Generative AI at the Edge

Challenges and Opportunities

THE NEXT PHASE IN AI DEPLOYMENT

VIJAY JANAPA REDDI

By 2030, more than 50 billion edge devices—smartphones, AR (augmented reality) glasses, wearables, and industrial IoT (Internet of things) systems—will be generating, interpreting, and acting on data in real time. Imagine AR glasses that describe landmarks as you walk through a foreign city, a smartwatch that predicts stress levels from biometric data, or a home robot that collaborates with other devices to execute household tasks. These scenarios, once the realm of science fiction, are becoming reality as a paradigm shift unfolds: the transition from cloud-centric AI to edge-deployable intelligence.

This transformation is particularly significant with the advent of generative AI systems. Today's powerful models such as ChatGPT and DALL·E exemplify the current paradigm. These are massive frontier models hosted in centralized cloud environments serving millions of users across diverse applications.[3] These generative systems create new content—writing human-like text, producing realistic images, composing music, or generating

code—by learning patterns from vast training datasets and synthesizing novel outputs that weren't explicitly programmed. Their ability to understand context, follow instructions, and produce creative content has revolutionized how we interact with technology.

Yet as applications demanding real-time response, privacy protection, and operation with limited bandwidth become more prevalent, the limitations of cloud-dependent AI grow increasingly apparent. Edge computing addresses these challenges by shifting AI processing directly onto the billions of devices where data originates and users interact. Several compelling factors drive this shift to edge-based generative AI. First, by eliminating round-trip delays to distant data centers, local processing delivers the near-instantaneous responses essential for augmented reality experiences and robotic systems. Second, edge deployment enhances reliability, allowing devices to maintain AI capabilities even when network connectivity is intermittent or unavailable. Perhaps most significantly, on-device processing preserves privacy by keeping sensitive information—medical records, personal conversations, and biometric data—secure on the user's device rather than transmitting it to external servers. Beyond these user benefits, distributed edge inference offers broader sustainability advantages by potentially reducing the energy consumption and costs associated with massive centralized cloud infrastructure.

Despite these compelling advantages of edge deployment, significant technical hurdles remain. Today's state-of-the-art generative models are extraordinarily resource-intensive, often comprising hundreds of billions

of parameters, making them difficult to run outside specialized data centers. LLMs (large language models) commonly suffer from high inference latency, heavy memory footprints, and substantial power draw, which conflict with the constraints of edge environments. Large models also assume constant cloud connectivity for retrieving updates or external knowledge, whereas edge deployments may need to operate autonomously for extended periods of time.

These factors raise the central question: *How can we bring the power of large generative models to the edge while balancing efficiency, safety, and autonomy?* This article explores generative AI's evolution toward efficient models, proposes a taxonomy by size and deployment characteristics, and examines practical applications across healthcare, wearables, robotics, and IoT. It addresses critical challenges in the data-model-compute triangle, introduces novel metrics like hallucinations per watt-hour, and argues that deployable intelligence will ultimately unlock generative AI's potential in everyday devices.

