ASPDAC '24, 2024. p. 429–434. IEEE Press. ISBN 9798350393545. doi:10.1109/ASP-DAC58780.2024.10473847.

61. ^Benmeziane H, Maghraoui KE (2024). "Are Large Language Models Good Neural Architecture Generators for Edge?" In 2024 IEEE International Conference on Edge Computing and Communications (EDGE), 162-165. doi:10.1109/EDGE62653.2024.00029.

62. ^Bercovich A, et al. Puzzle: Distillation-Based NAS for Inference-Optimized LLMs. arXiv preprint arXiv:2411.19146. 2024.

63. a, bDeepSeek-AI, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv preprint arXiv:2501.12948. 2025. doi:10.48550/arXiv.2501.12948.

64. ^Sanh V, et al. (2019). "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter". arXiv preprint arXiv:1910.01108.

65. ^Jiao X, et al. (2020). "TinyBERT: Distilling BERT for Natural Language Understanding". arXiv preprint arXiv:1909.10351.

66. ^Lan Z, et al. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In: International Conference on Learning Representations (ICLR); 2020.

67. ^Sun Z, et al. MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices. arXiv preprint arXiv:2004.02984. 2020.

68. ^Wang W, et al. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. 2020. Available from: https://arxiv.org/abs/2002.10957.

69. ^Wang W, et al. MiniLMv2: Multi-Head Self-Attention Relation Distillation for Compressing Pretrained Transformers. 2021. Available from: https://arxiv.org/abs/2012.15828.

70. ^Kitaev N, et al. Reformer: The Efficient Transformer. In: International Conference on Learning Representations (ICLR); 2020.

71. ^Black S, et al. GPT-NeoX-20B: An Open-Source Autoregressive Language Model. arXiv preprint arXiv:2204.06745. 2022.

72. ^Touvron H, et al. (2023). "Llama 2: Open Foundation and Fine-Tuned Chat Models". arXiv preprint arXiv:2307.09288.

73. ^Zhang P, et al. TinyLlama: An Open-Source Small Language Model. arXiv preprint arXiv:2401.02385. 2024.

74. ^Singer P, et al. H2O-Danube-1.8B Technical Report. arXiv preprint arXiv:2401.16818. 2024. Available from: https://arxiv.org/abs/2401.16818.

75. ^Chung HW, et al. (2022). "Scaling Instruction-Finetuned Language Models". arXiv preprint arXiv:2210.11416.

76. ^Taori R, et al. Alpaca: A Strong, Replicable Instruction-Following Model. 2023. Available from: https://crfm.stanford.edu/2023/03/13/alpaca.html.

77. ^Geng X, et al. Koala: A Dialogue Model for Academic Research. 2023. Available from: https://bair.berkeley.edu/blog/2023/04/03/koala/.

78. ^Kim S, et al. SqueezeLLM: Dense-and-Sparse Quantization. In: Proceedings of the 41st International Conference on Machine Learning (ICML). 2024. doi:10.48550/arXiv.2306.07629.

79. ^Frantar E, et al. (2022). "GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers". arXiv preprint arXiv:2210.17323. doi:10.48550/arXiv.2210.17323.

80. ^Dettmers T, et al. SpQR: A Sparse-Quantized Representation for Near-Lossless LLM Weight Compression. arXiv preprint arXiv:2306.03078. 2023. doi:10.48550/arXiv.2306.03078. {Extended Preprint}.

81. ^Hu E, et al. LoRA: Low-Rank Adaptation of Large Language Models. arXiv preprint arXiv:2106.09685. 2021.

82. ^Dettmers T, et al. QLoRA: Efficient Finetuning of Quantized LLMs. Advances in Neural Information Processing Systems. 2023:10088–10115.

83. ^MosaicML NLP Team (2023). "Introducing MPT-7B: A new standard for open-source, commercially usable LLMs". https://www.mosaicml.com/blog/mpt-7b.

