

A Case Study of Environmental Footprints for Generative AI Inference: Cloud versus Edge

Pengfei Li
University of California,
Riverside
pli081@ucr.edu

Mohammad J. Islam
University of California,
Riverside
misla056@ucr.edu

Shaolei Ren
University of California,
Riverside
shaolei@ucr.edu

ABSTRACT

The rapid growth of generative AI has placed significant strain on traditional data center infrastructures and existing power grids, leading to soaring energy demands and environmental burdens that may disproportionately affect the local communities. Shifting AI inference from the cloud to edge devices could potentially reduce the reliance on network connections, enhance user privacy, and alleviate the escalating pressure data centers impose on the local electricity grid. In this work, we present a case study examining the environmental footprint and energy consumption when deploying a generative AI model on cloud and edge platforms. To this end, we model and evaluate the water consumption and carbon emissions associated with AI inference across these deployment scenarios. Our empirical results demonstrate that, for several state-of-the-art generative AI models deployable on both cloud and edge devices, a reduced environmental footprint is observed for edge platform deployments. More specifically, edge platforms can achieve over 90% energy savings while reducing carbon emissions and water consumption by more than 80%. Putting the accuracy and latency performance aside, these findings highlight the potential of edge inference to lower the energy demands and environmental footprint of generative AI compared to cloud-based inference.

1. INTRODUCTION

The proliferation of generative AI has accelerated significantly in recent years, driven by the immense potential across diverse fields, including healthcare, education, drug discovery, and climate modeling. Many generative AI models, particularly those with expansive parameter sets, are typically trained and deployed in energy-intensive data centers. In 2022, data centers consumed roughly 2% of global electricity [1], contributing approximately 1% of global greenhouse gas emissions [2]. Furthermore, the AI market and energy consumption is projected to expand tenfold within the next five years [3], driven by the escalating demand for generative AI. This surge will pose significant challenges to existing data center infrastructure and energy systems. Not only the physical space hinders data center capacity expansion, limitations imposed by power capacity are also particularly acute, especially in regions with large data center concentrations

such as Virginia [4, 5]. Industry leaders have mirrored this trend by calling for an imperative breakthrough in power systems to meet the immense energy requirements of the burgeoning AI industry [6].

Simultaneously, the expanding processing capabilities of edge devices have recently spurred a potential shift in computational paradigms, transitioning from centralized data centers to decentralized edge devices such as smartphones. Offloading the AI inference from the cloud to edge devices helps to minimize the effects of unreliable network connections [7], which is essential to improve the users' quality of experience. Moreover, by storing both the model weights and users inputs locally, the on-device inference can better safeguard user privacy and avoid the leakage of sensitive information. In addition, the energy consumption of edge computing is more distributed than cloud computing, as the cloud computation platforms are typically hosted in warehouse-level data centers. Considering the escalating demand of AI computing, such concentrated energy pressure of datacenter would not only drain the local electricity capacity, but could also lead to disproportionate social and environmental costs [8]. For instance, the concentrated electricity consumption required to power data centers in Virginia can contribute to higher transmission congestion costs, carbon emissions and/or hazardous solid wastes [9, 5]. Additionally, water evaporation needed for electricity generation and data center cooling [10] can exacerbate freshwater scarcity in certain regions.

Quantifying the environmental footprint of such AI model inferences is a non-trivial task, requiring distinct considerations for both cloud and edge platforms. For carbon emissions, we not only account for emissions generated during electricity production but also utilize life cycle analysis (LCA) tools to evaluate embodied carbon. A significant difference exists between the carbon emissions associated with manufacturing and the nominal lifespans of cloud servers versus mobile devices, resulting in varying embodied carbon impacts for cloud and edge platforms. Similarly, water consumption during AI inference arises partly from electricity generation on both cloud and edge platforms. However, cloud platforms also contribute to additional water consumption through onsite evaporation[11, 12] required for server cooling.

That being said, leveraging the centralized infrastructure, cloud platforms excel at scalability and parallel data processing. Besides, these powerful cloud servers can handle extensive computational workloads and support most advanced AI models. On the other hand, edge platforms, which

deploy AI models closer to the data source—such as on mobile devices, IoT devices, or local servers—often attain better energy efficiency. By minimizing the need for data transmission to distant cloud servers, edge computing also reduces latency and energy consumption, thereby lowering the environmental impact of generative AI inference.

Recent studies have demonstrated that edge platforms could be more environmentally sustainable than cloud-based solutions, particularly when running some machine learning (ML) models. For instance, research [13, 14] highlights the substantial reduction in carbon footprint achieved by deploying AI on edge devices. These findings highlight the potential of edge computing to mitigate the environmental costs associated with AI, especially as the demand for generative AI continues to grow. By leveraging localized processing, edge platforms not only enhance energy efficiency but also contribute to a more sustainable AI ecosystem, aligning with global efforts to reduce carbon emissions and promote greener technologies.

Our empirical findings provide additional insights by considering a concrete case study in environmental footprint evaluation between cloud and edge platforms, which includes both water consumption and carbon emissions. Our methodology helps understand the environmental benefits of edge inference over their cloud-based alternatives. Specifically, we compare the energy usage and environmental footprint of several popular up-to-date generative models running on both cloud servers with high-end GPUs (e.g., Nvidia A100) and edge devices (e.g., Samsung S24). Using identical model inputs and architectures, our results suggest that edge inference can potentially achieve a 90% energy saving and an 80% environmental footprint reduction compared to cloud inference for generative AI.