EVOLUTION OF GENERATIVE AI

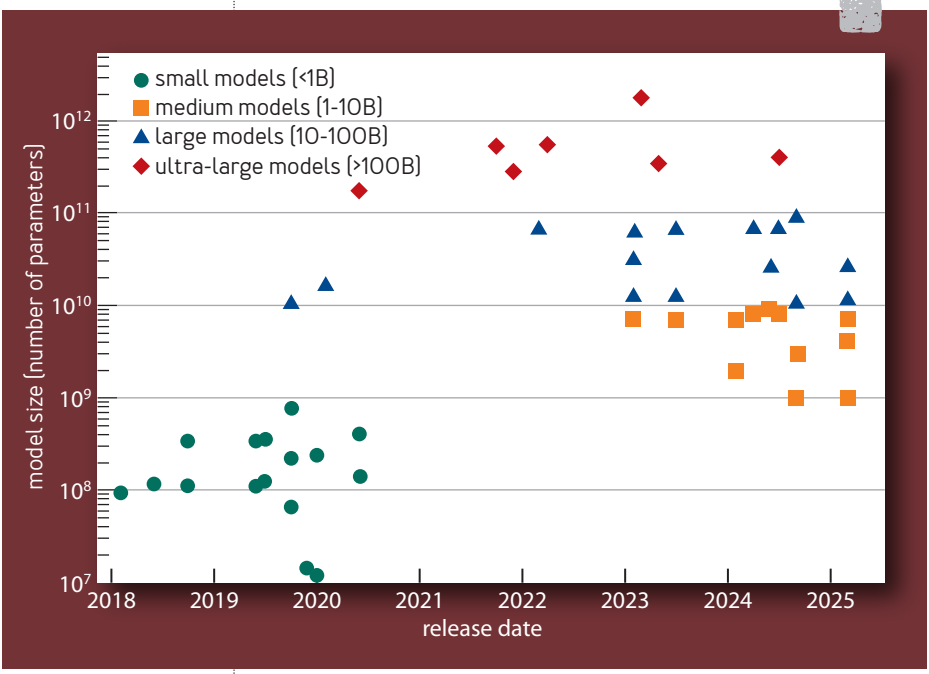
Understanding the evolution of generative AI is essential to this discussion of edge deployment. First, history reveals how models have grown increasingly powerful but also more resource-intensive—a tension that lies at the heart of edge AI challenges. Second, examining the technological breakthroughs that shaped today's models leads to identifying which innovations might be adapted or reimagined for resource-constrained environments. Finally, this historical context leads to an appreciation

of the current bifurcation in the field: While frontier models continue to grow in size and capability, a parallel trend toward smaller, more efficient models is emerging, specifically to address edge deployment needs.

Generative AI has evolved through several waves of innovation, from early sequence-to-sequence models to today's instruction-following multimodal behemoths. The journey began in the mid-2010s with RNN (recurrent neural network)-based seq2seq (sequence-to-sequence) models for tasks such as machine translation. By training an encoder-decoder RNN (often with long short-term memory cells) to map input sequences to output sequences,[24] Sutskever et al. demonstrated the first end-to-end neural translation system that could convert text from one language to another without relying on separate components for analysis, transfer, and generation. Soon after, the introduction of the attention mechanism allowed decoders to focus on relevant input parts, greatly improving sequence generation quality.[1] These advances cemented the seq2seq plus attention architecture as the workhorse of generative models in NLP (natural language processing).

A major inflection point came with the advent of the Transformer architecture. Vaswani et al. eliminated RNN recurrence in favor of self-attention, allowing much deeper and more parallelizable sequence models.[26] Transformers scaled sequence generation to new heights, leading to the first wave of LLMs. As illustrated in figure 1, this period marks the beginning of exponential growth in model parameters, with clear stratification into distinct size bands. The plot depicts the growth of language models

FIGURE 1: **EVOLUTION OF LANGUAGE MODELS**



over time, with the y-axis representing the raw parameter count on a logarithmic scale. Models are categorized into four distinct size bands: small models (10^8–10^9 parameters, highlighted in green), medium models (10^9–10^{10} parameters, highlighted in orange), large models (10^{10}–10^{11} parameters, highlighted in blue), and ultra-large models (>10^{11} parameters, highlighted in red).

OpenAI's GPT series and Google's BERT (though BERT is technically an encoder-only model) showed that scaling parameter counts into the billions yielded

dramatic improvements in language understanding and generation. The 175-billion-parameter GPT-3, for example, demonstrated the ability to learn new tasks after seeing just a few examples, known as few-shot learning, treating the text input it receives as a makeshift "program" to perform diverse tasks without needing any complicated updates to its underlying model.