84. ^**AI Squared** (2023). DLite V2. Available from: https://huggingface.co/aisquared/dlite-v2-774m.

85. ^Together Computer (2023). RedPajama: An Open Source Recipe to Reproduce LLaMA training dataset [software]. Available from: https://github.com/togethercomputer/RedPajama-Data.

86. [a], [b] *Heo J, et al. EXION: Exploiting Inter-and Intra-Iteration Output Sparsity for Diffusion Models. arXiv prepr int arXiv:2501.05680. 2025.*

87. [a], [b] *Guo R, et al. 20.2 A 28nm 74.34TFLOPS/W BF16 Heterogenous CIM-Based Accelerator Exploiting Denoisi ng-Similarity for Diffusion Models. In: 2024 IEEE International Solid-State Circuits Conference (ISSCC). 202 4; **67**:362-364. doi:10.1109/ISSCC49657.2024.10454308.*

88. [a], [b] *Qin Y, et al. A 52.01 TFLOPS/W Diffusion Model Processor with Inter-Time-Step Convolution-Attention-R edundancy Elimination and Bipolar Floating-Point Multiplication. In: 2024 IEEE Symposium on VLSI Tech nology and Circuits (VLSI Technology and Circuits). 2024. p. 1-2. doi:10.1109/VLSITechnologyandCir46783.2 024.10631322.*

89. [a], [b] *Yoo S, et al. A 28nm 4.96 TOPS/W End-to-End Diffusion Accelerator with Reconfigurable Hyper-Precisio n and Unified Non-Matrix Processing Engine. In: 2024 IEEE European Solid-State Electronics Research Conf erence (ESSERC). 2024. p. 253-256. doi:10.1109/ESSERC62670.2024.10719558.*

90. [a], [b], [c] *Kong W, et al. Cambricon-D: Full-Network Differential Acceleration for Diffusion Models. In: 2024 AC M/IEEE 51st Annual International Symposium on Computer Architecture (ISCA); 2024. doi:10.1109/ISCA590 77.2024.00070.*

91. [a], [b] *Park J, et al. AttAcc! Unleashing the Power of PIM for Batched Transformer-based Generative Model Infe rence. In: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2; 2024.*

92. [a], [b] *Li C, et al. SpecPIM: Accelerating Speculative Inference on PIM-Enabled System via Architecture-Dataflo w Co-Exploration. In: Proceedings of the 29th ACM International Conference on Architectural Support for P rogramming Languages and Operating Systems, Volume 3; 2024.*

93. [a], [b] *Li H, et al. ASADI: Accelerating Sparse Attention Using Diagonal-based In-Situ Computing. In: 2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE; 2024.*

94. [a], [b] *Qin Y, et al. MECLA: Memory-Compute-Efficient LLM Accelerator with Scaling Sub-matrix Partition. In: 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA). 2024. p. 1032-104 7. doi:10.1109/ISCA59077.2024.00079.*

95. [a], [b] *Tambe T, et al. 2023. "22.9 A 12nm 18.1 TFLOPs/W sparse transformer processor with entropy-based earl y exit, mixed-precision predication and fine-grained power management". In: 2023 IEEE International Soli d-State Circuits Conference (ISSCC). IEEE. p. 370–372.*

96. [^] *Zadeh AH, et al. (2020). "GOBO: Quantizing Attention-Based NLP Models for Low Latency and Energy Effi cient Inference." In 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 8*

11-824. doi:10.1109/MICRO50266.2020.00071.

97. ^[a] ^[b] *Qin Y, et al. FACT: FFN-Attention Co-optimized Transformer Architecture with Eager Correlation Predict ion. In: Proceedings of the 50th Annual International Symposium on Computer Architecture, ISCA '23. New York, NY, USA: Association for Computing Machinery; 2023. Article No. 22. doi:10.1145/3579371.3589057. ISB N 9798400700958.*

98. ^ *Chen F, et al. ReGAN: A pipelined ReRAM-based accelerator for generative adversarial networks. In: 2018 2 3rd Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE; 2018.*

99. ^ *Kim S, et al. An Energy-Efficient GAN Accelerator with On-chip Training for Domain Specific Optimizatio n. In: 2020 IEEE Asian Solid-State Circuits Conference (A-SSCC). 2020. p. 1-4. doi:10.1109/A-SSCC48613.2020. 9336128.*

100. ^ *Kang S, et al. GANPU: An Energy-Efficient Multi-DNN Training Processor for GANs With Speculative Dual-Sparsity Exploitation. IEEE Journal of Solid-State Circuits. **56**(9): 2845-2857. doi:10.1109/JSSC.2021.3066572.*

101. ^ *Guo C, et al. Ant: Exploiting adaptive numerical data type for low-bit deep neural network quantization. I n: 2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE; 2022. p. 1414–1433.*

102. ^ *Yang G, et al. SDA: Low-Bit Stable Diffusion Acceleration on Edge FPGAs. In: 2024 34th International Conf erence on Field-Programmable Logic and Applications (FPL). IEEE; 2024.*

103. ^ *Wang X, et al. DTrans: A Dataflow-transformation FPGA Accelerator with Nonlinear-operators fusion aimi ng for the Generative Model. In: 2024 34th International Conference on Field-Programmable Logic and App lications (FPL). IEEE; 2024.*

104. ^ *Fu Y, et al. (2024). "SoftAct: A High-Precision Softmax Architecture for Transformers Supporting Nonlinear Functions". IEEE Transactions on Circuits and Systems for Video Technology. **34** (9). doi:10.1109/TCSVT.202 4.3386779.*

105. ^ *Dong P, et al. Genetic Quantization-Aware Approximation for Non-Linear Operations in Transformers. In: Proceedings of the 61st ACM/IEEE Design Automation Conference (DAC); 2024. p. 220.*

106. ^ *Stevens JR, et al. (2021). "Softermax: Hardware/Software Co-Design of an Efficient Softmax for Transform ers." In 2021 58th ACM/IEEE Design Automation Conference (DAC), 469-474. doi:10.1109/DAC18074.2021.95 86134.*

107. ^ *Yan B, et al. RRAM-based spiking nonvolatile computing-in-memory processing engine with precision-co nfigurable in situ nonlinear activation. In: 2019 Symposium on VLSI Technology. IEEE; 2019. p. C258–C259.*

108. ^ *Dao T, et al. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. arXiv preprint arXiv:2205.14135. 2022.*

109. ^Dao T (2023). "FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning". arXiv preprint arXiv:2307.08691.

110. ^Shah J, et al. FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision. arXiv preprint arXiv:2407.08608. 2024.

111. ^Lefaudeux B, et al. xFormers: A modular and hackable Transformer modelling library. 2022. Available from: https://github.com/facebookresearch/xformers.

112. ^Child R, et al. 2019. "Generating Long Sequences with Sparse Transformers". arXiv preprint arXiv:1904.10509.

113. ^Beltagy I, et al. Longformer: The Long-Document Transformer. arXiv preprint arXiv:2004.05150. 2020.

114. ^Zaheer M, et al. Big Bird: Transformers for Longer Sequences. In: Advances in Neural Information Processing Systems; 2020.

115. ^Wang S, et al. Linformer: Self-Attention with Linear Complexity. arXiv preprint arXiv:2006.04768. 2020.

116. ^Choromanski K, et al. Rethinking Attention with Performers. In: International Conference on Learning Representations (ICLR); 2021.

117. ^NVIDIA Corporation (2025). TensorRT-LLM. Available from: https://github.com/NVIDIA/TensorRT-LLM.

118. ^Yi R, et al. EdgeMoE: Fast On-Device Inference of MoE-Based Large Language Models. arXiv preprint arXiv:2308.14352. 2023.

119. ^Wang Q, et al. CoreInfer: Accelerating Large Language Model Inference with Semantics-Inspired Adaptive Sparse Activation. arXiv. 2024. Available from: arXiv:2410.18311.

120. ^Laskaridis S, et al. MELTing point: Mobile Evaluation of Language Transformers. arXiv preprint arXiv:2403.12844. 2024.

121. ^MIT-HAN-Lab (2024). TinyChatEngine: On-Device LLM Inference Library. Available from: https://github.com/mit-han-lab/TinyChatEngine.

122. ^Nikoghosyan KH, et al. Acceleration of Transformer Architectures on Jetson Xavier using TensorRT. Proceedings of Innovative Polytechnic. 2023.

123. ^Chen N, et al. TileSR: Accelerate On-Device Super-Resolution with Parallel Offloading in Tile Granularity. In: IEEE Annual International Conference on Computer Communications; 2024.

124. ^Yi J, et al. (2022). "Supremo: Cloud-assisted low-latency super-resolution in mobile devices". IEEE Transactions on Mobile Computing.

125. ^Lee R, et al. MobiSR: Efficient on-device super-resolution through heterogeneous mobile processors. In: Proceedings of the 25th Annual International Conference on Mobile Computing and Networking (MobiCom); 2019.

126. ^Wang Q, et al. (2024). "An Intelligent Co-Scheduling Framework for Efficient Super-Resolution on Edge Platforms With Heterogeneous Processors". IEEE Internet of Things Journal.

127. ^Zhao W, et al. A High-Performance Accelerator for Super-Resolution Processing on Embedded GPU. arXiv preprint arXiv:2303.08999. 2021.

128. ^Li H, et al. (2024). "An Energy-Efficient Look-up Table Framework for Super Resolution on FPGA". IEEE Transactions on Circuits and Systems for Video Technology.

129. ^Malathi L, et al. (2024). "FPGA design of FFT-based intelligent accelerator with optimized Wallace tree multiplier for image super resolution and quality enhancement". Biomedical Signal Processing and Control. doi:10.1016/j.bspc.2023.105599.

130. ^Choi J, et al. (2023). "A Resource-Efficient Super-Resolution FPGA Processor with Heterogeneous CNN and SNN Core Architecture". IEEE Transactions on Computers.

131. ^Gui D, et al. PCIe-based FPGA-GPU heterogeneous computation for real-time multi-emitter fitting in super-resolution localization microscopy. Biomedical Optics Express. 2022.

132. ^Kim Y, et al. (2019). "A Real-Time Convolutional Neural Network for Super-Resolution on FPGA With Applications to 4K UHD 60 fps Video Services". IEEE Transactions on Circuits and Systems for Video Technology.

133. ^Sun K, et al. An FPGA-Based Residual Recurrent Neural Network for Real-Time Video Super-Resolution. IEEE Transactions on Circuits and Systems for Video Technology. 2022.

134. ^Georgis G, et al. (2019). "Acceleration techniques and evaluation on multi-core CPU, GPU and FPGA for image processing and super-resolution". Journal of Real-Time Image Processing.

135. ^Liu H, et al. (2024). "A High-Performance Accelerator for Real-Time Super-Resolution on Edge FPGAs". ACM Transactions on Design Automation of Electronic Systems.

136. ^Fan H, et al. Co-ViSu: a Video Super-Resolution Accelerator Exploiting Codec Information Reuse. In: International Conference on Field-Programmable Logic and Applications (FPL); 2023.

137. ^Zhang S, et al. (2022). "An Efficient Accelerator of Deformable 3D Convolutional Network for Video Super-Resolution". IEEE Transactions on Multimedia.

138. ^Osorno-Ortiz RJ, et al. Implementation of the image super-resolution DWT based algorithm on Raspberry Pi platform for real-time applications. In: Proceedings of SPIE. 2024.

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.