2. METHODOLOGY OVERVIEW

We evaluate the inference time, energy consumption, and environmental footprints of several popular models on both cloud and edge devices. In our main experiment, we compare these metrics for several generative AI models deployed on cloud servers with either Nvidia A100 or L4 GPUs (hosted on Google Colab) and the Samsung S24 edge device. Additional measurements for several popular large language models running on the Google Pixel 8 are provided in Appendix A.

Profiling Metrics: In order to estimate the environmental footprints, we profile generative AI model performance with a focus on two key metrics: inference time and energy consumption. The inference time, or end-to-end latency, is the elapsed time between a user submitting a request and receiving the generative model’s response, excluding model loading and data preprocessing. Similarly, energy consumption is measured during this inference process. The software kits for measuring inference and energy consumption on cloud and edge are detailed in Section 3.2.

Environmental Footprint: Based on the measurements, we estimate the environmental footprint, including carbon footprint and water consumption incurred by generative AI model inference. Apart from the operational carbon emission during the model inference, we further incorporate the embodied missions during manufacturing, transport, and recycling. Our analysis is based on the widely used Life Cycle Analysis (LCA) tools (e.g. [15, 16]), following the practices of many large companies. More specifically, the carbon footprint of deploying a generative model i on device

d is given by

$$C_{(i,d)} = \gamma_c \cdot E_{(i,d)} + \frac{O_d}{U_d \cdot L_d} \cdot T_{(i,d)} \quad (1)$$

where $E_{(i,d)}$ and $T_{(i,d)}$ are the energy consumption (including applicable overheads such as data center cooling and mobile phone charging loss) and inference time associated with model i on device d , γ_c denotes the average carbon intensity of electricity generation, and O_d , L_d , and U_d denote the life-cycle carbon emission, the nominal lifetime and utilization rate of device d , respectively. The average carbon intensity γ_c is determined by the average regional fuel mix of electricity generation. The two terms in Eqn. (1) denotes the operational carbon emission and amortized embodied carbon during the inference period, respectively

In the cloud computing environment, the energy consumption, denoted as $E_{(i,d)}$, includes not only the energy used by IT equipment for computational tasks, but also the energy consumed by non-IT infrastructure, such as cooling systems, power distribution units, and other supporting facilities. This comprehensive energy consumption is typically calculated by incorporating the Power Usage Effectiveness (PUE), represented as δ_d , which serves as a multiplier to account for the energy overhead associated with non-IT equipment. The PUE metric is a critical factor in evaluating the energy efficiency of data centers, as it quantifies the ratio of total facility energy to the energy consumed solely by IT devices. A lower PUE value indicates a more energy-efficient data center, as less energy is wasted on non-computational activities.

On the other hand, when considering edge devices, such as smartphones, IoT devices, or local servers, the energy consumption model also account for additional factors beyond just the operational energy for a consistent comparison. One significant aspect is the energy overhead incurred during the battery charging process. For instance, energy losses occur during the conversion of alternating current (AC) from the power grid to direct current (DC) required by the device’s battery. These losses, often referred to as charging inefficiencies, can substantially impact the overall energy footprint of edge devices. Additionally, factors such as battery degradation over time, thermal management, and idle power consumption further contribute to the total energy expenditure.

By incorporating these considerations into the energy consumption models for both cloud and edge environments, researchers and practitioners can gain a more accurate understanding of the energy dynamics involved in deploying AI and other computational workloads. This holistic approach enables the identification of opportunities for optimizing energy efficiency, reducing environmental impact, and making informed decisions about the trade-offs between cloud and edge computing platforms.

Due to the lack of reliable and comprehensive data sources for conducting a life cycle analysis (LCA) of embodied water consumption across the supply chain, this study narrows the the water footprint assessment of both cloud and edge platforms to operational water consumption. Operational water consumption primarily refers to the water utilized during the active use phase of AI systems, such as the cooling of data centers and the operation of servers, which are critical for maintaining optimal performance and preventing overheating in both cloud and edge computing environments. To sum up, the methodology of the operational water footprint is similar

to the operational carbon footprint, by simply replacing the carbon intensity γ_c with water usage efficiency γ_w .

3. EXPERIMENT SETUP

This section first describes the experimental platform and profiling tools used in our empirical study. Subsequently, we detail the model and dataset employed for generative model inference. Finally, we present the latency profiling methodology for generative AI models on cloud and edge platforms.

3.1 Experiment Platform

We choose Google Colab as the cloud environment, which offers affordable access to a range of server configurations, spanning from servers equipped with NVIDIA’s A100 GPUs to those featuring the highly efficient L4 GPUs. Specifically, our profiling analysis on Google Colab utilizes two distinct configurations: the first features one NVIDIA A100 GPU (with 40 GB GPU RAM), 83.5 GB system RAM, and 201.2 GB allocated disk space, while the second employs one NVIDIA L4 GPU (with 22.5 GB GPU RAM), 53.0 GB system RAM, and 201.2 GB allocated disk space. This setup allows us to compare performance across different GPU architectures within the Colab environment.

For the on-device analysis, we select devices based on their capacity to deploy generative models, considering the resource constraints of edge devices. Concretely, our primary experiment is conducted with Samsung Galaxy S24, equipped with the Qualcomm Snapdragon 8 Gen 3 chipset. The chipset is the most advanced Snapdragon mobile platform [17] at the time of these experiments. This configuration ensures compatibility with a diverse set of generative models. The inference time of the generative models are mainly profiled using the Neural Processing Unit (NPU) [18] within the Qualcomm chipset, which is specially designed for accelerating the AI workloads. The measurement of NPU-based inference is implemented on the cloud-hosted edge device platform, Qualcomm AI hub, supporting the deployment of generative models on NPUs. To provide a more comprehensive comparison, we also benchmark the performance of on-device AI model deployment on GPUs of several mobile devices, including the devices (e.g., Google Pixel 8). Google Pixel 8, released in the same year as the Samsung Galaxy S24, provides a set of additional data points for our analysis. The detailed profiling of model performance on the Google Pixel 8 can be found in Appendix A, further validating our comparison of environmental footprints between cloud and edge devices.

3.2 Profiling Tools

Compared with edge devices, cloud servers often have better tools and support to deploy generative AI models and measure their latency and energy in an end-to-end manner. Unless specified otherwise, all the models profiled on the cloud are downloaded from Huggingface and executed with their official transformer API [19]. The inference time and consumed energy for each prompt is collected using the context manager provided by CodeCarbon [20], which records the elapsed time solely for the model inference code segment and measures the energy usage of the server components (e.g. CPU, GPU, RAM) during this period.

On the other hand, measuring such latency and energy consumption for generative AI models deployed on edge

devices presents more challenges, due to the resource constraints and limited Software Development Kits (SDK). For instance, Google’s MediaPipe Solution [21], while adept at cross-platform deployment, currently supports a limited subset of large language models. For Samsung Galaxy S24, it is challenging to deploy models with more than 2 billion parameters and run inference on NPUs through MediaPipe. Qualcomm’s toolchain [22], though supporting Llama-2 inference with 7 billion parameters on NPUs, focuses more on devices with recent Qualcomm chipsets at the current stage. Thus, in the main experiment, we select Qualcomm’s toolchain to evaluate energy consumption and latency using AI-optimized NPUs, as detailed in Section 4.1. Cross-chipset comparisons across various LLMs are presented in Appendix A.

3.3 Generative AI Models

We choose several popular generative AI models with various sizes and model architectures. These models are deployable both on cloud servers and edge devices. We begin with TrOCR, a light-weight transformer-based optical character recognition (OCR) model for reference, which generates character sequence from the input image [23]. Next, we experiment with the diffusion models, a family of text-to-image generative AI models. More specifically, we consider the following four models, Stable Diffusion V2.1 [24], Stable Diffusion V1.5 [25], Riffusion [26], ControlNet [27], each with distinct architectural designs, application focuses, and training techniques. For text-to-text generative models, we deploy and profile the latency of Llama-2-7B, one of the established open-source large language models, both on-cloud and on-device [28]. We choose the 7B version as it achieves a good tradeoff between memory requirements and model performance.

When deploying generative AI models in the cloud, further quantization is typically avoided to preserve optimal model performance, especially for models with small and medium sizes [29, 30]. However, the limited resources of edge devices (e.g. RAM), often necessitate additional modifications to adapt these models to hardware constraints. For example, the Qualcomm AI Hub[31] employs a hybrid 4-bit and 8-bit quantization approach for the Llama-2-7B model. The quantized model is optimized for Qualcomm chipset neural processing units (NPUs), and employs sequential loading of sharded models to address memory constraints. These optimizations, while designed to suit specific SoCs, do not always guarantee reduced latency or energy consumption on the AI servers hosted on cloud. Therefore, we evaluate the energy consumption and environmental footprint of state-of-the-practice representative deployment strategies across both cloud and edge platforms.

3.3.1 Llama-2-7B

Llama-2 [32] is a transformer-based auto regressive language model developed by Meta AI which is the updated version of Llama-1 [33] inheriting most of the settings from it. However, the key difference is in the maximum context length which is double (4k) compared to the context length of Llama-1[33]. The model is pre-trained using AdamW optimizer on 2 trillion tokens with a learning rate of 3.0×10^{-4} . Further, the Llama-2 model is trained using Nvidia A100 GPUs for 184320 GPU hours. Llama-2-7B is one of the versions of Llama-2 [32] with 7B parameters.

We use the checkpoint of Llama-2-7B from Huggingface [34]

for the Google Colab experiment which is quantized to either 4 or 8-bit and the on-device experiment uses Qualcomm’s special adaptation of Llama-2-7B from [35] which is quantized in a hybrid manner (Part of it quantized to 4-bit and the rest is 8-bit). During the inference, the maximum length of the token sequence generated by these models are capped by 1,024 in our experiment.

Due to resource constraints (e.g. RAM), the edge deployable version of Llama-2-7B model [35], is split into eight components which are loaded and executed sequentially. Among these eight components, the first four processes the prompt and generate the first output token, which are known as prompt processors. The latter four components are denoted as token generators, which generate one token per iteration until it reaches max token length. Such partitioning ensures efficient on-device inference and resource utilization.

3.3.2 Latent Diffusion Model

We consider three variants of latent diffusion models, which stem from the same base architecture [36]. These diffusion models have three main components: the Text-Encoder, U-Net, and the Variational Autoencoder (VAE). The details about the architecture of each diffusion model are outlined as below.

Stable Diffusion v2.1: A latent diffusion model derived from the base latent diffusion model described in [36], which achieves state-of-the-art performance on image generation. For our experiments on Google Colab, we utilized the Stable Diffusion v2.1 model available on Hugging Face [24], which is fine-tuned from the weights of Stable Diffusion v2 [37]. The on-device version of Stable Diffusion v2.1 [38] is adapted by Qualcomm-AI-Hub and has 340M parameters for the Text-Encoder, 865M parameters for the U-Net, and 83M parameters for the VAE Decoder. The model takes text prompts as input, which are processed by the text-encoder OpenCLIP-ViT/H [39].

