On-device Generative AI: The Need, Architectures, and Challenges

Siva Sai
Birla Institute of Technology and Science Pilani, Pilani campus

Garima Dashore, Vinay Chamola
Birla Institute of Technology and Science Pilani, Pilani campus

Manish Prasad
Vellore Insititute of Technology Bhopal, Bhopal campus

Biplab Sikdar
National University of Singapore, Singapore

***Abstract*—The area of Generative Artificial Intelligence (GenAI) is rapidly expanding, as seen by the regular release of new models and applications every few months. While these GenAI models have impressive capabilities, their computational intensity has presented issues, especially in applications demanding low latency. Hence, substantial research is being conducted to develop ways to scale down these models so that they may be used for on-device computing on edge devices. Examining successful examples of GenAI models implemented on mobile devices with minimum latency becomes critical in understanding the practical consequences of these breakthroughs. Notable instances, such as the deployment of Diffusion-based GenAI models on flagship smartphones like Samsung S23 Ultra and iPhone 14, demonstrate the possibility and promise of bringing GenAI applications to consumers' fingertips. We further analyze and find out the approaches and strategies that make these on-device deployments successful.**

GENERATIVE AI (GENAI) describes a branch of artificial intelligence that is concerned with producing new material, including text, pictures, and other data. GenAI models, in contrast to typical AI systems, can generate original and creative outputs that were not explicitly built into them. Neural networks are widely used in such GenAI models to extract patterns and structures from massive datasets. Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) are two common categories of GenAI models. These models can provide fresh, realistic samples that closely match the training data after being trained on a variety of datasets. However, the majority of GenAI models in the market are extremely computationally intensive and have large model sizes, which require strong centralized computing infrastructures (such as cloud servers) to handle user requests. Moreover, the architecture for centralized computing is unsustainable, expensive, and not environmentally friendly. With the growing demand for privacy, reduced latency, and the ability to operate in environments with unreliable connectivity, there has been a notable shift towards deploying GenAI directly on edge devices, such as smartphones, IoT devices, and wearables. This paradigm is often referred to as On-device Generative AI, which provides various advantages over centralized cloud-based infrastructures (Figure 1) and unlocks many more potential for the deployment of GenAI. On-device Generative AI is poised to bring many innovative and interesting applications of GenAI through

mobile consumer gadgets [1]. However, actual implementations of these technologies are fraught with many challenges, mainly because of the limited processing power, memory, and energy resources of edge devices, which are generally unsuitable for computationally intensive tasks like that required by the generative models.

The major contributions of the paper can be highlighted as follows:

1) Our paper consolidates and systematically compares the various trending on-device optimization strategies, including quantization, model pruning, and efficient architecture designs such as Optimized Latent Diffusion (OLDN) and Advanced U-Net Optimizations (ANOT).
2) Our review offers a practical framework for understanding the trade-offs involved in deploying GenAI models on edge devices by effectively categorizing the on-device technologies based on their core underlying principles, such as latency reduction, memory efficiency, and energy consumption.
3) Additionally, our review highlights critical challenges faced by the current implementations, such as the balancing act between maintaining model accuracy and minimizing resource consumption while also drawing attention to gaps in the current research
4) The paper also emphasizes the potential of emerging techniques like neurosymbolic AI and federated learning for future advancements in the field of On-device GenAI technologies and discusses innovative areas for future research in the domain.

NEED FOR SCALING GENAI MODELS

GenAI systems have evolved rapidly in recent times by training larger and larger models with an increasing number of parameters and computational complexity, achieving superior performance across all domains. However, this extension in model sizes follows an exponential trajectory, with the GenAI models doubling in size every six months, and the computation capabilities of Central Processing Units (CPUs) and Graphics Processing Units (GPUs) in the semiconductor manufacturing sector experiences a much slower doubling rate, occurring approximately every two years. This growing disparity between the rapid growth of GenAI model sizes and the comparatively sluggish advancements in semiconductor capabilities raises concerns about an imminent misalignment between computing demand and supply.

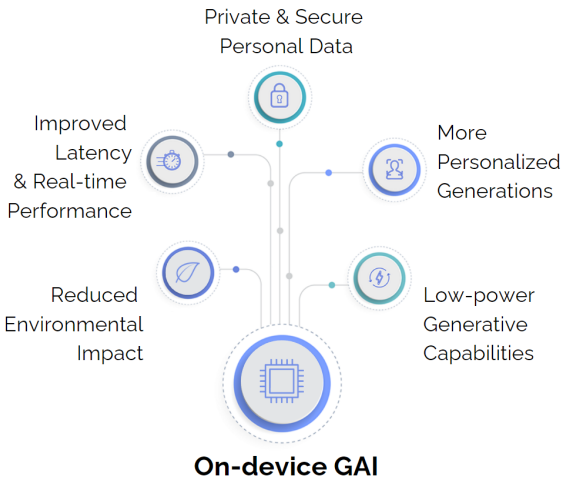

Figure 1: Advantages of on-device generative models over traditional cloud-based models

Data Privacy

Addressing the broader landscape of AI, the question of data ownership emerges as a crucial consideration. Storing user data locally, rather than relying solely on centralized databases, emerges as a potential solution to allay concerns from governing bodies and consumers regarding the misuse and theft of sensitive information. Localized storage of user data not only enhances privacy and security but also aligns with evolving ethical AI frameworks, which prioritize user autonomy and consent in data handling. Federated learning has approached as a complementary approach to this process [2], which allows for decentralized training across edge devices and ensures that individual data never leaves the user's device. This decentralized model improves privacy and maintains the benefit of training models with aggregated insights across multiple users.

Easy Access

Despite the remarkable advancements in GenAI, a significant bottleneck hindering mass adoption is the restrictive data transfer limits on local networks and the strain on centralized AI systems to handle all incoming requests. An innovative remedy to this challenge lies in the deployment of localized AI models directly on devices, such as mobile phones. This decentralized ap-

proach not only alleviates data transfer limitations but also enhances real-time processing capabilities. This approach is especially beneficial for mission-critical applications, such as autonomous driving, augmented reality (AR), and healthcare diagnostics, where delays in processing can have severe consequences. Recent developments in model quantization like being able to use reduced model precisions from 32-bit floating points (FP32) to 8-bit integers (INT8), prove to be crucial in the deployment of complex deep learning networks on resource-constrained edge devices by ensuring that models remain computationally efficient without making any compromises on their accuracy.

Personalization

The crux of GenAI's prowess lies in its immense potential for personalization through its access to vast amounts of consumer data, which is instrumental in fine-tuning models for improved inferences. By leveraging a local database, GenAI models can adapt and personalize their outputs based on individual user preferences. The shift towards On-device GenAI empowers users by enabling their personal data to be directly fed into generative models securely and privately. It opens many avenues for user-specific inferences, from personalized cooking guides that tailor recipes to individual tastes to virtual therapists providing tailored mental health support. The ability to fine-tune models also opens up new avenues for continuous learning [3]. Users benefit from AI models that evolve in response to their changing needs, preferences, and environments. In scenarios like fitness tracking or personalized healthcare, where privacy is paramount, having a model that remains entirely on-device while adapting to individual behavior ensures both accuracy and security.

Energy-efficiency and Sustainability

It is a well-known fact that GenAI models, especially large diffusion models and LLMs, are computationally very intensive, and training and running them requires a substantial amount of energy sources. The environmental impact of energy-hungry AI systems has drawn considerable attention. Therefore, by moving towards more efficient on-device processing, reliance on large, energy-intensive data centers can be reduced, and the carbon footprint associated with AI computations can be decreased. This pursuit aligns with growing industry efforts to build green AI, which prioritizes models that are both energy-efficient and effective. Recent studies [4] have demonstrated that efficient deployment of GenAI models on edge can reduce energy consumption by up to 90% compared to cloud-based systems, highlighting the importance of this trend.

ARCHITECTURE AND FOUNDATIONS

Currently, many popular GenAI models have been successful in being scaled to perform with minimal latency on mobile devices or high-powered GPU devices. Each model requires its own optimizations to run on a smaller scale. Going further, we will explore the underlying foundational principles and architectures for optimizing GenAI models for edge devices with a critical focus on the Stable Diffusion models, which were recently successful in being scaled to run on GPU-powered devices, and the SnapFusion model [5] (recently announced, last revision Oct 16, 2023) which achieved latency of only 2 sec on mobile devices (Figure 3), and few more. An overview of the literature review we have performed is illustrated in Table 1.

Optimized Latent Diffusion and U-Net Architecture (OLDN)

Many recent studies in this area have conducted experiments on Stable diffusion v1.4 (or higher) models since they are open source and many other GenAI models work on similar principles. Core ways to scale such models lie in efficient network architecture design and improving upon the step distillation process. One way to execute this is to design an efficient U-Net identifying redundancies in the original model and reducing the image decoder's computation via data distillation. Applying Group Normalization across the U-Net of the SD v1.4 model along with Winograd Convolution and improving attention module efficiency, the model was able to generate images on iPhone 14 in under 15s [6].

Group Normalization By utilizing group normalization technique, in which we apply individual normalizations on separate groups formed by breaking up the channels of the feature map, we can make the approach less dependent on the batch-size and thus generalize it across multiple different network designs and varied batch sizes. In this process, we standardize every feature value x_i using the group mean and variance of the corresponding group that it belongs to using Equation 1.

$$\hat{x}_i = \frac{1}{\sigma_g} \cdot (x_i - \mu_g) \tag{1}$$

Thus, instead of sequentially carrying out the previously specified operations — "mean," "variance," "normalize," and "reshape", a special GPU-shader kernel is created that can carry out each of aforementioned operations in a single GPU command without any intermediary tensors.

Partially Fused Softmax Attention Module efficiency was dealt with by reducing softmax operations [7] or by flash attention technique. Reducing the number of elemental softmax operations: Two steps may be distinguished in the softmax operation on the matrix $A = \frac{QK^T}{\sqrt{d}} \in \mathbb{R}^{N \times M}$ where A is attention, reducing processes and operations on elements. The computation of most significant value of each row in A and its adjusted exponential sum S, as shown in the equation below, is referred to as the reduction operations. Next, we normalize the values in A using the vectors L and S per element (Equation 2).

$$L = max_j[a_{ij}], S = [X_j \exp(a_{ij} - max_k[a_{ik}])] \in \mathbb{R}^N \tag{2}$$

Chen et al. [6] created a GPU shader to circumvent running the entire softmax computation on the big matrix A, allowing the L and S vectors to be computed throughout the reduction processes, yielding a tensor of size N × 2. The following matrix multiplication using matrix V is then combined with the element-wise softmax algorithm. This process is visualized in Figure 2.

It is important to emphasize that the computation mapping from A to L, S has restricted parallelism since the resultant tensors have fewer components than the input tensor A. We divide the pieces in A into blocks to further boost parallelism and reduce latency by dividing the reduction operations into many phases. Every block undergoes computations, which are then reduced to the outcome. Through careful memory cache management and threading, this method across multiple stages may be completed using a single instruction to the GPU, resulting in even further latency reduction.

Flash Attention It is an IO-aware, accurate attention technique that uses tiling to reduce memory reads and writes between on-chip SRAM and GPU high bandwidth memory (HBM). Because this method requires fewer HBM accesses than normal attention, it is more efficient overall and works well with a variety of SRAM sizes. Despite aiming to improve latency and decrease read/write global memory, FlashAttention's kernel requires a lot of registers. Due to this major drawback, this method is only viable on high-functioning GPUs like Adreno and Apple GPUs with smaller attention matrices of dimensions around $d = 40$. In other situations, the previously discussed partly fused softmax technique might perform better.

Winograd Convolution The convolution process is converted into a sequence of matrix multiplications using Winograd convolution. The central realization is that many of the necessary multiplications may be eliminated by carefully selecting the transformation matrices, which results in a more efficient computation. However, it also results in more memory use and numerical inaccuracies, especially when utilizing bigger tile sizes. The 3×3 convolution layers are the mainstay of Stable Diffusion; they make up more than 90 percent of the layers in the image decoder, for example. According to the findings of the authors [6], a 4 × 4 tile size is ideal for this convolution, as it perfectly balances the memory utilization and computational efficiency.

Advanced U-Net Optimization Techniques for Mobile GPUs (ANOT)

The SnapFusion model [5] employs further advanced optimization techniques for the U-Net discussed in the previous section, particularly targeting on improving the speed of inference and reducing the number of denoising steps via step distillation. The starting point for the model optimization was the Stable Diffusion v1.5 (SD-v1.5) model. The authors performed tests on the MS-COCO dataset on iPhone 14 pro and compared the performances of SnapFusion and SD-v1.5.

U-Net Architecture Optimizations The authors built an efficient U-Net by applying a self-designed algorithm. The algorithm assesses every model block and carries out robust training. It evaluates the change in CLIP score for each block to direct the architectural development process, should it become necessary at some time. To strengthen the network's resistance to variations in design, they also incorporate a training augmentation. This process allows for a steady architectural evolution and an accurate evaluation of every

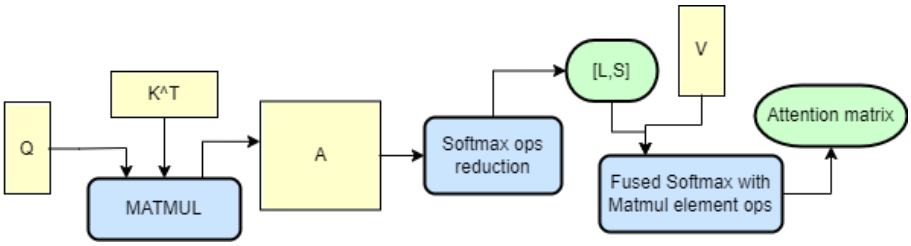

Figure 2: Optimized softmax implementation proposed for OLDN architecture

block. A validation set, a table for lookup latency with Cross-Attention and ResNet timings, and U-Net architecture are needed for the algorithm. The aim is to reach convergence of the U-Net while satisfying the latency objective.

Efficient Image Decoder The authors were successful in reducing the size of the original decoder by applying 50% uniform channel pruning, resulting in a decoder that was 1/4th the size of the original. Further optimizations were carried out by utilizing a distillation pipeline in which they used synthetic data to train the decoder in a more efficient way. Using text prompts the latent representation from the U-Net of SD-v1.5 was extracted and forwarded to the efficient image decoder, and along with SD-v1.5 was used to obtain two images. The mean squared error of the two images was minimized to further optimize the decoder. The significant advantage of using this technique is that we can enhance the dataset during the training procedure itself by sampling different noises and thus synthesizing large number of image sets from each prompt.

Step Distillation Step distillation is a technique that has become popular in the last few years. It mimics a teacher-student dynamic where we reduce the inference steps by distillation of the the teacher model to a student model that runs on fewer steps [8]. The creators of the SnapFusion model made a distillation pipeline consisting of 3 steps. They employed a step distillation approach on SD-v1.5, obtaining a U-Net with 16 steps that achieved performance comparable to a 50-step model. The direct distillation method from a 32-step SD-v1.5 was chosen over a progressive approach, as empirical observations indicated superior results. Additionally, the same distillation strategy was applied to derive a 16-step efficient U-Net, and ultimately, a final 8-step efficient U-Net was obtained by distilling from the 16-step SD-v1.5. They achieved this by mixing the two techniques of Vanilla step distillation and CFG-Aware Step distillation. They adjusted two hyperparameters, CFG range and CFG probability, to gain an effective balance of FID and CLIP scores. They also tested their proposed CFG-aware distillation on SD-v2, which obtained similar promising results when running with the same parameters used for SD-v1.5. Further tuning of the parameters may lead to better results.

Refined Cascaded Diffusion Architecture with Qualcomm AI Stack (RSCA)

For this architecture various optimizations were done utilizing the techniques of hardware acceleration and quantization on the FP32 open-source model of SD v1.5. This model was then successfully deployed on a Snapdragon 8 Gen 2 Mobile Platform processor. Qualcomm AI Research also developed post-training quantization techniques for the AI Model Efficiency Toolkit (AIMET) to shrink the model from FP32 to INT8. AIMET includes techniques like Adaptive Rounding (AdaRound), which helped to preserve the accuracy of the model [9].

AdaRound is an improved weight-rounding technique for post-training quantization. In this, the rounding job is given as a unconstrained quadratic binary optimization problem which is used to approximate the task loss using a Taylor series expansion. This can be then optimized using soft-relaxation by reducing the obtained task loss to layer-wise local loss. These techniques were applied to all the components of the Stable diffusion model. To run the model on Snapdragon 8 chip efficiently, it is compiled as a program using the Qualcomm AI Engine direct framework. Based on the Qualcomm Hexagon Processor's hardware architecture and memory hierarchy, the Qualcomm AI Engine sequences the framework activities to optimize speed and minimize memory leakage. The newest Snapdragon 8 Gen 2 aids in the effective operation of big models such as Stable Diffusion using micro-tile inferencing.

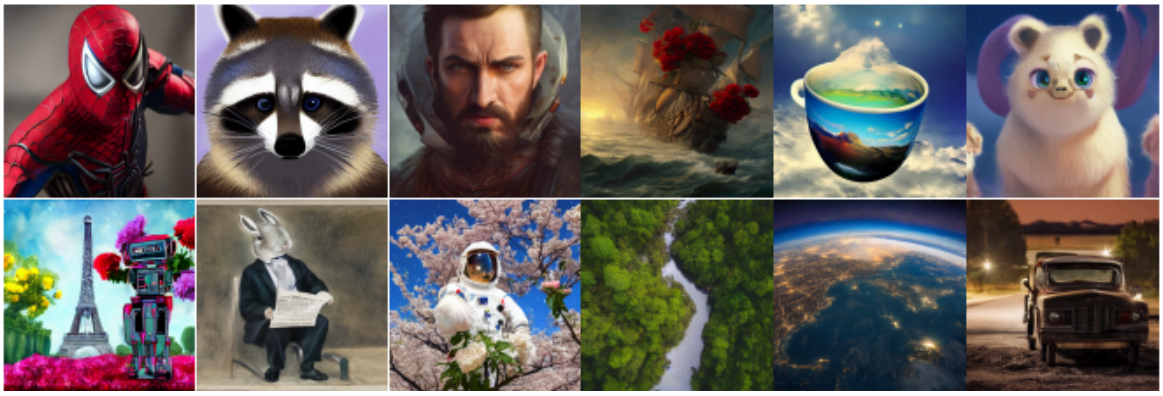

Figure 3: Some example 512x512 images [5] generated by the SnapFusion model by mobile devices in under 2 seconds

Furthermore, multi-head attention is employed across the component models in Stable Diffusion to speed up the inference process. After all these optimizations, the model was successful in generating a 512x512 image in 15 seconds after 20 inference steps of the model. With the new release of Snapdragon 8 Gen 3 and using knowledge distillation teacher-student techniques. This creates a smaller student model which does not need the same training pipeline as the teacher.

Advanced Quantized Diffusion System with TensorFlow Lite (AQDS)

This research was focused on deploying SD v-2.1 using TensorFlow Lite framework [10], which is compatible with both iOS and Android. They were successfully able to generate 512x512 images on Samsung Galaxy S23 in around 7 seconds. While the TFLite GPU delegate accelerates the majority of Stable Diffusion operators, it fails to delegate even officially supported operators when the input activation size is big. To solve the incomplete GPU delegation, they offer three techniques that include changes to the model's computation network.

Converting Fully Connected Layers of the U-Net into Equivalent Convolution Layers There are multiple fully connected layers with large inputs in the spatial transformer blocks of the denoising U-Net network. These layers often fail to be delegated, so we can convert them into equivalent convolution layers by changing FullyConnected operators into Conv2D operators everywhere.

Serialization of Conv2D Layers It was observed that due to its enormous input and output, the 3x3 convolution layer in the denoising network was unable to be delegated using the OpenCL backend. Serializing the Conv2D operator can fix this problem by lowering activation sizes, but it comes at the expense of many kernel calls. The smallest serialization factor is selected to prevent unnecessary overhead.

Broadcast Free Group Normalization In the TFLite, group normalization is represented as a computation graph composed of fundamental operators such as "Mean," "Square," "Rsqrt," and "BroadcastTo" rather than as a single operator. However, because the TFLite GPU delegate does not enable BroadcastTo, the implementation of the group normalization layer must be modified. When the activations are 4-dimensional or lower tensors, the TFLite converter does not generate an explicit BroadcastTo operator. As a result, they restructured the group normalization layer so that the dimensions of the activation tensors are no more than four.

Other methods included GELU (Gaussian Error Linear Unit) approximation to make it more numerically stable, a pipelined execution technique where the denoising network is kept in memory during the execution for the entire duration, the text encoder and picture decoder are loaded alternatively via a child thread that runs concurrently with the main thread. Additionally, to minimize total memory usage, quantization and pruning techniques were applied to the pre-trained model. The resultant memory usage of each component in the proposed pipeline is shown in Figure 4. Since mobile GPUs do not support integer matrix multiplications, the activations are performed using float16. However, to minimize model size, the authors quantized the weights into 8-bit precision;

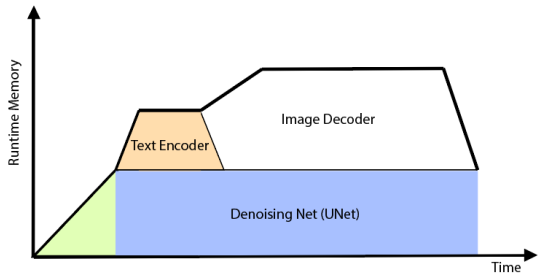

Figure 4: Memory usage of various components in the proposed AQDS pipeline

consequently, the weights were converted from 8-bit integers to 16-bit floating p oints b efore b eing used in the algorithm. Structured pruning was also used to reduce memory needs on large convolution layers.

BK-SDM: Architecturally Compressed Stable Diffusion

This presents a new approach to efficient text-to-image generation called Block-removed Knowledge-distilled Stable Diffusion Models (BK-SDMs). It proposes that classical architectural compression can be used to reduce the computational demands of Stable Diffusion models (SDMs) for general-purpose T2I synthesis [11].

This architecture showcases the effectiveness of classical architectural compression in general-purpose text-to-image synthesis by introduction of BK-SDMs which removes several of the attention and residual blocks from the U-Net of SDMs for the favor of 30 percent reduction in parameters. The results showcase the compact models to be still competitive with the larger multi-billion parameter models on the zero-shot MS-COCO benchmark. Distillation-based pre-training for the performance of BK-SDM is essential. Without the knowledge-distillation block, the models either fail in generating the subject of the prompt entirely or cannot consistently preserve the identity features. Distillation proves to be a crucial component for this technique, which enables robust pre-training even under highly constrained environments. This framework presents an innovative method for efficient text-to-image generation that complements the previous works in diffusion models. Text-to-image generation remains a challenging task that involves generating realistic images from textual descriptions. Its capabilities hold significant p otentials i n t he a reas o f n atural language, computer vision and robotics. However, the computational demands of existing models for text-to-image generation often limit their practicality for many real-world problems.

BK-SDMs can be used for personalized generation with DreamBooth fine-tuning. Comparing the results of DreamBooth fine-tuning across various pre-trained models illustrate that BK-SDMs can preserve upto 95% to 99% performance compared to regular SDMs with the added advantages of reduced fine-tuning cost and number of parameters. Recent research on efficient text-to-image generation has primarily focused on reducing the number of sampling steps and carrying out network quantization. However, this model proposes a different approach that uses classical architectural compression to reduce the computational demands of SDMs.

In conclusion, this model presents a new approach to efficient text-to-image generation that uses classical architectural compression to reduce the computational demands of SDMs. The authors demonstrate that BK-SDMs achieve competitive results against larger models and can be used for personalized generation with DreamBooth fine-tuning. They also highlight the importance of distillation-based pre-training for the performance of their method. The authors aspire for their work to encourage further research on the topic of structural compression in large diffusion models.

LLMCad Inference Architecture

A significant obstacle to LLM (Large Language Models) scalability on mobile devices is the memory wall, which causes prolonged inference delay by repeatedly releasing and loading model weights. Users are forced to pick between emergent ability and real-time generation due to this memory wall, which impedes the scaling law. For on-device generative NLP tasks, LLMCad is the first efficient inference engine that tackles this issue by delegating majority portion of the tokens to smaller real-time LLM that can be fit entirely within the device's memory [12]. LLMCad uses a special kind of model cooperation known as "generate-then-verify," which guarantees quick verification without sacrificing correctness. Compared to sequential token generation, this method offers two crucial advantages: quicker verification and no compromise in accuracy. This was achieved by three techniques:

Modifying Token Tree Generation and Verification

Instead of doing verification linearly, it takes a

Architecture	Authors	Base Model	Quantization Techniques	Deployed Platform	Inference Speed
OLDN	Chen et al. [6]	Stable Diffusion (v1.4)	• Group normalization • Partially fused softmax • Flash attention • Winograd convolution	• Samsung S23 Ultra • iPhone 14	$\sim$ 15 sec
ANOT	Li et al. [5]	Stable Diffusion (v1.5)	• U-Net optimization • Efficient image decoder • Step distillation enhancements	• iPhone 14 Pro	$\sim$ 2 sec
RSCA	Hou et al. [9]	Stable Diffusion (v1.5)	• FP32 to INT8 quantization • AIMET post-training quantization • Adaptive rounding (AdaRound)	• Snapdragon 8 Gen 2 Mobile Platform	$\sim$ 7 sec
AQDS	Choi et al. [10]	Stable Diffusion (v2.1)	• Serialization of Conv2D layers, • Broadcast-free group normalization • Fully connected to Conv2D conversion	• Samsung Galaxy S23	$\sim$ 7 sec
BK-SDM	Kim et al. [11]	Stable Diffusion (v1.5)	• Residual and attention block removal • Distillation-based pre-training • Network quantization	• General Purpose	Improved over baseline
LLMCad	Xu et al. [12]	Various GPT-based models	• Token tree generation • Self-adaptive fallback • Fine-tuning generative pipeline	• Jetson TX2 • Xiaomi 10/11	9.3x faster than baseline

Table 1: Comparative overview of model architectures and optimization strategies

different approach. Due to the token tree it employs, any token can have several possible successor tokens. The system makes use of modules such as a non-autoregressive token tree verifier, a tree decoder, and a confidence-based branch pacer.

Self Adaptive Fallback If a memory-resident LLM issues an erroneous token, this strategy will be immediately implemented. The approach evaluates the generating capabilities using past data and uses a more precise indicator known as cumulative uncertainty.

Fine-tuning Speculative Generation Pipeline To prevent the disruption of the regular verification process, LLMCad incorporates a fine-tuned pipeline that limits speculative generation only to instances where loading target LLM parameters remain below the memory upper bound. This approach stems from the insight that the verification process may not always consistently detect the errors and thus allow the speculatively generated tokens to be utilized effectively.

The performance of LLMCad was evaluated on two IoT devices (Jetson TX2 and Jetson Orin NX) as well on smartphones such as Xiaomi 10 and Xiaomi 11. LLMCad demonstrated speeds which were up to 9.3 times faster than the existing inference engines. Additionally, it could reduce the average per-token generation time by factors ranging from 2.9 to 9.3 seconds on IoT devices and 3.5 to 4.7 seconds on smartphones, all while maintaining the performance accuracy [12].

CHALLENGES IN SCALING OF GENAI MODELS

Scaling down large generative models poses a significant challenge primarily rooted in the scarcity of computational resources necessary for effective model execution. The intricacies of extensive generative models, particularly exemplified by architectures like Generative Adversarial Networks (GANs), demand substantial computational power. Deployment of these models on resource-constrained devices, such as smartphones and Internet of Things (IoT) devices, becomes a formidable task due to processing power, memory, and energy efficiency constraints.

Computational Bottlenecks

GenAI models are known for their complexity, requiring substantial processing capabilities. This complexity becomes especially apparent when attempting to run these models on devices characterized by inherent limitations in processing power. Resource-constrained devices, like smartphones and IoT devices, possess finite memory capacities, and executing large generative models can quickly exhaust these limits. Additionally, the energy efficiency of these devices is a critical concern, as continuous model computation for real-time generative activities can lead to substantial energy consumption, adversely affecting the device's battery life. Traditional approaches to mitigating these challenges involve size reduction through techniques such as pruning and quantization. However, simply reducing the size of a generative model through prun-

ing and quantization often results in a trade-off with accuracy. The loss of data or insufficient representation due to these size reduction techniques can compromise the model's capability to synthesize high-definition images, undermining the very purpose of deploying such models.

Achieving Low-Latency Inference Times

Real-time applications require constant model computations. However, the associated energy consumption can be detrimental to the device's battery life, especially in the case of edge devices. Exploring hardware acceleration solutions enables us to strike a balance between computational demands and energy conservation. In the context of on-device deployment, this challenge extends to model updates and adaptability, which must be judiciously balanced against the constraints of limited bandwidth and storage capacities inherent to these devices. Recent studies on the implementation of scaled-down GenAI models have been focused on the Stable Diffusion model, which is a Denoising Diffusion Probabilistic Model (DDPM) based model [6]. The main challenge faced in scaling down was reducing the number of iterative denoising steps.

Maintaining Accuracy and Performance During Model Compression

The techniques of knowledge distillation, quantization and pruning play a crucial role in minimizing the size and complexity of generative models to make them more viable for on-device applications. However, these techniques can lead to a reduction in model accuracy or the output quality. For instance, when applying post-training quantization (e.g., AdaRound), maintaining accuracy while significantly r educing the model's size remained a key technical challenge. Models like BK-SDM (Architecturally Compressed Stable Diffusion) reduce the number of attention and residual blocks, resulting in reduced latency, but there is always the risk of compromising image quality.

Handling Diverse Hardware Architectures

The heterogeneity of edge devices, ranging from smartphones to IoT devices to specialized accelerators like TPUs and Neural Engines, introduces yet another challenge. The model optimizations done specific to one hardware platform may not carry over well on another due to differences in architecture, processing power, and available memory. For instance, TensorFlow Lite-based models may experience incomplete GPU delegation, limiting their effectiveness on specific platforms. Consequently, the optimizations done for scaling GenAI models must be done in a way that ensures maximum compatibility and hardware independence, which can be challenging due to the need to fine-tune implementations for different hardware specifications.

Table 2 presents a comprehensive overview of the present challenges in the deployment of On-device GenAI, what is the extent of their impact in the field, and potential solutions to address them.

FUTURE RESEARCH AND DIRECTIONS

In the context of On-device GenAI, several promising techniques relevant to the topic have emerged in recent years. This section provides a detailed exploration of some of the prominent of these topics.

1) Using adaptive model compression techniques to dynamically adjust the model size and complexity based on the device's current computational resources and user needs. This can be implemented by real-time monitoring of device performance and user interactions to optimize the model's architecture on the fly using model pruning, quantization, and weight sharing. This could also solve the problem of cross-hardware incompatibility for model optimizations.
2) Integration of neurosymbolic AI [13] with On-device GenAI modules to provide a robust framework for reducing reliance on large-scale neural networks. The hybrid approach of using deep learning with neural networks and symbolic reasoning for reasoning tasks could help significantly by improving instructability, optimizing resource usage, and increasing interpretability. This provides an effective means to reduce the model size without sacrificing accuracy, making it ideal for resource-constrained devices.
3) Federated learning 2.0 which combines privacy-preserving techniques such as differential privacy and homomorphic encryption [14], opens up new opportunities for on-device training, allowing AI models to evolve locally while keeping user data secure. Using differential privacy, we can introduce noise into datasets during training, ensuring that individual data points cannot be reverse-engineered or identified. At the same

Challenge	Impact	Potential Solution
Computational bottlenecks in resource-limited environments	Limits the ability to run complex AI models efficiently on edge devices	Hybrid AI architectures that offload heavy computations to the cloud when needed
Achieving low-latency inference for real-time applications	High latency degrades user experience in AR/VR, gaming, and live applications	Adaptive AI systems that dynamically adjust model complexity based on real-time conditions
Memory optimization for large models on edge devices	Large models quickly exhaust limited device memory and storage	Neurosymbolic AI to reduce reliance on large-scale neural networks
Energy efficiency and battery constraints in On-Device AI	AI models drain battery life, reducing device longevity	Dynamic Voltage and Frequency Scaling (DVFS) and energy-aware scheduling algorithms
Preserving model accuracy during compression and pruning	Compression techniques often lead to reduced model accuracy and quality	Federated learning 2.0 with techniques like differential privacy and distillation
Hardware compatibility and platform-specific optimization	Models optimized for one hardware platform may not work efficiently on others	Hardware-software co-optimization with custom AI-specific accelerators for edge devices
Maintaining real-time responsiveness across complex workloads	Delays in inference affect performance in applications requiring continuous processing	Speculative execution pipelines and multi-core processing optimizations
Scaling Large Language Models (LLMs) on mobile and edge devices	LLMs have a large memory footprint, creating performance delays	Better token pruning and model quantization to reduce memory usage without sacrificing performance
Securing data privacy without compromising performance	User data privacy is a concern when AI is processed locally	Privacy-preserving techniques like homomorphic encryption and differential privacy

Table 2: Comprehensive overview of related challenges, their impact on advancement of on-device GenAI, and potential solutions

time, homomorphic encryption techniques allow computations to be performed on encrypted data without needing to decrypt it first. This ensures that sensitive information remains secure throughout the training process.

4) Optimization Algorithms for Enhanced Mobile Performance Advancements in hardware-software co-optimization, such as the design of next-generation AI-specific hardware accelerators tailored to generative models, could help in further enhancing computational efficiency and open up possibilities for deploying even more sophisticated AI models on edge devices.

CONCLUSION

In the process of scaling down models, a discernible pattern emerges, characterized by a set of common steps applicable across various models. Notable techniques include employing the GELU activation function, implementing Group Normalization, incorporating Winograd convolutions, undertaking channel pruning, and leveraging Step Distillation methods. Particularly in models structured akin to U-Net, the optimization algorithm utilized in the SnapFusion model during U-Net training proves to be an effective starting point. Furthermore, crafting compatible models involves implementing optimizations akin to those discussed in SD-v2.1, especially when utilizing the TensorFlow framework, presenting a promising avenue for exploration. Despite these advancements, a significant limitation surfaces — the optimal performance of these techniques is often contingent on devices equipped with robust GPUs. With the release of the new Snapdragon 8 Gen 2 and 3 by Qualcomm, we can expect more powerful edge devices in the future, which will raise this challenge.

REFERENCES

1. V. T. Truong and L. B. Le, "Text-guided real-world-to-3d generative models with real-time rendering on mobile devices," in *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 1, no. 1, 2024,

pp. 1–6.

2. S. Saha, A. Hota, A. K. Chattopadhyay, A. Nag, and S. Nandi, “A multifaceted survey on privacy preservation of federated learning: progress, challenges, and opportunities,” *Artificial Intelligence Review*, vol. 57, no. 7, p. 184, Jun. 2024. [Online]. Available: https://doi.org/10.1007/s10462-024-10766-7
3. A. Awasthi and S. Sarawagi, “Continual learning with neural networks: A review,” in *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, ser. CODS-COMAD ’19, vol. 1, no. 1. New York, NY, USA: Association for Computing Machinery, 2019, p. 362–365. [Online]. Available: https://doi.org/10.1145/3297001.3297062
4. P. Kuswiradyo, B. Kar, and S.-H. Shen, “Optimizing the energy consumption in three-tier cloud–edge–fog federated systems with omnidirectional offloading,” *Computer Networks*, vol. 250, no. 1, p. 110578, 2024. [Online]. Available: https://doi.org/10.1016/j.comnet.2024.110578
5. Y. Li, H. Wang, Q. Jin, J. Hu, P. Chemerys, Y. Fu, Y. Wang, S. Tulyakov, and J. Ren, “Snapfusion: Text-to-image diffusion model on mobile devices within two seconds,” *Advances in Neural Information Processing Systems*, vol. 36, no. 3, p. 1, 2024.
6. Y.-H. Chen, R. Sarokin, J. Lee, J. Tang, C.-L. Chang, A. Kulik, and M. Grundmann, “Speed is all you need: On-device acceleration of large diffusion models via gpu-aware optimizations,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, vol. 1, no. 1, 2023, pp. 4650–4654.
7. N. P. Pandey, M. Fournarakis, C. Patel, and M. Nagel, “Softmax bias correction for quantized generative models,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, vol. 1, no. 1, 2023, pp. 1453–1458.
8. W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, “Federated learning in mobile edge networks: A comprehensive survey,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.
9. J. Hou and Z. Asghar, “World’s first on-device demonstration of stable diffusion on an android phone,” 2023. [Online]. Available: https://www.qualcomm.com/news/onq/2023/02/worlds-first-on-device-demonstration-of-stable-diffusion-on-android
10. J. Choi, M. Kim, D. Ahn, T. Kim, Y. Kim, D. Jo, H. Jeon, J.-J. Kim, and H. Kim, “Squeezing large-scale diffusion models for mobile,” *preprint*, vol. 1, no. 1, pp. 1–7, 2023.
11. B.-K. Kim, H.-K. Song, T. Castells, and S. Choi, “Bk-sdm: Architecturally compressed stable diffusion for efficient text-to-image generation,” in *Workshop on Efficient Systems for Foundation Models@ ICML2023*, vol. 1, no. 3, 2023.
12. D. Xu, W. Yin, X. Jin, Y. Zhang, S. Wei, M. Xu, and X. Liu, “Llmcad: Fast and scalable on-device large language model inference,” *preprint*, vol. 1, no. 1, p. 1, 2023.
13. A. Sheth, K. Roy, and M. Gaur, “Neurosymbolic artificial intelligence (why, what, and how),” *IEEE Intelligent Systems*, vol. 38, no. 03, pp. 56–62, may 2023.
14. R. Aziz, S. Banerjee, S. Bouzefrane, and T. Le Vinh, “Exploring homomorphic encryption and differential privacy techniques towards secure federated learning paradigm,” *Future Internet*, vol. 15, no. 9, p. 1, 2023. [Online]. Available: https://www.mdpi.com/1999-5903/15/9/310

Siva Sai is currently working toward the PhD degree in Blockchain & Machine Learning aided Healthcare at Birla Institute of Technology and Science-Pilani, Pilani, India. Contact him at p20220063@pilani.bits-pilani.ac.in.

Manish Prasad is currently working toward the undergraduate degree in Computer Science and Engineering at Vellore Institute of Technology Bhopal, Bhopal, India. Contact him at manishprsd563@gmail.com.

Garima Dashore is currently working toward the undergraduate degree in Electrical and Electronics Engineering at Birla Institute of Technology and Science-Pilani, Pilani, India. Contact her at f20201938@pilani.bits-pilani.ac.in.

Vinay Chamola is currently an Associate Professor with the Department of Electrical and Electronics Engineering, Birla Institute of Technology and Science-Pilani, Pilani, India. Contact him at vinay.chamola@pilani.bits-pilani.ac.in.

Biplab Sikdar is currently an Associate Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore Contact him at bsikdar@nus.edu.sg.