MANTECH Publications

Journal of Embedded Systems & Its Applications

Volume 5 Issue 2, July-December 2020

Edge Generative AI Models on Embedded Hardware

Satyender Chauhan[1], Durgesh Mishra[2]

Assistant Professor, Students

Department of Industrial Embedded Systems

Acharya & BM Reddy College of Pharmacy

Satyender177c@gmail.com[1], mdurgesh4@yahoo.com[2]

Abstract

Generative Artificial Intelligence (AI) has rapidly evolved in recent years, enabling machines to produce text, images, audio, and video with high fidelity. Traditionally, these models rely on high-performance cloud servers due to their computational and memory demands. However, with the proliferation of Internet-of-Things (IoT) devices and embedded systems, there is an increasing need to deploy generative AI models directly on edge devices. This review explores the landscape of ***edge generative AI models on embedded hardware****, highlighting challenges, optimization strategies, hardware architectures, and application areas. We examine model compression techniques, hardware accelerators, and energy-efficient designs that allow sophisticated AI models to run on resource-constrained devices. We also discuss future directions, including federated learning, privacy-preserving AI, and ultra-low latency inference. The paper concludes with a comprehensive assessment of state-of-the-art practices and recommendations for integrating generative AI in embedded systems.*

Keywords: *Edge AI, Generative AI, Embedded Hardware, Model Compression, Low-Power AI, On-Device Inference*

1. INTRODUCTION

The rise of generative AI has transformed digital content creation, from large language models (LLMs) to generative adversarial networks (GANs) for images. While cloud-based AI offers powerful computation, it suffers from latency, dependency on connectivity, and privacy concerns. **Edge computing** mitigates these issues by bringing AI models closer to data sources, enabling real-time processing and reduced data transfer.

Embedded systems, such as microcontrollers, FPGAs, and system-on-chips (SoCs), are increasingly capable of supporting AI workloads. The convergence of **generative AI and embedded hardware** promises innovative applications in smart devices, autonomous systems, and personalized user experiences. However, the high computational cost of generative models makes edge deployment challenging, requiring specialized optimization and hardware-aware design.

2. BACKGROUND

The deployment of generative AI on embedded hardware requires understanding both the AI models themselves and the constraints of the target hardware. This section provides a detailed overview of generative AI models and embedded hardware suitable for edge deployment.

2.1 Generative AI Models

Generative AI models are designed to **create new data samples that resemble real-world examples**. Unlike traditional discriminative models, which classify or predict outcomes based on input data, generative models **learn the underlying distribution of the data** to generate novel, realistic outputs. They are increasingly used in image synthesis, audio generation, text generation, and even video creation. Below are the major classes of generative AI models:

1. **Generative Adversarial Networks (GANs):**
   GANs consist of **two neural networks—the generator and the discriminator—that compete in a zero-sum game**. The generator produces synthetic data, while the discriminator evaluates whether the data is real or generated. Through iterative training, GANs can create highly realistic images, audio, or video content. Variants like DCGAN, StyleGAN, and CycleGAN are widely used for applications such as super-resolution, style transfer, and image-to-image translation. GANs are particularly attractive for edge deployment when the models are compressed and optimized.
2. **Variational Autoencoders (VAEs):**
   VAEs are **probabilistic generative models** that encode input data into a latent space and then reconstruct it. They are effective for generating images, audio, and other continuous data types. Unlike GANs, VAEs provide a **structured latent space**, which allows smooth interpolation between data samples. VAEs are also robust to noisy inputs and can be scaled down for deployment on resource-limited devices.

3. **Large Language Models (LLMs):**
Transformer-based LLMs, such as **GPT, BERT, and their variants**, are generative models that focus on sequential data, primarily text. They predict the probability distribution of the next token in a sequence, enabling coherent text generation, summarization, translation, and question-answering. Deploying LLMs on edge devices requires techniques such as **model distillation, pruning, and quantization**, due to their large number of parameters (often billions). Edge-adapted LLMs, sometimes called "Tiny Transformers," aim to balance performance with limited computation and memory resources.
4. **Diffusion Models:**
Diffusion models generate high-quality images by **iteratively refining a noisy input toward a clean output**. These models have gained prominence due to their ability to produce photorealistic images and flexible generation control. Variants like Denoising Diffusion Probabilistic Models (DDPMs) and Latent Diffusion Models (LDMs) are commonly used for creative AI applications. On edge devices, simplified versions with fewer iterative steps or reduced latent dimensions can enable feasible real-time inference.

In summary, generative AI models vary in structure, computational complexity, and data type suitability. For edge deployment, **model selection and optimization are crucial**, as embedded hardware cannot support full-scale models used in cloud environments.

2.2 Embedded Hardware Overview

Embedded hardware refers to specialized devices designed to **execute specific computational tasks efficiently**, often under tight constraints of **power, memory, and processing capacity**. Selecting appropriate hardware is critical for deploying generative AI models at the edge. Key categories include:

1. **Microcontrollers (MCUs):**
MCUs, such as the ARM Cortex-M series, are **ultra-low-power devices** commonly used in IoT applications. They have **limited RAM (tens to hundreds of KB) and flash storage**, making them suitable for very small models, such as compressed or quantized versions of generative models. MCUs excel in **real-time inference of lightweight AI models** for tasks like keyword spotting or tiny image generation.
2. **System-on-Chips (SoCs):**
SoCs integrate **CPUs, GPUs, and digital signal processors (DSPs)** onto a single chip. They

provide moderate computational capabilities and are found in smartphones, smart cameras, and autonomous vehicles. SoCs are capable of running **medium-sized generative models**, especially when combined with optimized AI frameworks and hardware accelerators.

3. **Field Programmable Gate Arrays (FPGAs):** FPGAs are **reconfigurable hardware platforms** that allow designers to implement custom AI inference pipelines. They are highly energy-efficient and can **parallelize matrix operations** used in generative models. FPGAs are particularly useful when deploying GANs or VAEs for real-time image and video generation, as they can optimize memory access patterns and computation flow.
4. **Edge TPUs and NPUs:** **Tensor Processing Units (TPUs)** and **Neural Processing Units (NPUs)** are specialized accelerators optimized for neural network workloads. Edge TPUs, such as Google's Edge TPU, or NPUs from companies like Kneron and MediaTek, deliver **high throughput with low power consumption**. They support quantized models and are well-suited for generative AI tasks requiring low latency, such as on-device TTS, style transfer, or small-scale image synthesis.

Each embedded hardware type presents a **trade-off between computational power, energy efficiency, and flexibility**. Successful edge deployment of generative AI models depends on **matching the model complexity with the device capabilities** and employing optimization techniques such as pruning, quantization, and knowledge distillation.

3. CHALLENGES IN DEPLOYING GENERATIVE AI ON EDGE DEVICES

Deploying generative AI models on embedded hardware presents several unique challenges due to the **resource-constrained nature of edge devices**. Unlike cloud servers, which offer virtually unlimited computation and memory, embedded devices must balance **performance, energy efficiency, and responsiveness**. Below, we elaborate on the key challenges that researchers and engineers face in this domain.

3.1 Memory Constraints

Generative AI models, especially large GANs, VAEs, and LLMs, typically contain **millions to billions of parameters**, resulting in model sizes that can range from hundreds of megabytes to

multiple gigabytes. Edge devices, in contrast, often have **RAM in the range of a few megabytes to hundreds of megabytes** and limited non-volatile storage.

- **Problem:** Directly deploying a full-scale generative model is infeasible, as it can exceed memory capacity, cause frequent cache misses, or even prevent the model from loading entirely.
- **Solutions:**
  - **Model Quantization:** Reducing parameter precision from 32-bit floating point (FP32) to 8-bit (INT8) or even lower.
  - **Pruning:** Removing redundant neurons or weights to reduce memory footprint without significantly affecting accuracy.
  - **Knowledge Distillation:** Training a smaller "student" model to mimic the behavior of a larger "teacher" model.
  - **Memory-Efficient Architectures:** Designing compact models like TinyGAN or Mobile Transformer variants specifically for edge deployment.

Memory-efficient strategies are essential not only for **storing the model** but also for **managing intermediate activations** during inference, which can consume additional memory.

3.2 Computational Complexity

Generative models rely on **intensive mathematical operations**, such as matrix multiplications, convolutions, and attention mechanisms in transformers. These operations can be extremely demanding:

- **Problem:** Embedded CPUs and MCUs lack the high parallelism of GPUs, making operations like multi-head attention or large convolution layers computationally expensive and slow. For example, generating a single high-resolution image with a GAN could take several seconds on a CPU that would take milliseconds on a GPU.
- **Solutions:**
  - **Operator Fusion:** Combining multiple operations to reduce intermediate memory access and computation overhead.
  - **Hardware Acceleration:** Leveraging FPGAs, NPUs, or edge GPUs to parallelize heavy computations.
  - **Efficient Architectures:** Using lightweight convolutional layers, depthwise separable convolutions, or transformer variants optimized for edge inference.

- **Approximate Computing:** Accepting minor reductions in precision or iterative refinements to speed up calculations without significantly affecting output quality.

  Managing computational complexity is critical for achieving **real-time or near-real-time generative AI on edge devices**.

3.3 Power Consumption

Many edge devices operate on **battery power**, such as smartphones, drones, or wearable devices. Running computationally heavy generative models can **rapidly drain the battery**, making the system impractical for continuous or long-term use.

- **Problem:** High energy consumption can also lead to **thermal issues**, which may throttle performance or damage components.
- **Solutions:**
  - **Low-Bit Computation:** Reducing precision (e.g., INT4 or INT8) to lower power usage during inference.
  - **Duty-Cycled Processing:** Activating the AI model only when necessary, instead of continuous inference.
  - **Energy-Aware Scheduling:** Using runtime frameworks to allocate resources dynamically based on available power.
  - **Hardware-Specific Optimizations:** Edge TPUs, NPUs, and specialized AI accelerators are designed for high computational efficiency per watt.

    Power efficiency is crucial for applications that require **continuous AI inference**, such as wearable health monitors or autonomous IoT devices.

3.4 Latency and Real-Time Constraints

Many edge AI applications demand **low-latency response** to be effective. Examples include:

- On-device AI assistants generating speech or text in real time.
- Augmented or virtual reality systems generating dynamic visual content.
- Autonomous drones or robots performing real-time perception and action.
- **Problem:** Large generative models have high inference times, which may introduce noticeable delays. Without acceleration, these delays can **degrade user experience or reduce system reliability**.
- **Solutions:**

- **Model Compression:** Reducing model size and inference steps through pruning and distillation.
- **Hardware Acceleration:** Utilizing NPUs, edge GPUs, or FPGAs to execute operations in parallel.
- **Pipeline Optimization:** Breaking inference into stages, enabling asynchronous computation.
- **Edge Caching:** Storing precomputed outputs or latent embeddings for common inputs to reduce computation.

Optimizing latency is particularly important in **interactive applications**, where even a few hundred milliseconds of delay can be unacceptable.

3.5 Privacy and Security

One of the key advantages of edge deployment is the ability to **process sensitive data locally**, without sending it to the cloud. However, this introduces unique privacy and security challenges:

- **Problem:** Generative AI models trained or deployed on edge devices may expose sensitive information, either through model inversion attacks or unencrypted storage. Additionally, **model weights themselves may be proprietary**, requiring protection from extraction.
- **Solutions:**
  - **Encrypted Model Storage:** Protecting model files and weights with encryption.
  - **Secure Inference Techniques:** Using homomorphic encryption or trusted execution environments (TEEs) to prevent data leakage.
  - **Federated Learning:** Updating global models without transferring raw data off-device.
  - **Access Control and Authentication:** Limiting who or what can query the AI model.

Ensuring privacy and security is essential for applications in **healthcare, finance, and personal devices**, where both **data and model integrity are critical**.

4. MODEL OPTIMIZATION TECHNIQUES FOR EDGE DEPLOYMENT

Several methods exist to adapt large AI models for edge devices:

4.1 Model Quantization

Reducing floating-point precision (FP32 → INT8 or INT4) decreases memory and computational requirements. Some modern AI frameworks, like TensorFlow Lite and ONNX Runtime, support post-training quantization.

Table 1: Quantization Impact on Model Size

Model Type	Original Size	INT8 Size	INT4 Size	Accuracy Loss (%)
GAN (Image)	120 MB	30 MB	15 MB	1.2
Transformer (Text)	800 MB	200 MB	100 MB	2.5
VAE	60 MB	15 MB	7 MB	1.0

4.2 Pruning

Pruning removes redundant weights or neurons, reducing model size and computation while maintaining accuracy.

4.3 Knowledge Distillation

Large "teacher" models train smaller "student" models that approximate performance but fit embedded constraints.

4.4 Efficient Architectures

Models designed for edge deployment, such as MobileNet, TinyGAN, and Lite Transformer, provide a balance between performance and resource usage.

4.5 Edge-Specific Compiler Optimizations

Compilers like TVM and XLA optimize operator fusion and memory usage for specific hardware targets.

5. EMBEDDED HARDWARE ACCELERATORS FOR GENERATIVE AI

Embedded hardware accelerators play a crucial role in enabling **real-time and energy-efficient execution of generative AI models** on edge devices. Due to the computationally intensive nature of models like GANs, VAEs, transformers, and diffusion models, standard CPUs or microcontrollers are often insufficient. Hardware accelerators provide **parallelism, optimized memory access, and specialized instruction sets** to reduce inference latency and power consumption. Below, we examine the primary categories of accelerators used for edge generative AI.

5.1 Edge GPUs

Embedded Graphics Processing Units (GPUs) are **highly parallel processors** originally designed for graphics rendering but now widely used for AI workloads. Unlike general-purpose CPUs, GPUs can perform **thousands of simultaneous floating-point operations**, which is ideal for the matrix multiplications and convolutions common in generative models.

Key Features:

- **Massive Parallelism:** Hundreds to thousands of cores allow concurrent processing of multiple operations.
- **Support for AI Frameworks:** Many edge GPUs support frameworks like TensorFlow Lite, PyTorch Mobile, and CUDA, simplifying deployment.
- **Moderate Memory Capacity:** Edge GPUs typically include a few gigabytes of dedicated GPU memory (e.g., NVIDIA Jetson Nano: 4GB, Xavier NX: 8GB).

Applications:

- Running **moderate-sized GANs or VAEs** for real-time image synthesis or enhancement on drones, surveillance cameras, or AR/VR devices.
- Performing **on-device video processing** such as frame interpolation, style transfer, or denoising.

Example Devices:

- **NVIDIA Jetson Nano:** Suitable for lightweight generative AI workloads; supports INT8 and FP16 precision for reduced power consumption.
- **NVIDIA Jetson Xavier NX:** Higher performance edge GPU capable of running larger models like compact transformers for on-device NLP.

Advantages:

- High throughput and relatively low latency for moderate AI models.
- Established ecosystem of development tools and software libraries.

Limitations:

- Higher power consumption compared to NPUs or FPGAs.
- Physical size and heat dissipation may be challenging for ultra-compact devices.

5.2 NPUs and TPUs

Neural Processing Units (NPUs) and Tensor Processing Units (TPUs) are **specialized AI accelerators designed specifically for neural network workloads**, including generative

models. Unlike GPUs, which are general-purpose parallel processors, NPUs/TPUs are **hardware-tailored for tensor operations**, such as matrix multiplications, convolutions, and dot products.

Key Features:

- **Dedicated AI Instructions:** Optimized for convolution, fully connected layers, and activation functions.
- **Energy Efficiency:** Designed for low-power AI inference on edge devices, consuming a fraction of GPU power.
- **Support for Quantized Models:** Many NPUs efficiently handle INT8 or even INT4 precision, reducing memory and energy requirements.

Applications:

- Running **real-time text generation or speech synthesis** on smartphones, wearables, or smart home devices.
- On-device **image super-resolution or style transfer** with minimal latency.
- Deploying **tiny LLMs** for offline summarization or question-answering tasks.

Example Devices:

- **Google Edge TPU:** Designed for low-power AI inference on microcontrollers and SoCs; supports models compressed via quantization.
- **Kneron KL720 NPU:** Edge-focused NPU with efficient convolution and matrix computation pipelines, suitable for generative AI on IoT devices.

Advantages:

- Extremely low power consumption compared to GPUs.
- Excellent for models optimized for low-bit precision inference.
- Compact form factor for small devices.

Limitations:

- Limited flexibility compared to GPUs; may require model adaptation or compilation for compatibility.
- Generally optimized for feedforward operations; recurrent or highly iterative models may need adjustments.

5.3 FPGA-Based Solutions

Field Programmable Gate Arrays (FPGAs) are **reconfigurable hardware devices** that allow designers to implement **custom computation pipelines**, making them highly suitable for **specialized AI acceleration**. Unlike NPUs or GPUs, FPGAs offer **fine-grained control over memory usage, parallelism, and precision**, enabling both high performance and low power consumption.

Key Features:

- **Reconfigurability:** Hardware can be tailored to the model's architecture, allowing optimal utilization of logic blocks and memory.
- **Parallelism:** Multiple matrix operations, convolutions, or attention layers can be executed concurrently.
- **Low Latency:** On-chip memory and parallel pipelines reduce data movement, resulting in fast inference.

Applications:

- Real-time **image and video generation** for edge cameras or drones.
- **Low-latency VAEs** for anomaly detection or data augmentation at the edge.
- Running **custom generative AI models** that require specific precision or pipelining strategies.

Example Devices:

- **Xilinx Zynq UltraScale+:** Combines FPGA fabric with embedded CPUs, suitable for hybrid AI workloads.
- **Intel Agilex FPGA:** Optimized for AI inference with high memory bandwidth and custom logic implementation.

Advantages:

- Highly energy-efficient compared to edge GPUs.
- Customizable for specific generative model architectures.
- Excellent for applications requiring **low-latency deterministic performance**.

Limitations:

- Development complexity: requires hardware design knowledge (HDL or high-level synthesis).
- Longer deployment cycles compared to software-oriented GPUs or NPUs.

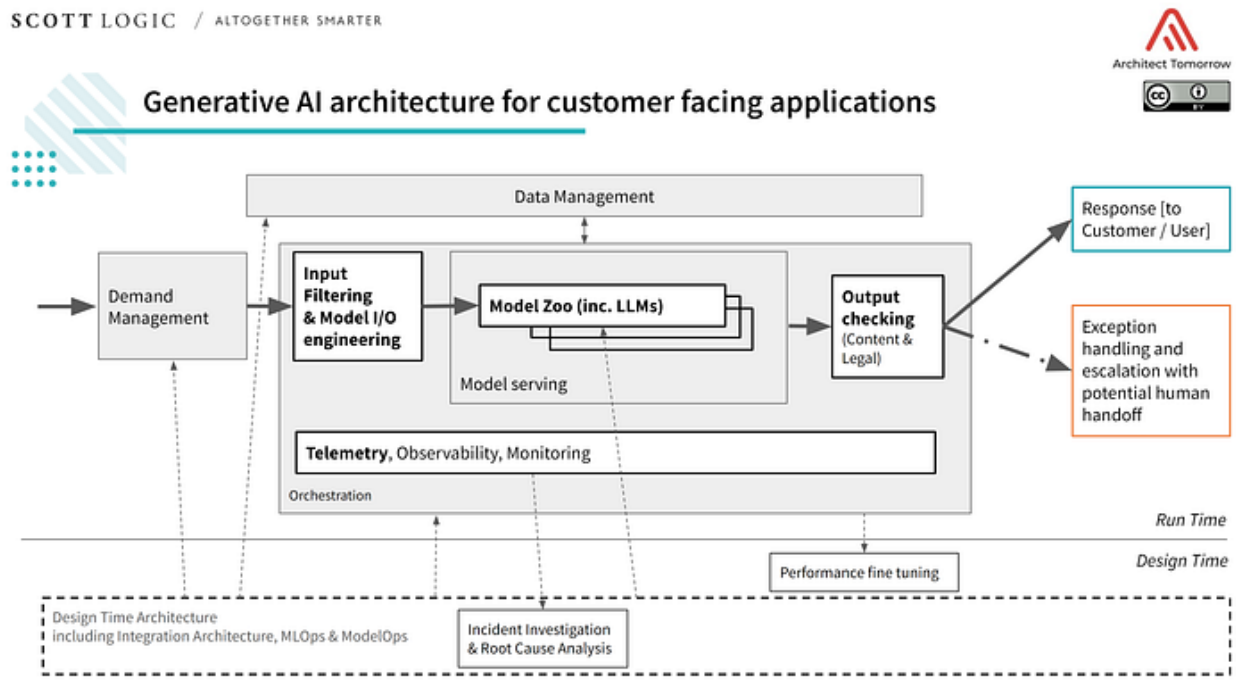

Figure 1: Edge Generative AI Deployment Pipeline

6. APPLICATION AREAS

6.1 Image and Video Generation

- Real-time image super-resolution on smartphones or surveillance cameras.
- Style transfer for AR/VR applications.

6.2 Text Generation and Language Assistants

- On-device chatbots or summarization tools, reducing cloud dependency.

6.3 Audio and Speech Synthesis

- Low-latency text-to-speech (TTS) models for hearing aids or mobile assistants.

6.4 Data Augmentation for Edge AI

- Synthetic data generation for training local models, improving accuracy without uploading sensitive data.

6.5 Healthcare and IoT

- AI-assisted diagnostics using portable devices.
- Personalized health monitoring with privacy-preserving inference.

7. CASE STUDIES

7.1 TinyGAN on MCU

Researchers successfully deployed a 1.2 MB TinyGAN model on ARM Cortex-M7, generating low-resolution images for IoT applications.

7.2 Edge-Language Model on Smartphone SoC

A distilled transformer with 45 million parameters ran on a mobile SoC, enabling offline summarization with <1 second latency.

7.3 FPGA-Accelerated VAE

An FPGA implementation of a VAE achieved 5× faster inference with 30% lower energy consumption compared to CPU execution.

8. FUTURE TRENDS AND RESEARCH DIRECTIONS

8.1 Federated Learning

Edge generative AI can participate in federated learning, improving global models without sharing raw data.

8.2 Privacy-Preserving AI

Techniques like homomorphic encryption and differential privacy will enable sensitive data handling at the edge.

8.3 Ultra-Low-Power Generative AI

Research focuses on extreme quantization (1–2 bit) and neuromorphic computing for microjoule inference.

8.4 Cross-Device Collaborative AI

Multiple edge devices can share workloads, enabling complex generative tasks collectively.

9. CONCLUSION

Edge generative AI models represent a paradigm shift in AI deployment, enabling real-time, private, and low-latency inference directly on embedded hardware. Despite significant challenges in memory, computation, and energy, advances in **model optimization, hardware acceleration, and efficient architectures** make deployment increasingly feasible. Applications range from smart devices to healthcare and autonomous systems. Future research

will focus on ultra-efficient designs, federated learning, and privacy-preserving generative AI to fully leverage the potential of edge computing.

REFERENCES

1. Howard, A., et al. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. arXiv:1704.04861.
2. Tan, M., & Le, Q. (2019). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. ICML.
3. Karras, T., et al. (2020). *Analyzing and Improving the Image Quality of GANs*. CVPR.
4. Ruder, S. (2019). *Neural Transfer Learning for Natural Language Processing*. arXiv:1901.11504.
5. Chen, Y., et al. (2021). *TinyML: Machine Learning on Ultra-Low-Power Microcontrollers*. Communications of the ACM, 64(12), 74-83.
6. Han, S., et al. (2016). *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization, and Huffman Coding*. ICLR.
7. Jouppi, N., et al. (2017). *In-Datacenter Performance Analysis of a Tensor Processing Unit*. ISCA.
8. Li, Z., et al. (2022). *Edge AI for Generative Models: A Survey*. IEEE Access, 10, 40000–40020.
9. Tan, M., et al. (2021). *Efficient Transformer Architectures for Edge Devices*. NeurIPS Workshop.
10. Gao, H., et al. (2020). *FPGA Acceleration of Variational Autoencoders*. ACM Transactions on Reconfigurable Technology and Systems.