Stable Diffusion v1.5: Another latent diffusion model [36] with the same number of parameters in the U-Net and VAE model, as Stable Diffusion v2.1 [38]. However, these weights are initialize differently and fine-tuned on images with a resolution of 768×768 . Another key difference of the Stable Diffusion V1.5 is that it uses CLIP ViT-L/14[40] as the text-encoder to process the input prompts

Riffusion: Derived from the base model [36], Riffusion is fine-tuned from Stable Diffusion v1.5 [25]. It employs the CLIP ViT-L/14 [40] as the text encoder and maintains the same number of parameters as the previous diffusion models. However, this model’s use case is different it generates a spectrogram from the input prompts.

3.3.3 ControlNet

ControlNet [41] is a neural network model designed to augment diffusion models with additional conditional control. It enables the incorporation of spatially localized input conditions into a pre-trained diffusion model through fine-tuning. ControlNet consists of four key components: Text-Encoder, U-Net, ControlNet, and VAE. Text prompts, along with an image, serve as inputs to the model, which then generates an output image. ControlNet employs CLIP ViT-L/14 [40] as the text encoder. In our Google Colab experiment, the baseline model of ControlNet is the Stable Diffusion Model V1.5 [25], augmented with Canny edges. For the on-device experiment, we use Qualcomm’s adapted version [42], which

has 340M parameters for the Text-Encoder, 865M parameters for the U-Net, 83M parameters for the VAE Decoder, and 361M parameters for ControlNet.

3.3.4 Text Recognition Model

TrOCR [43] is a transformer-based character/text recognition model designed to take text images as input and convert them into equivalent text. TrOCR utilizes an image transformer as an encoder that processes an image input (with a resolution of 384×384 as per [43]) by splitting the image into sequence patches (16×16). This sequence of patches generates embeddings that the decoder transformer uses to produce the output tokens. TrOCR has various variants, with the smallest base model containing 62M parameters, encompassing both the encoder and decoder. This model was trained with a learning rate of 5×10^{-5} on 40 V100 GPUs, each with 32 GB of memory.

Qualcomm optimizes this model [44] for the on-device deployment, where the encoder consists of 23M parameters and the decoder consists of 38.3M parameters. The input resolution for the on-device model is 320×320 .

3.4 Dataset

Our experiment encompasses three distinct types of tasks: text generation, image generation, and optical character recognition. For each task, we select one widely used dataset for the inference of generative models. For the text generation task, we use the `vicgalle/alpaca-gpt4` dataset from the Hugging Face Hub [45]. This dataset contains 52,000 entries, each with four columns. We combine the `instruction` and `input` columns to prepare the prompts. For the image generation task, we utilize the `poloclub/diffusiondb-Text` dataset [46]. This dataset has several subsets, so we select one subset with 1,000 entries, employing the `prompt` column as input for inference. Finally, for the task optical character recognition (OCR), we use the `corto-ai/handwritten-text` dataset [47]. This dataset contains 2,920 entries, each with handwritten text images in one column, which we use as input for inference.

For the inference of each task, we sample 50 prompts from the corresponding dataset. To ensure the consistency in performance comparison, the same set of samples is used as input for the generative model hosted both on-cloud and on-device. Additionally, the sampled prompts or images are processed without batching operations for a fair comparison between the cloud and edge devices.

3.5 Latency Profiling Methodology

In addition to the model architecture, some hyperparameters of these generative models also play a key role in the latency measurement. For instance, the number of diffusion steps directly impacts the trade-off between the overall latency and generation quality. To ensure reproducibility, we explicitly state key hyperparameters: diffusion steps were consistently set to 50 for all models across cloud and edge deployments. The maximum token length for all LLMs was 1024. Our empirical results for Llama-2-7B, with an average token length of approximately 373, confirm the validity of this setting. With the latency estimates detailed as below, the energy consumption of each edge device is derived by multiplying its measured average power (e.g. 6.22 Watt for Samsung S24).

Profiling such end-to-end latency for generative models

deployed on cloud is straightforward using standard tools. In contrast, profiling NPU-deployed models presents a challenge due to the lack of reliable software. We address this by profiling the latency of each component separately on Qualcomm AI Hub[31], then obtain the total latency according to the usage of each component. In this section, we delve into the detailed composition of the chosen generative AI models and how to profile the individual components.

3.5.1 Llama-2-7B

The Llama-2-7B model processed by Qualcomm is quantized and sharded into several components, to facilitate the deployment in edge devices. More concretely, suppose the prompt and token generator of Llama model are divided to N_p and N_g parts, we denote the i -th prompt generator and j -th token generator as p_i and g_j , respectively. The profiled latencies of sharded model p_i and g_j , obtained from Qualcomm AI Hub, are T_{p_i} and T_{g_j} respectively. To sum up, the overall latency is given by

$$T = \sum_{i=1}^{N_p} T_{p_i} + (N_t - 1) \cdot \sum_{j=1}^{N_g} T_{g_j} \quad (2)$$

where N_t is the number of tokens generated by Llama-2 model, depending on the input prompt. As the token length differs with different prompt inputs, the actual latency of Llama-2 also shows a significant variance. In our experiment, the number of generated tokens was determined by deploying the same model on GPUs with identical input prompts. Both the prompt and token generators were sharded into four parts ($N_p = 4$, $N_g = 4$).

3.5.2 Latent Diffusion Models

The selected diffusion models introduced in Section 3.3.2 share a similar model architecture, composed of text encoder, the single-step diffusion model (e.g. U-Net or its variants) and the variational autoencoder (VAE). For simplicity, all their overall latency are denoted as below

$$T = T_{TE} + T_{VAE} + N_{df} \cdot T_{U-Net} \quad (3)$$

where N_{df} is the number of diffusion steps, T_{U-Net} represents the single-step diffusion time, T_{TE} denotes the text encoder's inference time, and T_{VAE} denotes the inference time of VAE.

3.5.3 ControlNet

Similar to diffusion models, ControlNet employs a shared encoder-diffusion-decoder architecture. However, the distinguishing feature of ControlNet is its augmented diffusion setup, incorporating an additional module, denoted as C , to process contextual inputs. Consequently, the overall latency of the ControlNet module is given by

$$T = T_{TE} + T_{VAE} + N_{df} \cdot (T_C + T_{U-Net}) \quad (4)$$

where T_C denotes the inference time of module C , and the other parts are kept the same as the original diffusion model.

3.5.4 Text Recognition Model

The transformer based model for optical character recognition, TrOCR, is split into two components for on-device deployment. Specifically, the encoder and decoder of the transformer were loaded sequentially during inference. As there's no need to iterate over these components, the overall

latency can be calculated by

$$T = T_{En} + T_{De} \quad (5)$$

where T_{En} and T_{De} denote the latency of the transformer's encoder and decoder, respectively.

4. EMPIRICAL RESULTS

In this section, we provide a summary of our results, suggesting that edge inference can potentially reduce the energy consumption and environmental footprints compared to the cloud counterpart.

4.1 Inference Time and Energy

For the generative models deployed on Google Colab, we measure their latency and energy consumption in an end-to-end measure using CodeCarbon [20]. Then we augment the measured energy with PUE $\delta = 1.10$ to take non-IT equipment energy into consideration. Such PUE factor matches the fleet-wide average annual PUE in Google, according to its 2024 ESG report [12]. For edge devices, we measure the end-to-end latency by comparing the difference between the system timestamp before and after executing generative models on CPUs or GPUs. In our experiment, we first fully charge the phone battery, then execute generative model inferences, and finally measure the energy consumed to recharge the battery back to its original state. By accounting for the charging efficiency, the overall energy consumption of generative models on edge devices considers the energy overhead for the charger's AC-DC conversions. According to the Level-VI efficiency regulation in the U.S. [48] and the self-reporting charging efficiency by Samsung [49], we choose the charging efficiency of a wired charger as 0.8. On the NPUs, such end-to-end energy and latency measurement is technically challenging due to the lack of open-source solutions to readily deploy large language models on NPUs. Instead, we profile the latency of each component of generative models using Qualcomm AI hub, then derive the overall latency according to the usage of each component, as detailed in Section 3.5. The results are summarized as Table 1, where we observe a significant energy saving when shifting the generative AI workload to the edge.

4.2 Environmental Footprints

In our experiment, we focus on two key environmental footprints — carbon footprint and water footprint — estimated based on the profiling results in Section 4.1.

Carbon Footprint: We choose the carbon intensity $\gamma_c = 390 \text{ gCO}_2/\text{kWh}$ for the operational carbon footprint, according to the average carbon emission by electricity generation in the U.S. [50]. For the two types of GPU servers we consider on Google Colab, the embodied carbon per GPU is set as 518 kgCO₂, on a par with [51]. We do not distinguish the embodied carbon difference between different GPUs, as the commonly-used embodied carbon footprints of GPUs are only rough estimates due to the lack of accurate data sources [52]. During the manufacturing process, distribution, and disposal, the carbon emission of Samsung S24 is estimated to be 39.2 kgCO₂ based on the Life-Cycle-Assessment result from Samsung's environmental report [53]. All these embodied carbon footprints are amortized over 5 years and the utilization ratios of cloud and mobile phone are set as 1 and 0.19, reflecting an estimated of 279 minutes per day for non-voice phone activities in the U.S. based on [54]. The

Model Name	NVIDIA L4		NVIDIA A100		Samsung S24 NPU		
	Latency (s)	Energy (J)	Latency (s)	Energy (J)	Latency (s)	Energy (J)	Energy Saving
TrOCR	0.03	3.24	0.02	3.39	0.11	0.84	75%
Stable Diffusion 1.5	3.35	539.36	2.24	714.86	4.64	34.27	95%
Stable Diffusion 2.1	6.95	1118.12	2.27	1154.89	9.57	70.73	94%
Riffusion	3.38	542.60	2.27	719.95	4.70	34.72	95%
ControlNet	6.64	1067.77	3.52	1292.34	14.03	103.62	92%
Llama-2-7B	23.56	3477.18	22.00	3803.71	35.50	262.26	93%

Table 1: Average inference time (seconds) and energy consumption (joules) for profiled AI models on three devices: Nvidia A100, Nvidia L4, and Samsung S24. The average token length of Llama-2 responses is **373**. The minimum energy values for each generative model are highlighted in bold. Energy savings of the Samsung S24 are calculated relative to the energy consumption of the Nvidia A100 when running the same model inputs.

Model Name	Total Carbon Footprint (Operational, Embodied)						
	NVIDIA L4		NVIDIA A100		Samsung S24		Carbon Saving
TrOCR	0.45	(0.35, 0.10)	0.45	(0.37, 0.08)	0.24	(0.09, 0.15)	46%
Stable Diffusion 1.5	69.44	(58.43, 11.01)	84.81	(77.44, 7.37)	9.86	(3.71, 6.15)	88%
Stable Diffusion 2.1	143.95	(121.13, 22.82)	132.56	(125.11, 7.45)	20.35	(7.66, 12.69)	85%
Riffusion	69.87	(58.78, 11.09)	85.44	(77.99, 7.45)	9.99	(3.76, 6.23)	88%
ControlNet	137.47	(115.67, 21.79)	151.57	(140.00, 11.57)	29.81	(11.23, 18.59)	80%
Llama-2-7B	454.04	(376.69, 77.34)	484.29	(412.07, 72.22)	75.45	(28.41, 47.04)	84%

Table 2: The average carbon footprint (gCO₂) per 1000 queries of the generative models, estimated with our model profiling results on Nvidia L4, Nvidia A100 and Samsung S24. The average carbon saving of Samsung S24 is calculated relative to the total carbon footprint of Nvidia A100 under the identical model inputs.

Model Name	Total Water Footprint (Scope 1, Scope 2)						
	NVIDIA L4		NVIDIA A100		Samsung S24		Water Saving
TrOCR	0.004	(0.001, 0.003)	0.004	(0.001, 0.003)	0.001	(0.000, 0.001)	81%
Stable Diffusion 1.5	0.621	(0.150, 0.471)	0.822	(0.199, 0.624)	0.030	(0.000, 0.030)	96%
Stable Diffusion 2.1	1.286	(0.311, 0.976)	1.329	(0.321, 1.008)	0.062	(0.000, 0.062)	95%
Riffusion	0.624	(0.151, 0.474)	0.828	(0.200, 0.628)	0.030	(0.000, 0.030)	96%
ControlNet	1.228	(0.297, 0.932)	1.487	(0.359, 1.128)	0.090	(0.000, 0.090)	94%
Llama-2-7B	4.001	(0.966, 3.035)	4.376	(1.057, 3.320)	0.229	(0.000, 0.229)	95%

Table 3: The average water footprint (L) per 1000 queries of the generative models, estimated with our model profiling results on Nvidia L4, Nvidia A100 and Samsung S24. The average water saving is calculated relative to the total water footprint of Nvidia A100 under the identical model inputs.

higher utilization ratio in cloud servers offers some advantages for the cloud, as this can reduce the amortized carbon and incur a lower embodied carbon under the same condition. As shown in Table 2, the generative models deployed on our mobile phone shows significant reduction in both operational and embodied carbon emissions.

Water Footprint: Adopting the same methodology as the carbon footprint estimation, the water footprint is derived from energy consumption using water intensity γ_w . In our experiment, we mainly evaluate the scope-1 and scope-2 water consumption associated with deploying the generative AI model. Scope 1 consumption, primarily driven by the cooling requirements of cloud servers hosted in data centers, represents the direct water usage. Based on the data center water usage efficiency of Google [12] and Equinix (a major colocation provider) [11], we set the on-site water intensity factor as 1 L/kWh. As the edge device does not directly consume water, its scope-1 water consumption is set as zero. Moreover, the scope-2 water footprint is primarily attributed to the electricity generation process, with cloud servers and edge devices sharing the same water intensity. As reported in [55], the average grid water consumption factor in the U.S. is 3.142 L/kWh. Table 3 summarizes the scope-1 and -2 water consumptions of our selected generative AI models on cloud servers and edge devices.

Additional results that compare cloud inference and GPU-based edge inference are available in Section A, suggesting a

similar trend that edge devices can be a competitive alternative to clouds in terms of the energy and environmental footprints for generative AI inference.

5. CONCLUSION

This paper presents a case study to investigate the energy consumption and environmental footprints of selected generative AI models deployed in cloud and edge platforms. Our analysis focuses on the inference stage, where we profile the energy consumption and estimate the environmental footprint through detailed system modeling and publicly available data. Our findings suggest that potential energy saving and environmental benefits could be realized by shifting generative AI inference workloads from the cloud to edge devices. It is important to note that the generative AI models used in our experiment can be further optimized in both cloud and edge environments. Thus, the actual energy saving and environmental benefits may vary based on the specific optimization strategies implemented, and further research is needed.

Acknowledgement

This work was supported in part by the NSF under CCF-2324941.

6. REFERENCES

- [1] Eren Çam, Zoe Hungerford, Niklas Schoch, Francys Pinto Miranda, Carlos David Yáñez de León, Carlos Fernández Álvarez, International Energy Agency, et al. *Electricity* 2024. 2024.
- [2] International Energy Agency. Data centres and data transmission networks. <https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks>.
- [3] Facts and Factors. Global artificial intelligence market, 2021.
- [4] Datacenter Frontier. Virginia state legislators target data center development with new bills. <https://www.datacenterfrontier.com/site-selection/article/21545689/virginia-state-legislators-target-data-center-development-with-new-bills>.
- [5] The Washington Post. Internet data centers are fueling drive to old power source. <https://www.washingtonpost.com/business/interactive/2024/data-centers-internet-power-source-coal/>, 2024.
- [6] Thomson Reuters. OpenAI CEO Altman says at Davos future AI depends on energy breakthrough, 2024.
- [7] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B. Letaief. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials*, 19(4):2322–2358, 2017.
- [8] Pengfei Li, Jianyi Yang, Adam Wierman, and Shaolei Ren. Towards environmentally equitable ai via geographical load balancing. In *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*, e-Energy '24, page 291–307, New York, NY, USA, 2024. Association for Computing Machinery.
- [9] U.S. EPA. About the U.S. electricity system and its impact on the environment. <https://www.epa.gov/energy/about-us-electricity-system-and-its-impact-environment>.
- [10] Pengfei Li, Jianyi Yang, Mohammad A Islam, and Shaolei Ren. Making ai less "thirsty": Uncovering and addressing the secret water footprint of ai models. *arXiv preprint arXiv:2304.03271*, 2023.
- [11] Equinix. Equinix sustainability report 2023. https://sustainability.equinix.com/wp-content/uploads/2024/07/Equinix-Inc_2023-Sustainability-Report.pdf.
- [12] Google. Google 2024 environmental report - google sustainability. <https://www.gstatic.com/gumdrop/sustainability/google-2024-environmental-report.pdf>.
- [13] Shvetank Prakash, Matthew Stewart, Colby Banbury, Mark Mazumder, Pete Warden, Brian Plancher, and Vijay Janapa Reddi. Is tinyml sustainable? *Commun. ACM*, 66(11):68–77, oct 2023.
- [14] Arya Tschand, Arun Tejusve Raghunath Rajan, Sachin Idgunji, Anirban Ghosh, Jeremy Holleman, Csaba Kiraly, Pawan Ambalkar, Ritika Borkar, Ramesh Chukka, Trevor Cockrell, et al. Mlperf power: Benchmarking the energy efficiency of machine learning systems from microwatts to megawatts for sustainable ai. *arXiv preprint arXiv:2410.12032*, 2024.
- [15] VUPICO SDP. Sustainability data platform. <https://vupico-sdp.com/>.
- [16] Pré Consultants. Simapro: Life cycle assessment software. <https://simapro.com/>.
- [17] Qualcomm. The most advanced snapdragon mobile platform, snapdragon 8 gen 3 for galaxy, powers the new samsung galaxy z series globally.
- [18] Malladi Durga and Lawlor Pat. What is an NPU? and why is it key to unlocking on-device generative AI? <https://www.qualcomm.com/news/onq/2024/02/what-is-an-npu-and-why-is-it-key-to-unlocking-on-devic>
- [19] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.
- [20] Benoit Courty, Victor Schmidt, Sasha Luccioni, Goyal-Kamal, MarionCoutarel, Boris Feld, Jérémy Lecourt, LiamConnell, Amine Saboni, Inimaz, supatomic, Mathilde Léval, Luis Blanche, Alexis Cruveiller, ouminasara, Franklin Zhao, Aditya Joshi, Alexis Bogroff, Hugues de Lavoreille, Niko Laskaris, Edoardo Abati, Douglas Blank, Ziyao Wang, Armin Catovic, Marc Alencon, Michał Stęchły, Christian Bauer, Lucas Otávio N. de Araújo, JPW, and MinervaBooks. mlco2/codecarbon: v2.4.1, May 2024.
- [21] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for perceiving and processing reality. In *Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [22] Qualcomm. Qualcomm Neural Processing SDK for AI. <https://www.qualcomm.com/developer/software/neural-processing-sdk-for-ai>.
- [23] Minghao Li, Tengchao Lv, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. TrOCR: Transformer-based optical character recognition with pre-trained models, 2021.
- [24] Stability AI. Stable Diffusion 2.1. <https://huggingface.co/stabilityai/stable-diffusion-2-1>.
- [25] RunwayML. Stable Diffusion v1.5. <https://huggingface.co/runwayml/stable-diffusion-v1-5>.
- [26] Riffusion. Riffusion Model v1. <https://huggingface.co/riffusion/riffusion-model-v1>.
- [27] llyasviel/sd-controlnet-canny · Hugging Face — huggingface.co. <https://huggingface.co/llyasviel/sd-controlnet-canny>.
- [28] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton

- Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenxin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [29] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Inigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 118–132, 2024.
- [30] Jovan Stojkovic, Chaojie Zhang, Ínigo Goiri, Josep Torrellas, and Esha Choukse. Dynamollm: Designing llm inference clusters for performance and energy efficiency. *arXiv preprint arXiv:2408.00741*, 2024.
- [31] Qualcomm. Qualcomm AI Hub. <https://app.aihub.qualcomm.com/docs/index.html>.
- [32] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenxin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.
- [33] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.
- [34] Meta-llama llama-2-7b-chat-hf. <https://huggingface.co/meta-llama/Llama-2-7b-chat-hf>.
- [35] Llama2-7b chat. https://aihub.qualcomm.com/models/llama_v2_7b_chat_quantized.
- [36] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.
- [37] Robin Rombach, Patrick Esser, and David Ha. Stable diffusion v2 model card. <https://huggingface.co/stabilityai/stable-diffusion-2>.
- [38] Stable-diffusion-v2.1. https://aihub.qualcomm.com/models/stable_diffusion_v2_1_quantized.
- [39] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.
- [40] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.
- [41] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023.
- [42] Controlnet. https://aihub.qualcomm.com/models/controlnet_quantized.
- [43] Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Li, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. TrOCR: Transformer-based optical character recognition with pre-trained models, 2022.
- [44] TrOCR. <https://aihub.qualcomm.com/models/trocr>.
- [45] vicgalle/alpaca-gpt4 · Datasets at Hugging Face — huggingface.co. <https://huggingface.co/datasets/vicgalle/alpaca-gpt4>.
- [46] Zijie J. Wang, Evan Montoya, David Munechika, Haoyang Yang, Benjamin Hoover, and Duen Horng Chau. DiffusionDB: A large-scale prompt gallery dataset for text-to-image generative models. *arXiv:2210.14896 [cs]*, 2022.
- [47] corto-ai/handwritten-text · Datasets at Hugging Face — huggingface.co. <https://huggingface.co/datasets/corto-ai/handwritten-text>.
- [48] CUI Inc. Efficiency standard for external power supplies. <http://www.cui.com/catalog/resource/efficiency-standards-for-external-power-supplies.pdf>.
- [49] Samsung Newsroom. Smartphone chargers that put the planet first. <https://news.samsung.com/global/a-greener-galaxy-1-smartphone-chargers-that-put-the-pl>.
- [50] U.S. Energy Information Administration. How much carbon dioxide is produced per kilowatthour of u.s. electricity generation? <https://www.eia.gov/tools/faqs/faq.php?id=74&t=11>.
- [51] GenAI Impact. Environmental impacts of llm inference. https://ecologits.ai/latest/methodology/llm_inference/.

- [52] Boavizta. Digital & environment : How to evaluate server manufacturing footprint, beyond greenhouse gas emissions? <https://boavizta.org/en/blog/empreinte-de-la-fabrication-d-un-serveur>.
- [53] Samsung. From raw materials to disposal, efforts to reduce carbon continue. <https://semiconductor.samsung.com/sustainability/environment/green-chip/life-cycle-assessment/>.
- [54] Statista. Time spent with nonvoice activities on mobile phones every day in the United States from 2019 to 2024. <https://www.statista.com/statistics/1045353/mobile-device-daily-usage-time-in-the-us/>, 2024.
- [55] Paul Reig, Tianyi Luo, Eric Christensen, and Julie Sinistore. Guidance for calculating water use embedded in purchased electricity, February 2020. World Resources Institute.

APPENDIX

A. MEASUREMENT FROM ADDITIONAL EDGE DEVICES

Our primary findings reveal a substantial energy reduction when generative AI workloads are shifted from the cloud to edge devices. To demonstrate this, we deployed large language models (LLMs) on two Google phones, the Google Pixel 8 and Pixel 9. The Google Pixel 8 is a contemporary device released in the same year as the Samsung Galaxy S24, and Google Pixel 9 is the upgraded version of Pixel 8 with more powerful SoC and larger RAM. Using the MediaPipe Solution toolchain, we executed several LLMs entirely on-device and measured their end-to-end latency.

Beyond model size and quantization, factors like thermal throttling significantly influence real-world latency and energy consumption. For instance, as inferring LLMs on mobile SoCs is computationally intensive, excessive heat will be accumulated on chipset, triggering thermal throttling. Such procedures may include the clock frequency or reduce CPU utilization, leading to increased inference latency for LLMs. To account for these thermal effects, we averaged the end-to-end latency over 100 input prompts executed consecutively on each edge device.

The average latency and energy consumption per prompt are summarized in Table 4. We choose to mobile GPU as the backend of LLMs inference since it’s the most computational efficient approach for the edge LLM deployment. The chipset Google Tensor G3, adopted by Google Pixel 8, is mainly designed for extended battery life instead of solely optimizing performance. According to our measurement, the average power of Google Tensor G3 is only 31% of the power of Qualcomm Gen 3 when inferring with LLMs. Therefore, the GPU performance of the Google Tensor G3 is not as powerful as the Qualcomm Gen 3 used by Samsung S24, resulting in longer latency for LLM inference on Google Pixel 8. And the Google Pixel 9, the successor of Pixel 8, is equipped with a more powerful chipset (Google Pixel G4) and significantly reduces the latency of LLM inference. Despite this, we observed at least 72% energy savings for both devices compared to deploying on high-end GPU servers.

Model Name		Falcon	Stablelm	Gemma-2B
Nvidia L4	Duration (sec)	13.79	20.48	8.25
	Energy (J)	2034.61	3032.00	1086.78
	Token/s	60.71	29.97	23.13
	Joule/Token	2.43	4.94	5.70
Nvidia A100	Duration (sec)	13.73	21.65	6.58
	Energy (J)	2380.51	3496.22	999.93
	Token/s	61.13	28.90	29.00
	Joule / Token	2.84	5.59	5.24
Samsung S24	Duration (sec)	43.22	35.69	6.07
	Energy (J)	279.45	243.22	47.25
	Token/s	23.09	13.35	33.08
	Joule / Token	0.29	0.51	0.22
	Saving	88%	93%	95%
Google Pixel 8	Duration (sec)	305.25	280.25	50.71
	Energy (J)	661.50	589.05	123.53
	Token/s	3.21	1.68	4.03
	Joule/Token	0.68	1.25	0.59
	Saving	72%	83%	88%
Google Pixel 9	Duration (sec)	63.35	52.32	11.04
	Energy (J)	388.80	336.83	71.33
	Token/s	14.13	8.40	14.08
	Joule/Token	0.43	0.74	0.41
	Saving	84%	90%	93%

Table 4: Latency and energy consumption of large language models deployed on cloud servers and edge devices. The edge devices leverage their GPUs to accelerate LLM inference. The average energy saving percentage is calculated against the total energy consumption of Nvidia A100 given the set of model inputs.