Merely making models larger, however, revealed limitations: They often produced incoherent or "unfactual" outputs (so-called *hallucinations*), lacked fine-grained control, and could exhibit toxic or biased behaviors learned from training data. The next evolution addressed these issues via instruction tuning and human feedback alignment. Researchers fine-tuned large models on instruction-response datasets and employed RLHF (reinforcement learning from human feedback) to align model outputs with human preferences. Notably, Ouyang et al. showed that a 1.3B parameter GPT-3 model fine-tuned with RLHF (dubbed InstructGPT) was preferred by humans over the original 175B GPT-3's outputs.[17] This startling result indicated that *smaller, aligned models can outperform larger but unaligned models* in following user instructions. Instruction-tuned models such as OpenAI's InstructGPT and Anthropic's RLHF-trained Claude proved far more usable, marking an industrywide pivot from raw generative models to those tuned for helpfulness and safety.

Meanwhile, generative AI expanded into new modalities. Vision models progressed from early GANs (generative adversarial networks) to autoregressive and diffusion models that could synthesize high-fidelity images. For example, OpenAI's image generator DALL·E (2021) used a

Transformer to sequentially generate image tokens, while its successor DALL·E 2 (2022) employed diffusion models to iteratively refine images. Diffusion models, which gradually denoise random inputs into coherent images, have now become state of the art for image and video generation. Multimodal generative models soon followed. Models such as Flamingo and PaLM-E can accept both text and visual inputs and generate multimodal outputs (e.g., image captions or robot actions).[6] By 2023, multimodal LLMs such as OpenAI's GPT-4 demonstrated the ability to process text and images together, blurring boundaries between language and vision domains.

As model capabilities grew, so did model size—until practical deployment became a concern. A countertrend has emerged to develop SLMs (small language models)—that is, models with far fewer parameters (on the order of 10^8–10^9) that retain useful generative abilities. As illustrated in figure 1, the evolution of language models shows both the trend toward increasingly large models and the emerging countertrend focusing on smaller, more efficient models suitable for deployment at the edge. Sun et al. introduced MobileBERT,[23] a compact variant of BERT optimized for resource-constrained devices, while Lan et al. demonstrated how ALBERT reduced model size while maintaining performance through parameter sharing and embedding factorization.[12]

Two key approaches have enabled surprisingly capable small models. First, knowledge distillation compresses a large model's knowledge into a smaller model.[8] For example, DistilBERT (2019) retained 97 percent of BERT's language understanding capability with only about 40

percent of the parameters, running 60 percent faster. Second, quantization uses lower-precision arithmetic to reduce memory requirements. Recent 8-bit and 4-bit quantization methods allow large models to run with minimal loss in quality.[5] These advances fuel the shift toward SLMs that can operate on consumer hardware.

Companies are already shipping SLMs on edge devices: Recent smartphones incorporate on-device language models (e.g., Google's Gemini Nano service), and Apple's Neural Engine runs local Transformer models for tasks such as auto-text and dictation. The trajectory of generative AI is thus bifurcating: On one side, ever-larger frontier models push state-of-the-art performance; on the other, optimized smaller models bring AI to everyday devices.

Another notable development is knowledge retrieval. Even a small model can appear highly knowledgeable if it can query an external knowledge base. RAG (retrieval-augmented generation) techniques prepend relevant documents from a database or the web to the model's input context,[7] supplying facts the model may lack. Lewis et al. showed that a retrieval-augmented model set new state-of-the-art on open-domain QA tasks, outperforming purely parametric models.[13] By offloading the "memorization" of world knowledge to an external store, retrieval allows the core model to remain relatively compact without sacrificing factual accuracy. This idea is now common in production systems (e.g., search engine chatbots): A moderately sized language model backed by a search index or vector database can rival a much larger stand-alone model in practical utility.

Generative AI has evolved from modest seq2seq models to giant multimodal LLMs but is now entering an era of optimization and deployment-conscious design. While significant interest has focused on scaling LLMs to achieve state-of-the-art performance through massive parameter counts, GenAI at the edge requires a fundamentally different approach. Key inflection points, such as the invention of Transformers, the use of human feedback alignment, the rise of retrieval augmentation, and advances in model compression, are enabling a new class of small, specialized, and deployable models. These SLMs do not aim to win on leaderboard benchmarks through sheer size; instead, they strive for a "good-enough" sweet spot between capability and efficiency suitable for running at the edge.

TAXONOMY OF GENERATIVE MODELS

As deploying generative AI to the edge comes closer to reality, a clear taxonomy becomes essential for several reasons. First, it helps systematically evaluate which model types can realistically operate under edge constraints. Second, it provides a framework for identifying optimization opportunities specific to each model category. Third, a structured taxonomy allows researchers and practitioners to track progress and identify gaps in the development of edge-friendly generative AI. Finally, it facilitates meaningful comparison between models across different dimensions relevant to edge deployment. To this end, generative models can be classified along multiple axes: model size (parameter count), architecture, modality of input/output, and intended deployment environment.

Model size

Models are categorized into small, medium, large, and ultra-large frontiers:

- *Small models* (under ~1 billion parameters), such as DistilGPT-2, ALBERT,[12] and MobileBERT,[23] make efficiency a priority and often result from compressing larger models. These models typically range in size between ~100MB and 2GB, making them suitable for smartphones, tablets, and edge devices with limited computing resources.
- *Medium models* (1–10 billion parameters), such as the 7B-parameter LLaMA, balance performance and efficiency, with model sizes ranging from ~2GB to 20GB, making them suitable for high-end mobile devices or single GPUs.
- *Large models* (10–100 billion parameters), including flagship LLMs such as GPT-3 (175B) and Meta's 200B+ models, typically require server-class accelerators and occupy ~20GB to 200GB of memory.
- *Ultra-large frontier models* (100 billion+ parameters), such as Google's PaLM (540B) and GPT-4 (1.8T sparse parameters), push the limits of current hardware with sizes exceeding ~200GB, making them exclusive to large-scale cloud environments.

Model size correlates with knowledge and linguistic fluency but also impacts memory, runtime, and energy—key factors in the Deployment Readiness Matrix. Table 1 illustrates how these model size categories align with different deployment environments. The Deployment Readiness Matrix maps the feasibility of running different model sizes across various deployment environments.

TABLE 1: : DEPLOYMENT READINESS MATRIX FOR EDGE GENERATIVE AI

	IOT/ MICROCONTROLLER	EDGE DEVICE (PHONE/WEARABLE)	FOG/ENTERPRISE (EDGE SERVER)	CLOUD (DATA CENTER)
SMALL (<1B PARAMETERS) ~100MB-2GB MODEL SIZE	CHALLENGING Heavily quantized. Specialized arch. Limited capabilities.	IDEAL 8-bit quantization. 100-500ms latency. Modern NPU required.	IDEAL Can serve multiple users/devices. Efficient & responsive.	IDEAL Massive batching. Low-latency API. Energy efficient.
MEDIUM (1-10B PARAMETERS) ~2-20GB MODEL SIZE	INFEASIBLE Exceeds memory constraints.	CHALLENGING High-end phones only. 4-bit quantization. Thermal limitations.	IDEAL GPU/TPU accelerated. Good inference speed. Multi-user capable.	IDEAL High throughput. Cost effective. Scalable.
LARGE (10-100B PARAMETERS) ~20-200GB MODEL SIZE	INFEASIBLE Exceeds memory constraints.	INFEASIBLE Exceeds memory & compute constraints.	CHALLENGING Multiple GPUs. Higher latency. Power constraints.	IDEAL Standard deployment. GPU/TPU clusters. Elastic scaling.
ULTRA-LARGE (100B+ PARAMETERS) ~200GB+ MODEL SIZE	INFEASIBLE Exceeds memory constraints.	INFEASIBLE Exceeds memory & compute constraints.	INFEASIBLE Exceeds memory & compute constraints.	IDEAL Distributed inference. High throughput. Dynamic workload balancing.

Color coding indicates deployment feasibility: green (ideal), yellow (challenging but possible), and red (infeasible). Small models (<1B parameters) are viable across most environments, while ultra-large models (100B+ parameters) are restricted primarily to cloud deployments.

The matrix highlights how memory, computational, and power constraints progressively limit deployment options as model size increases. It maps the feasibility of running various model sizes across the spectrum from microcontrollers to cloud servers, highlighting the practical constraints each combination faces. As shown, only small models are viable candidates for true edge deployment, while medium models remain challenging for most mobile devices but work well in enterprise settings. Large and ultra-large models are primarily restricted to cloud environments, with some potential for large models in well-provisioned on-premises servers.

Architecture

Generative models span various neural architectures. The dominant class for language is the Transformer (using self-attention), which powers GPT, BERT, T5, etc., such as those shown in figure 1. For images, autoregressive Transformers (e.g., GPT), GANs, VAEs (variational autoencoders), and diffusion models have all been popular. GANs consist of a generator and discriminator in a min-max game, often producing photorealistic outputs (e.g., StyleGAN for images). VAEs learn probabilistic latent representations to generate data, which is useful for tasks such as anomaly detection.

Diffusion models are newer likelihood-based models that have eclipsed GANs for image and audio generation because of their stability and quality (e.g., the latent diffusion in stable diffusion, or audio diffusion models for speech). There are also hybrid architectures: VQ-VAE combined with autoregression (as in DALL·E's discrete

VAE + Transformer) or Transformer models trained with diffusion-based objectives. Notably, model architecture often dictates computational complexity—Transformers scale quadratically with sequence length, while diffusion requires many iterative steps—influencing their suitability for edge runtime. Simpler RNN- or CNN (convolutional neural network)-based generators might be preferred for low-power devices if they achieve acceptable results.

Modality

Generative AI now covers text, vision, audio, code, and combinations. Models can be categorized by the type of content they generate. Text generators include language models (GPT-2, GPT-3, etc.) primarily producing natural language. Image generators include GANs, such as BigGAN and StyleGAN, and diffusion models, such as Imagen, which create or transform images. Audio generative models produce speech or music (e.g., WaveNet for realistic speech audio, Jukebox for music). Video generators (e.g., Diffusion Models) extend image models along the time axis, synthesizing video clips. Some models handle multimodal outputs: For example, a model might generate an image from a text description (text-to-image, such as DALL·E) or vice versa (image captioning via a generative language model). There are also code-generation models (such as OpenAI Codex, which generates programming code from natural language).

Each modality has different input/output structure and evaluation metrics (e.g., BLEU/ROUGE for text, FID for image fidelity). Still, interestingly, the underlying model technology (Transformers, diffusion, etc.) has started to

converge or cross-pollinate across modalities. For edge deployment, the modality matters because it dictates the required sensors (cameras, microphones) and the real-time requirements (e.g., generating audio for a live conversation is more time-sensitive than generating an email draft).

Deployment class

This axis distinguishes models by the environments in which they are intended to run. A cloud model assumes ample GPU/TPU (tensor processing units) resources and benefits from virtually infinite memory and elastic scaling; it might prioritize maximum accuracy and accept high computational cost (e.g., GPT-4 or Imagen's largest configuration). An edge model is optimized for stand-alone operation on consumer-grade hardware (mobile SoC, laptop CPU/GPU, etc.), prioritizing low latency and efficiency.

There is also an intermediate fog/enterprise model class: models deployed on premises in controlled environments such as factory servers or 5G edge servers, where moderate computing is available but not to the scale of a hyperscaler's cloud.

At the extreme low end, microcontroller or IoT models operate under severe constraints (e.g., a few megabytes of memory, no hardware accelerator). The TinyML community has achieved remarkable feats such as keyword-spotting neural nets on microcontrollers; generative TinyML (such as on-device phrase suggestion) is an emerging frontier.[27]

USE CASES AT THE EDGE

The taxonomy reveals that edge deployment involves specific combinations of model size (typically small to

medium), architectures optimized for efficiency, modalities relevant to the application, and deployment configurations suited to device constraints. Based on this information, there are several domains where on-device or edge-local generative models enable transformative applications: healthcare, wearables and AR, robotics, and industrial/IoT. Each brings unique latency, modality, and privacy requirements, illustrating why a one-size-fits-all cloud model is often insufficient.

Healthcare and medical devices

In medicine, data privacy and immediacy can be matters of life or death. Imagine a smart endoscope that, during a procedure, generates a summary of observations for the surgeon in real time, or an insulin pump that continually translates glucose sensor readings into natural-language dietary suggestions for the patient. These generative tasks must be performed locally because of privacy (patient data cannot leave the device) and reliability (no Internet in an operating room).

On-device language models are being explored for clinical decision support and documentation drafting. For example, a doctor's smart assistant might summarize a patient's medical history and suggest possible diagnoses on the tablet *during* an examination, without sending sensitive records to the cloud. Early studies indicate this is feasible: Nissen et al. benchmarked compact models[15] such as a 2.7B-parameter Phi-3 on smartphones for clinical reasoning, finding they could achieve reasonable accuracy with acceptable speed, especially when fine-tuned on medical knowledge. Specialized small models (MedLMs)

such as Med42 and Aloe have been fine-tuned on medical QA and show high accuracy, albeit often requiring more memory than older devices have.

Key challenges in this domain include the need for guaranteed factuality (a hallucinated diagnosis can be dangerous) and the ability to learn from new data while preserving patient confidentiality continuously. Privacy-preserving fine-tuning (e.g., on-device learning or federated learning among hospitals) is an active research area so that local generative models can stay up to date without a central data pool.

Wearables and augmented reality

Wearable devices such as smart glasses, earbuds, and watches are highly personal and context-aware, making them ideal hosts for generative AI that tailors itself to an individual's needs. Consider AR glasses, such as the Ray-Ban Meta glasses, that see what the user sees and whisper contextual information: "This product has four-star reviews" or "The person approaching is John; you met him at the 2019 conference." For such glasses to be socially acceptable, they must process visual inputs and generate outputs (text or audio) on-device, since continuously streaming egocentric video to the cloud would be a privacy nightmare. An edge-based generative model could perform on-the-fly scene description for visually impaired users, or translate foreign text in the user's view (a form of generative image-to-text). Wearable health monitors could use small generative models to translate raw sensor readings into coherent insights ("Your stress level today is higher than usual, maybe take a walk").

The modality constraints here are key: Wearables often deal with multimodal data (accelerometer, GPS, camera, microphone) but have limited computing (a smartwatch has perhaps a few-hundred-MHz CPU budget for AI). This drives interest in multimodal SLMs, efficient models that can jointly handle text, audio, and simple visuals. Qualcomm, for example, has demonstrated on-headset AI to generate mixed reality scenes locally. Another example is personal voice assistants on earbuds that run offline—recent flagship phones can run a condensed speech recognition and synthesis stack entirely on-device, enabling features such as real-time language translation during a conversation, all without the cloud.

For AR and wearables, low latency is paramount (the AI should augment reality in real time), and thermal limits are strict (no one wants hot or heavy glasses). These use cases push us toward specialized architectures, such as Transformer backbones optimized for low-power NPUs (neural processing units), and the integration of sensor-specific AI modules such as a small vision model feeding into a language model.

Robotics

Robots operating in the physical world, whether household robots, drones, or industrial automatons, increasingly incorporate generative AI for planning, perception, and interaction. The concept of a robotic foundation model has emerged: a large (often multimodal) model that can drive a range of robot behaviors. For example, Google's PaLM-SayCan system combines a language model with a robot execution model,[4] allowing a robot to parse

high-level instructions ("Bring me a snack") and generate a sequence of actions. More recent work such as RT-2 (Robotics Transformer) uses a vision-language model to directly output robot actions from visual inputs, essentially treating robot policy as a form of text generation (where the "text" is a sequence of motor commands).

Most of these demonstrations rely on cloud-scale models with server GPUs; the robot is tethered to hefty computing. The challenge at the edge is to embed sufficient intelligence on the robot's onboard computer to function in the field, where network connectivity may be unreliable (say, a drone inspecting a remote site). This calls for embodied SLMs, compact models that integrate sensor data—camera, lidar (light detection and ranging—with language or policy generation, all running on robotic hardware, which might be an NVIDIA Jetson or a DSP (digital signal processor).

The opportunities are vast: an autonomous vehicle could carry a generative model that narrates its decision logic to passengers in real time ("I'm slowing down because I see a cyclist ahead"), increasing trust. A home assistant robot could have a local LLM that converses naturally with a user and adapts to household-specific commands, without sending every voice query to an external server (addressing privacy for in-home interactions). In industrial settings, robots on a factory floor could share a generative model locally to coordinate their tasks, synthesizing and broadcasting plans to each other—a step toward multi-agent collaboration.

Robotics use cases benefit from edge AI for low-latency reflexes (a robot may need to react in milliseconds) and

for autonomy (a Mars rover must generate plans without Earth servers). That said, deploying generative models in robots introduces safety-critical considerations: An on-board model that generates an incorrect instruction could cause physical harm. This heightens the need for rigorous validation, fail-safes, or hybrid systems where a reliable classical control algorithm supervises a generative planner's "creative" suggestions.

Industrial and IoT applications

Many IoT devices generate massive streams of sensor data in industry. Generative models at the edge can transform this data into meaningful narratives or predictions. For example, an edge IoT gateway in a smart factory could use an SLM to generate summaries of machine-tool logs ("Machine A shows signs of wear in the spindle; maintenance recommended within 10 days.") instead of just sending raw log files upstream. In power grids, edge AI could simulate and forecast demand patterns by generating plausible future scenarios (a form of data synthesis).

Anomaly detection can also be cast as a generative task: A model learns the normal distribution of sensor readings and flags anything it cannot "generate" as likely abnormal. A concrete use case is an edge camera that not only detects an intruder (through a vision model), but also generates a textual report of the event ("At 3:05 pm, an unidentified person entered through the north gate, carrying what appears to be a toolbox."). Generating that report locally means sensitive security footage never leaves the premises unencrypted.

In industrial settings, privacy is less about personal data and more about proprietary data (trade secrets in manufacturing processes); local generation helps keep that information in-house. Latency is important when quick decision loops are needed (e.g., an edge AI that generates a control signal to shut off a valve upon detecting a hazardous situation). Many industrial edge devices are small computers with some acceleration (e.g., an NVIDIA Jetson Nano or Google Coral dev board), capable of running medium-size models but under tight power budgets.

Thermal constraints and 24/7 reliability are big here: An edge generative model might be expected to run continuously, so it must be efficient to avoid overheating or draining power reserves, and robust to avoid crashes (memory leaks in a factory robot's AI could halt a production line).

Across these use cases, some common themes emerge. First, multimodality is often required at the edge—devices have sensors (cameras, mics, etc.) and must generate various outputs (text, speech, images). This favors modular or composite models (e.g., a small vision model feeding into a language model) or unified multimodal models if they can be made efficient enough.

Second, personalization and contextualization are key: Edge AI serves a specific user or environment, so it must adapt to context. A cloud model serves billions and averages over contexts, whereas an on-device model can specialize (learning a user's vocabulary, a factory's typical conditions, etc.). The next section addresses personalization techniques such as fine-tuning or learning

on local data, as they are both an opportunity and a challenge (because of limited local data).

Third, privacy and security are selling points and requirements for edge generative AI. