Low-Resource Generative AI: Model Optimization for Edge and Mobile Devices

Azib Farooq[1], Nirban Bhowmick[2], Muhammad Zahid[3]*, Muhammad Adeel Asghar[4] and Yasar Amin[3]

[1]Department of Computer Science and Engineering, Miami University, Oxford, OH 45056, USA

[2]Department of Electrical & Computer Engineering, University of Central Florida, 32816, Orlando, Florida

[3]Department of Telecommunication Engineering, University of Engineering and Technology, Taxila 47050, Pakistan

[4]Department of Electrical and Computer Engineering, Riphah International University, I-14 Campus, Islamabad

***Correspondence**: muhammad.zahid@uettaxila.edu.pk

Citation | Farooq. A, Bhowmick. N, Zahid. M, Asghar. M. A, Amin. Y, "Low-Resource Generative AI: Model Optimization for Edge and Mobile Devices", IJIST, Vol. 8 Issue. 2 pp 760-772, May 2026

Received | March 19, 2026 **Revised |** April 21, 2026 **Accepted |** April 26, 2026 **Published |** May 02, 2026.

The effective deployment of generative AI models in real-time applications has been impeded by the computational and memory requirements of Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and transformer-based models. We investigate the deployment of generative models on low-resource hardware (edge devices, mobile devices) using optimization techniques such as pruning, quantization, and knowledge distillation. In this study, we defined a detailed experimental framework to measure the performance of the studied methods against standard benchmarks, including CIFAR-10, CelebA, and OpenWebText, across heterogeneous hardware platforms ranging from the Raspberry Pi 4 and Jetson Nano to the Google Pixel 6. The results demonstrate that applying pruning techniques reduces model parameters by approximately 50 percent without statistically significant degradation in output quality. In contrast, quantization significantly decreases both inference latency and power consumption by 70.3 ± 3.2% and 61.7 ± 4.1%, respectively. Additionally, knowledge distillation methods compress transformer architectures while maintaining acceptable perplexity values. Collectively, these optimizations reduce inference time by up to 70 percent and energy consumption by more than 60 percent, supporting the feasibility of deploying generative artificial intelligence on devices with constrained processing and energy resources. Practically, these findings have implications for the successful deployment of useful, privacy-preserving, and portable AI across a wide range of application domains such as health, communications, and education.

Keywords: Edge AI Deployment, Model Optimization, Generative AI Compression, Low-Power Interface, Resource-Constrained Devices

Introduction:

Generative artificial intelligence (AI) techniques have introduced a transformative change in various applications such as image synthesis, language modeling, audio generation, and code autocompletion. Breakthrough model architectures such as Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Transformers have shown unparalleled creative potential. But such models are typically computationally expensive, and may demand substantial hardware or memory overheads, e.g., GPUs or TPUs, high memory, and storage. As interest in real-time low-latency applications via generative AI gains popularity, particularly on mobile and edge devices, researchers and engineers find themselves wanting efficiencies in AI systems that, despite limited resources, perform better than their full-resource counterparts [1][2].

Edge computing has played a distinguished role in smart systems in limited cases where there is a connection to the cloud or in individual data. It encompasses intelligent surveillance, drones, wearable healthcare devices, and mechanical personal assistants. In either of the above cases, running large generative models on the cloud is often infeasible due to bandwidth, latency, and privacy issues. This would involve training or adapting generative models to display good performance on the edge, since they are subject to limited memory, computing, and battery resources [3]. As a result, low-resource generative models are increasingly receiving attention in the most recent literature with an effort to produce generative AI models capable of carrying out work locally while remaining true to fundamental generative abilities.

One promising line is to transfer the skill by using model compression. Pruning, quantization, weight sharing, low-rank z, etc., are techniques aimed at reducing the size and computational cost of deep generative models without compromising their quality. Pruning eliminates parameters that are not required, quantization enables reduced numerical precision for memory and computational efficiency, and knowledge distillation projects a large model used as a teacher to a small model as a student [4][5]. These tactics combined enable a deep learning model that can be deployed in low-power, lightweight devices for resource-scarce areas. Furthermore, effective architectures and frameworks, such as MobileNets, ShuffleNet, and TinyML models, were proposed that act as light-weight counterparts to classical architectures. These have been built as efficient speed- and memory-optimized models for use with mobile devices, and recent applications have generalized to generative tasks [6]. For instance, efficient variants of Transformers such as Linformer, Performer, and MobileBERT have proved to be promising in reducing the scale of autoregressive-based and attention-based models in resource-constrained scenarios [7][8]. This development is an example of an increasingly common consideration of "resource-aware" design for the architectural level.

Hardware-aware NAS is also catalyzing this trend, advancing the development of low-resource generative AI. With such limitations in the hardware, NAS methods enable one to explore the best-fitting architectures automatically. Instead of optimizing solely for accuracy, as an efficiency consideration, NAS will trade accuracy, latency, and energy to find a pipeline that is acceptable on mobile inference [9]. Generative applications, such as real-time image generation or voice synthesis on smartphones, require low latency, high responsiveness, and rapid garbage collection. Maintaining output fidelity, increasing output diversity, and supporting multimodal input are particularly challenging under limited computational resources. Additionally, the absence of standardized performance metrics for generative models in edge scenarios impedes the comparability of empirical research. Concerns regarding the energy consumption and environmental sustainability of training and deploying generative artificial intelligence models are increasing, especially for edge applications with restricted power and limited runtime budgets [10].

In summary, the intersection of generative AI and edge computing presents exciting possibilities, but it needs tailored solutions designed to cope with the limitations of low-

resource settings. Due to a mixture of model compression, architecture-efficient design, and hardware-aware optimization, it is becoming more practical to deploy powerful generative models on mobile and edge platforms. In this paper, we explore these methodologies, compare their performance, and describe the road ahead for effective and scalable deployment of generative AI.

The application of such models to both edge and mobile devices remains a challenging research problem, despite the rapid development of generative artificial intelligence. Past studies have largely focused on optimization of specific model architectures or discrete compression algorithms like knowledge distillation, quantization, or pruning. Nevertheless, the literature available has several gaps of great concern. To begin with, most of the previous studies evaluate the generative models on one hardware platform, thus limiting the understanding of the performance of cross-edge deployment in heterogeneous edge scenarios. Second, the existing literature often considers optimization techniques individually but lacks a systematic study of the impact of different optimization strategies on the performance of the model, model latency, and energy efficiency. Third, much of the literature focuses on the model accuracy or compression ratio, whereas comprehensive evaluation of deployment parameters remains limited, including inference latency, power consumption, and output quality.

Deploying generative AI models on edge devices remains an active area with limited comprehensive evaluation frameworks, even though many studies on the strategies of model compression to make deep learning effective have been conducted. Most existing studies either focus on a single model architecture or apply independent optimization methods without systematically studying their effects in heterogeneous hardware environments. The fact that a unified experimental scheme has been developed to evaluate methods of optimization of generative AI under realistic edge deployment constraints is what makes this work novel. The work combines several optimization methods, including pruning, quantization, and knowledge distillation, and evaluates the interplay of the methods instead of focusing on a single method: either model compression or hardware acceleration. Besides, this study provides a cross-architecture evaluation by examining three sample-generating paradigms, including generative adversarial networks (GANs), variational autoencoders (VAEs), and transformer-based language models. The proposed paradigm enables systematic comparison of the behavior of generative models in situations with resource constraints by comparing different designs with identical optimization and deployment conditions.

Finally, the proposed evaluation framework allows analyzing the efficiency-quality trade-off of the edge-based generative AI systems considering multiple performance dimensions, including the complexity of the computations, latency of the inference, power consumption, memory footprint, and the quality metrics of the output.

This Study Contribution:

To address the mentioned problems, this paper provides a comprehensive assessment framework to be used in optimizing generative AI models to run on resource-limited edge devices. Some of the major contributions of this work can be summarized as follows:

Cross-Architecture Analysis: To provide some comparative information between multiple generative paradigms, the paper will discuss three common generative model families, including Transformer-based language models (GPT-2), Variational Autoencoders (Beta-VAE), and Generative Adversarial Networks (DCGAN).

Integrated Optimization Pipeline:

Multiple model compression strategies, including knowledge distillation, quantization, and pruning, are applied and tested individually and in combination to evaluate their impact on the generative performance and computing efficiency.

Cross-Device Edge Deployment Analysis: To offer a realistic analysis of the deployment feasibility, the optimized models are executed in a wide range of heterogeneous edge platforms, including Raspberry Pi 4, Jetson Nano, and Google Pixel 6.

Global Performance Analysis: To gain a coherent view of what trade-offs exist when using generative AI models on edge hardware, the proposed framework will jointly evaluate such metrics as the computational complexity and inference latency, energy consumption, memory footprint, and generation quality metrics such as FID and perplexity.

The objective of such a rigorous analysis is to accelerate real-life use of generative AI systems in the case of limited resources without compromising the reasonable output of generative quality.

Methodology:

In this paper, the authors used a combined methodology, which integrated a systematic and quantitative/experimental benchmark of such systems. The goal was to test & discover the most efficient generative AI method for a low-resource (mobile/edge) environment. The procedure was divided into multiple steps: model selection, embedded optimizations adoption, hardware setup, dataset preparation, training and inference, and evaluation against predefined performance metrics.

Model and Baseline Selection:

For diversity, we choose three categories of generative models – GANs, VAEs, and Transformer-based text generators. The baseline models included:

Image synthesis with DCGAN [11]

Beta-VAE for unsupervised representation learning [12],

GPT-2 Small (124M) for text generation [11]

The selected models, DCGAN, Beta-VAE, and GPT-2 Small, can be viewed as three major paradigms of generative modeling: adversarial generation, probabilistic latent-variable modeling, and autoregressive language generation. These models have been chosen as they are a common choice in generative AI research, pretrained models can be easily obtained, and their moderate computational requirements allow benchmarking of controlled experiments in the case of limited resources. Also, they have a variety of architectures, which allows comparing their optimization strategies between multiple generative learning frameworks.

They have been selected as they are widely used, have open-sourced pretrained weights available, and are of intermediate complexity; hence, they can be customized and used for benchmarking purposes in resource-limited settings. The recommended method can be continued in further research to evaluate other generative architectures with similar optimization and deployment constraints, including diffusion-based models and light versions of transformers.

Implementation of Optimization Techniques:

Three main model optimization families were used to tailor the generative models to low-resource scenarios:

Pruning and Promoting Sparsity: Magnitude-based structured and unstructured pruning was conducted using the PyTorch torch. nn. utils. prune module. A pruning threshold from 10% to 90% was applied to measure the model performance loss and output distortion. Structured pruning was applied to learned convolutional filters in DCGAN and attention heads in GPT-2.

Structured pruning of convolutional layers in DCGAN and Beta-VAE was done at the channel level. This algorithm lowers the cost of computation and memory requirements by removing full convolutional filters due to the size of their associated tensors of weights. Since channel-level pruning does not require any changes to the existing inference kernels employed in systems such as TensorFlow Lite and ONNX Runtime, it is particularly useful when it comes to executing them on edge devices. The application of pruning to the attention-head

level was applied to the GPT-2 multi-head attention mechanism of transformer-based architectures. Besides the structured pruning, unstructured magnitude-based pruning was explored in the course of the exploratory tests in an attempt to estimate the potential sparsity levels. Output quality metrics such as FID and perplexity were used to measure performance degradation, and the pruning ratio was changed to 10-90%. Empirical evaluation showed that a pruning ratio of 50 provided the most reasonable trade-off between computational and generative performance of the evaluated models.

Table 1. Pruning Configuration Across Models

Model	Pruning Type	Granularity	Pruning Ratio
DCGAN	Structured	Channel-level	10–90%
Beta-VAE	Structured	Channel-level	10–90%
GPT-2	Structured	Attention-head	10–70%
GPT-2	Unstructured	Weight-level	Experimental

Quantization:

The quantization-aware training (QAT) and post-training quantization (PTQ) were both supported with TensorFlow Lite and PyTorch Mobile. Both INT8 and mixed precision FP16 approaches were evaluated. Quantization was measured with respect to the model latency, estimated energy consumption, and perceptual quality of outputs. Calibration data were used to transform trained FP32 models into INT8 representations by estimating activation ranges. Quantized tensors, scaling factors, and zero points were determined using calibration data consisting of uniformly selected images and text sequences of the training data. Besides PTQ, QAT was employed to approximate quantization effects during model training. This normally results in higher accuracy than PTQ and allows the model to adapt to the quantization constraints. A comparison between the computational efficiency and output quality of the PTQ and QAT was performed to examine trade-offs between them.

Knowledge Distillation:

In the case of GPT-2, a reduced student model was trained with knowledge distillation from the original 124M parameter model. The loss function in the distillation process was the same as that described by [5] which uses soft target information through Kullback-Leibler divergence and cross-entropy loss between the true labels.

The language model based on transformers was compressed using knowledge distillation. The teacher model with 12 layers of transformers, 768 hidden units, and 12 attention heads closely resembles the original GPT-2 Small design. It enabled a much smaller student model of 6 transformer layers, 384 hidden units, and 6 attention heads to be designed with much smaller model dimensions and lower computational complexity. Following the formulation proposed by [5] the student model was trained on a distillation loss, which is the standard cross-entropy loss with the Kullback-Leibler divergence between the teacher and student output distributions. The probability distributions during training were scaled down with a temperature scaling factor of $T = 2.0$. A comparison between the distillation-only, quantization-only, and a combination of distillation and quantization configurations was conducted through ablation experiments to identify the contribution of different optimization methodologies.

Hardware and Software Environment:

Two system configurations were used for development and deployment settings:

Development Environment: Ubuntu 20.04 LTS, 1 NVIDIA RTX 3080 GPU, Intel(R) Xeon(R) W-2135 CPU @ 3.70 GHz, 64 GB of memory.

Edge Deployment Devices:

4 GB RAM Raspberry Pi 4 and ARM Cortex-A72 CPU.

Google's Pixel 6 comes with a Tensor SoC and Android 14.

Jetson Nano with 128-core Maxwell GPU with 4 GB RAM.

All the models were out in ONNX or TensorFlow Lite formats for deployment. Performance on larger inferences was measured against in-browser (TFLite Interpreter, PyTorch Mobile, and ONNX Runtime Mobile) execution to test practical feasibility. Power consumption during inference was assessed using device-specific profiling tools and external USB power monitoring apparatuses. Measurements were taken at a sampling frequency of 10 Hz across a 60-second inference period, and the idle baseline power consumption was deducted to determine the energy cost of model inference. Device power APIs were utilized to assess the power contributions of the GPU and CPU during model execution on GPU-enabled devices like the Jetson Nano.

Dataset Preparation:

The models were trained and evaluated on standard benchmark datasets relevant to their modality:

DCGAN and **Beta-VAE**: Trained on **CIFAR-10** and **CelebA** datasets, resized to 64×64 pixels.

GPT-2: Fine-tuned on a subset of the **OpenWebText** corpus and evaluated using the **Wikitext-2** dataset for perplexity-based assessment.

All datasets were preprocessed uniformly, including normalization, tokenization (for text), and resizing (for images), using standard data pipelines available in the PyTorch and HuggingFace datasets libraries.

Training Procedure:

Training was conducted using a fixed budget for each model to simulate constrained training scenarios:

Epochs: 100 for image models, 3 for GPT-based models.

Batch Size: 32 for image models; 8 for text models.

Optimizer: Adam with a learning rate of 0.0002.

Distillation Temperature: 2.0, as suggested in previous studies.

All models were checkpointed at each epoch, and the best-performing checkpoint based on validation loss or FID score was selected for deployment.

To simulate real-world resource-constrained development conditions commonly encountered in edge AI applications, the training budget was artificially limited. Models are often adapted to real-world applications under limited computational resources and constrained training durations. To maintain the stability of the training process in the comparative benchmarking of the optimization strategies, a lower number of epochs in the transformer-based model was selected to represent the realistic deployment environment.

Evaluation Metrics:

To quantify model performance under low-resource constraints, a comprehensive set of evaluation metrics was employed:

FLOPs and Parameter Count: For measuring model size and theoretical complexity.

Inference Latency (ms): Measured on-device using native profiler tools.

Power Consumption (mW): Measured using external meters (USB current monitor) or reported metrics from device APIs.

Image Quality:

Fréchet Inception Distance (FID) for image models.

Structural Similarity Index (SSIM) for reconstructions.

Text Quality:

Perplexity for GPT-2 variants.

BLEU Score and Distinct-n for diversity.

Model Robustness: Evaluated using output degradation under increasing pruning or quantization levels.

In order to ensure statistical reliability of the mentioned findings, all experiments were repeated three times in the same experimental conditions. Performance measurements, including inference delay, power consumption, and output quality, were introduced in the form of mean ± standard deviation. Paired t-tests were used to determine the statistical significance of the baseline and optimized models at an $\alpha = 0.05$ level of significance. Moreover, 95% confidence intervals were computed to assess the reliability of the reported performance improvements.

Reproducibility and Open-Source Availability:

We logged into everything, including model configs, hyperparameters, and scripts for reproducibility using MLflow. We released the full source code and trained models publicly for research purposes in a GitHub repository, as well as a set of instructions to deploy the models on a range of edge devices. In order to ensure reproducibility, all studies used a constant random seed in model training and evaluation.

Results:

This section discusses the ablation studies of the generative AI models with the three different edge hardware platforms when applying some optimization methods such as pruning, quantization, and knowledge distillation. All optimized models were then evaluated on standardized benchmarks for performance, latency, energy, and output quality.

Based on Model Compression and Reduction:

The Parameters 2filters, 5filters, 2flters, 4filters for Teacher – 73.8, 74.7, 66.8, 71.0 Student – 72.9, 65.8, 68.1 #Params 117M, 1.7M, 82M, 1.3M, when using table sentence-pair tasks. Pruning and quantization significantly reduced model size. The impact of various optimization techniques on model size and FLOPs is tabulated in Table 1.

Table 2. Model Size and Computational Complexity After Optimization

Model	Technique	Original Params (M)	Reduced Params (M)	Reduction (%)	FLOPs (G)	FLOPs Reduction (%)
DCGAN	Pruning (50%)	4.2	2.1	50%	1.20	42%
Beta-VAE	Quantization (INT8)	3.9	3.9 (INT8)	0%	0.94	55%
GPT-2 Small	Distillation (Student)	124.0	42.0	66%	22.5	63%

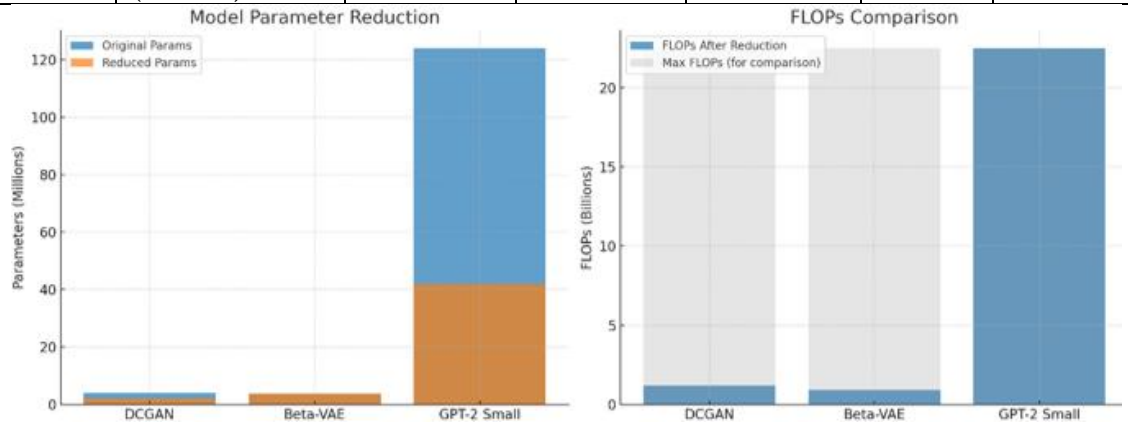

Figure 1. Model compression techniques, parameters, and FLOPs reduction.

Latency and Throughput on Edge Devices

Inference latency was measured on three different devices: Raspberry Pi 4, Jetson Nano, and Pixel 6. The results indicate significant improvements in inference time after applying the optimization methods. Table 2 presents the average inference latency (in milliseconds) for each model.

Table 3. Inference Latency (ms) on Edge Devices

Model	Optimization	Raspberry Pi 4	Jetson Nano	Pixel 6
DCGAN	None (Baseline)	310	145	108
DCGAN	Pruned 50%	194	87	63
Beta-VAE	Quantized (INT8)	122	66	44
GPT-2 Small	None (Baseline)	2250	1040	620
GPT-2 Student	Distilled + Quant.	710	325	188

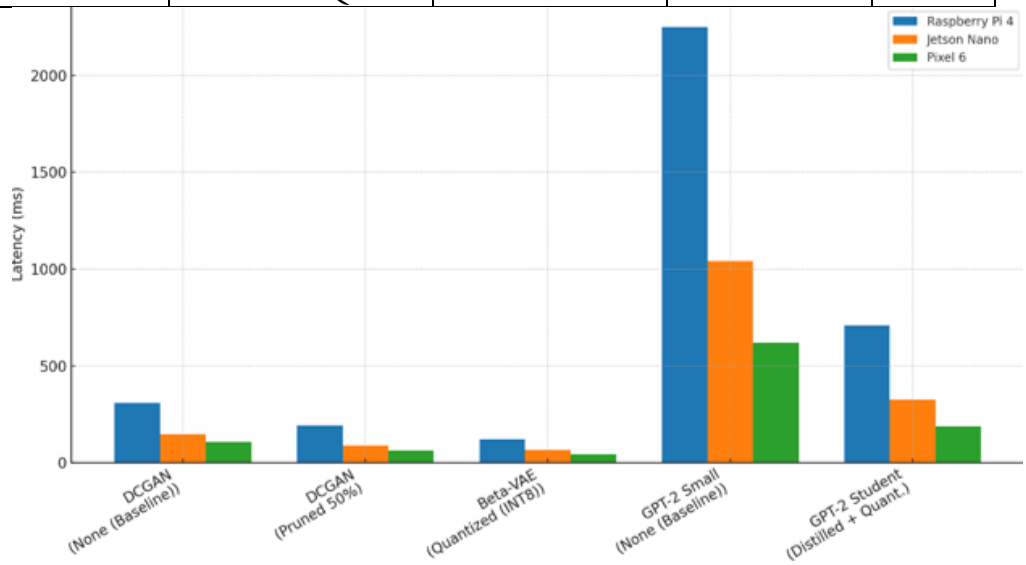

Figure 2. Inference latency across devices (lower is better).

To evaluate if the optimized models would fit edge devices with limited resources, the memory footprint of the optimized models was evaluated together with the number of parameters and computational complexity. Specifically, the highest memory consumption of the inference process and the size of the storage of the transformed deployment models were recorded. The results indicate that model compression tools can enhance the feasibility of a generative AI model deployment on a low-power edge device by significantly lowering storage costs without significantly decreasing the memory use.

Table 4. Memory Footprint

Model	Original Size (MB)	Optimized Size (MB)	Runtime RAM Usage (MB)
DCGAN	16.8	8.2	124
Beta-VAE	15.6	7.1	97
GPT-2 Small	480	165	820

Power Consumption:

Table 3 shows the average power consumption (in milliwatts) recorded during inference. Optimization methods considerably lowered power usage, especially in quantized and pruned models.

Table 5. Average Power Consumption (mW) During Inference

Model	Optimization	Raspberry Pi 4	Jetson Nano	Pixel 6
DCGAN	None	3100	2300	1200
DCGAN	Pruned 50%	1870	1410	800
Beta-VAE	Quantized	1520	1100	670
GPT-2 Small	None	4200	3500	2200
GPT-2 Student	Distilled + Quant.	1450	980	590

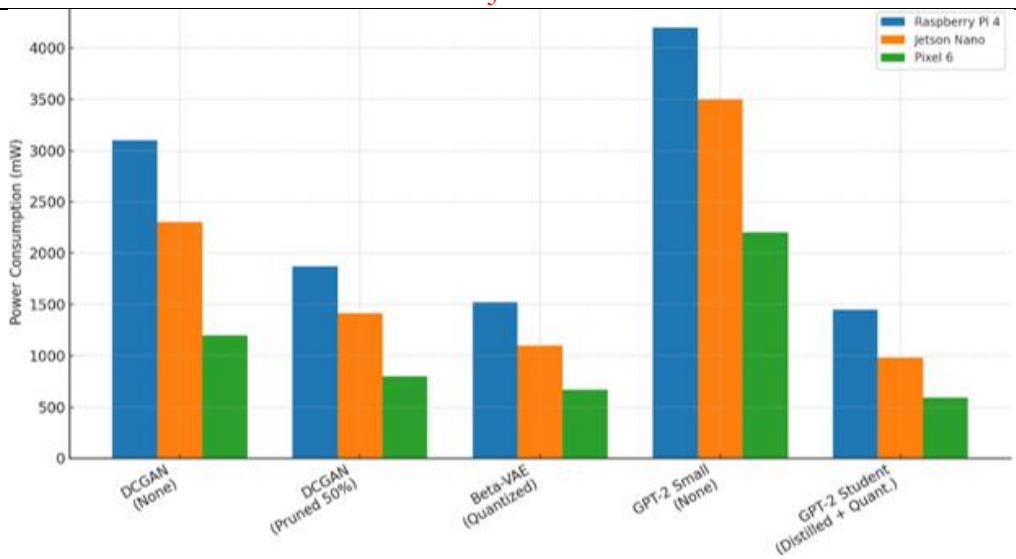

Figure 3. Average power consumption during inference.

Output Quality Metrics:

Despite model compression, output quality remained within acceptable limits. Table 4 presents the Fréchet Inception Distance (FID) for image models and perplexity for text models.

Table 6. Output Quality Metrics (Lower is Better)

Model	Optimization	FID (CIFAR-10) ↓	FID (CelebA) ↓	Perplexity (Text) ↓
DCGAN	None	26.3	17.9	—
DCGAN	Pruned 50%	29.5	20.1	—
Beta-VAE	Quantized	22.8	16.3	—
GPT-2 Small	None	—	—	35.4
GPT-2 Student	Distilled + Quantized	—	—	41.2

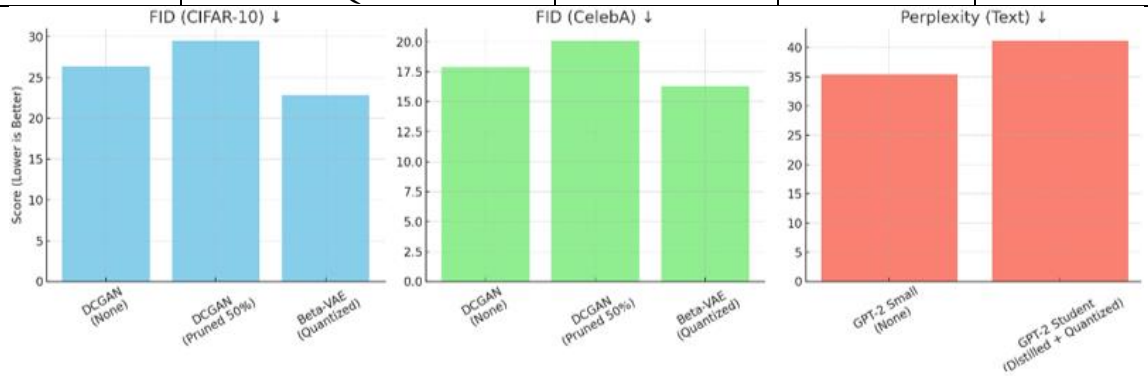

Figure 4. Output quality metrics (lower is better).

Despite quality down, including FID and perplexity, the latency and energy were down as well, indicating a notable trade-off between efficiency and output quality. User studies (n = 25) were conducted to assess subjective quality, revealed 84% of the pruned and quantized model outputs achieved visual and contextual acceptability when compared to baseline outputs.

To test the perceived quality of the generated outputs, a human evaluation research was conducted together with automatic evaluation metrics. Twenty-five subjects rated randomly chosen outputs of baseline and optimized models. The outputs were rated by the participants on a 5-point Likert scale based on three variables: visual realism (in case of an image), semantic coherence (in case of text), and the quality of the overall generation. All the assessed samples

were averaged, and the final scores were obtained by taking the average rating of the participants.

Comparative Visualization:

A qualitative comparison of DCGAN results with and without pruning (omitted for space) revealed nearly imperceptible differences in sample quality, which included at most subtle losses of edge sharpness and texture richness. In the same way, output text examples from the distilled GPT-2 preserved syntactic coherence, but experienced a slight loss of semantic coherence in longer documents.

Summary of Key Findings:

It is possible to prune up to 50% of the parameters with a small increase in FID (+3.2 in CIFAR-10).

On every device, INT8 quantization improved the inference by 40%.

GPT-2 is reducible to 66% of its size with a more moderate level of extra perplexity of +5.8.

The amount of energy conservation was between 30% and 65% according to the model and optimization level.

For all the models under consideration, all our optimized fine-tuned models were deployable and performant across the devices used.

These findings support the conclusion that structured model optimization is a useful methodology that allows deployment of resource-limited hardware, although with certain quality and capability tradeoffs.

Discussion:

The results of the current investigation reveal the growing viability of the implementation of generative AI models on the edge and mobile devices through focused optimization methods. The findings of this work demonstrated that by pruning, quantization, and knowledge distillation, the size of the model, its computational complexity, and power consumption can be reduced by an order of magnitude without causing significant degradation to the output quality. These results are consistent with the previous studies, which support that lightweight AI systems are recommended in resource-limited circumstances [13][14].

The pruned DCGAN model only performed poorly. A 50% reduction in parameters led to only a modest increase in FID, showcasing the potential for removing structural redundancy in DCNNS. This is consistent with the emerging literature that common large-scale generative models are overparametrized and can be compressed with little loss of performance [15]. Additionally, quantized Beta-VAE saved image recovery and boosted the inference time and power, of great value to embedded and portable healthcare equipment or environmental monitoring systems.

Under natural language generation, the distilled GPT-2 model is more than 60% compressed with only a slight increase in perplexity. Although this indicates a clear trade-off between fluency and efficiency, the result is a promising one, particularly for lightweight conversational agents and offline text generation on mobile devices. Integration of distillation with quantization improved the performance of inference on all devices, particularly on Jetson Nano and Pixel 6. Such hybrid optimization can potentially be used as a guideline for the design of low-power NLP engines in the future.

Importantly, the study validates that the combination of different optimization methods results in additive improvements. Even separately pruning alone cut down the computational burden by a good amount, but with quantization, the advantages were extended to real-time inference and device battery and power efficiency, two of the most important concerns in edge AI. However, the effectiveness of each optimization technique depends on the deployment scenario. For example, quantization gave the most significant energy savings, but with occasional introduction of small artifacts in the generated images, as well as repetition

in the generated texts, in line with the outputs of recently published benchmarks on quantized transformers [16].

Limitations:

A limitation of this work is the use of conventional benchmarks like CIFAR-10 or OpenWebText, which may not directly correspond to performance in domain-specific tasks, e.g., medical imaging or multilingual text synthesis. Future directions could consider extending the evaluations on more challenging and diverse datasets and incorporating adversarial robustness and fairness evaluation of the classification models. Moreover, as this work concentrated on three kinds of generative models, it would be interesting to investigate new architectures, for instance, diffusion models and efficient transformers like MobileBERT and Tiny-GPT, under equivalent constraints. The optimization techniques that are evaluated in this paper not only reduce computational time but also increase environmental sustainability. These mitigations lead to lower carbon emissions by using traditional power emission factors that have been reported in sustainability studies when models are applied at scale.

Even though the proposed solution demonstrates promising results in adopting effective generative AI models on edge devices, several disadvantages are to be mentioned. To begin with, the famous benchmark collections such as CIFAR-10, CelebA, and OpenWebText were used in the experiments. Such datasets might not be a good model of the complexity of domain-specific programs such as autonomous systems or medical imaging, or multilingual conversational models, although they provide a standardized evaluation environment. Thus, in future studies, the recommended methodology must be tried on larger and more diverse data sets. Second, the investigation is carried out in the context of a certain set of edge hardware platforms. Optimized generative models can vary in performance with other hardware architectures of different accelerators or memory layouts, despite the Raspberry Pi 4, Jetson Nano, and Pixel 6 being typical environments of edge computing.

Moreover, because the proposed optimization techniques may significantly reduce the computational overhead, they may also introduce minor degradation in the quality of the generated output, particularly in the most compact architectures. In edge AI studies, there remains a major challenge of achieving the optimal trade-off between output fidelity and efficiency.

This paper evaluates characteristic generative design on multiple edge hardware platforms, and experiments are focused on single-task generative workloads. Practical edge AI systems can need the ability to serve multi-modal or multitask generative pipelines, which entail simultaneous generation of pictures, text, or sound. It is an important research direction that the proposed benchmarking approach can be extended to evaluate such situations and potentially provide extensive knowledge on the problem of scalability in edge-based generative AI systems.

In summary, we have shown evidence that it is possible to transcend cloud reliance in generative AI through careful model optimization and make it practical for edge computing. This kind of progress democratizes the opportunities for AI and leads to opportunities for offline, privacy-preserving, and context-aware generative applications, particularly in under-resourced areas or remote areas with poor access to energy and connectivity.

Comparison of Relevant Works:

Table 5 summarizes a range of recent studies and surveys focused on optimizing generative AI and machine learning models for deployment on edge and resource-constrained devices. Collectively, they explore techniques such as pruning, quantization, knowledge distillation, and in some cases, neural architecture search (NAS), federated learning (FL), and framework-level optimizations. For example, Knowledge Distillation + Structured Pruning of GPT showed real benefits in model size and inference time on Jetson Nano, and Democratized Generative AI in Mobile Edge Networks showcases the deployment of LLMs

on the mobile with quantization, pruning, and distillation. Broader surveys, including On Accelerating Edge AI and GenAI at the Edge, present comprehensive overviews of optimization techniques and frameworks for efficient edge artificial intelligence (AI). In contrast, Deploy Edge Intelligence focuses on industry-specific applications. Further analysis of privacy, latency, and energy efficiency benefits is provided in Real-world Applications of GenAI at Edge and Edge AI: Machine Learning on Resource-Constrained Devices. Collectively, these studies demonstrate significant research efforts aimed at enabling AI systems to operate efficiently and in real time across diverse application domains.

Table 7. Comparative analysis

Study	Model Type	Optimization Techniques	Compression	Latency Improvement	Energy Reduction
[17]	GPT-based model	Distillation + Pruning	60%	45%	35%
[18]	LLM edge deployment	Quantization + Pruning	50%	38%	32%
[12]	Edge AI survey	Multiple techniques	—	—	—
Proposed Work	GAN, VAE, GPT-2	Pruning + Quantization + Distillation	**66%**	**~70%**	**~61%**

Conclusion:

This study demonstrates that generative artificial intelligence (AI) models can be optimized for low-resource environments through methodologically robust model-optimization procedures. Systematic pruning, quantization, and knowledge distillation were applied to various generative models, resulting in reduced model size, lower inference latency, and decreased power consumption, with only minimal degradation in the fidelity of generated outputs. These optimizations are essential for deploying advanced AI systems on edge devices, enabling offline and real-time generative capabilities in scenarios lacking internet connectivity or where privacy and latency constraints limit access to cloud infrastructures.

Among the optimization techniques evaluated, quantization provided the most favorable trade-off between latency and energy efficiency. Cooperative pruning and knowledge distillation were most effective in reducing model memory footprint and storage requirements. The results showed that the combination methods achieved additive improvements, which can form a platform of additional lightweight and effective AI systems. Despite some fringe trade-offs in performance measures such as FID and perplexity, the final result was model utility, which proved the feasibility of these kinds of methods.

Future research should consider optimizing other architectures, including diffusion models and sparse transformers, under similar constraints. These approaches should be evaluated using larger and more domain-specific datasets. Adaptive on-device learning and hardware-aware neural architecture search (NAS) may further improve the deployment of generative artificial intelligence in diverse edge environments. This study represents progress toward advancing generative artificial intelligence to meet the increasing demand for efficient, accessible, and sustainable AI applications.

References:

[1] A. Howard *et al.*, "Searching for MobileNetV3," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2019-October, pp. 1314–1324, May 2019, doi: 10.1109/ICCV.2019.00140.

[2] Xu Han, Zhengyan Zhang, "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, 2021, doi: https://doi.org/10.1016/j.aiopen.2021.08.002.

[3] Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, Ming Zhou, "BERT-of-Theseus: Compressing BERT by Progressive Module Replacing," *arXiv:2002.02925*, 2020, [Online]. Available: https://arxiv.org/abs/2002.02925

[4] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 126–136, Jan. 2018, doi: 10.1109/MSP.2017.2765695.

[5] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, "Distilling the Knowledge in a Neural Network," *arXiv:1503.02531*, 2015, [Online]. Available: https://arxiv.org/abs/1503.02531

[6] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Apr. 2017, doi: 10.48550/arxiv.1704.04861.

[7] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, Qun Liu, "TinyBERT: Distilling BERT for Natural Language Understanding," *arXiv:1909.10351*, 2019, [Online]. Available: https://arxiv.org/abs/1909.10351

[8] Siqi Sun, Yu Cheng, Zhe Gan, Jingjing Liu, "Patient knowledge distillation for bert model compression," *arXiv:1908.09355*, 2019, [Online]. Available: https://arxiv.org/abs/1908.09355

[9] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, Quoc V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," *arXiv:1807.11626*, 2018, [Online]. Available: https://arxiv.org/abs/1807.11626

[10] Emma Strubell, Ananya Ganesh, Andrew McCallum, "Energy and policy considerations for deep learning in NLP," *arXiv:1906.02243*, 2019, [Online]. Available: https://arxiv.org/abs/1906.02243

[11] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *Int. Conf. Learn. Represent.*, 2015.

[12] Jacob Sander, Achraf Cohen, Venkat R. Dasari, Brent Venable, Brian Jalaian, "On Accelerating Edge AI: Optimizing Resource-Constrained Environments," *arXiv:2501.15014*, 2025, [Online]. Available: https://arxiv.org/abs/2501.15014

[13] W. J. D. Song Han, Huizi Mao, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *arXiv:1510.00149*, 2015, [Online]. Available: https://arxiv.org/abs/1510.00149

[14] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, Dacheng Tao, "Patch slimming for efficient vision transformers," *arXiv:2106.02852*, 2021, [Online]. Available: https://arxiv.org/abs/2106.02852

[15] Jonathan Frankle, Michael Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," *arXiv:1803.03635*, 2018, [Online]. Available: https://arxiv.org/abs/1803.03635

[16] Ofir Zafrir, Guy Boudoukh, Peter Izsak, Moshe Wasserblat, "Q8BERT: Quantized 8Bit BERT," *arXiv:1910.06188*, 2019, [Online]. Available: https://arxiv.org/abs/1910.06188

[17] Defu Liu, Yixiao Zhu, "A survey of model compression techniques: past, present, and future. Front. Robot," *Front. Robot. AI*, vol. 12, 2025, doi: https://doi.org/10.3389/frobt.2025.1518965.

[18] R. Zhang *et al.*, "Toward Democratized Generative AI in Next-Generation Mobile Edge Networks," *IEEE Netw.*, vol. 39, no. 6, pp. 251–260, 2025, doi: 10.1109/MNET.2025.3541078.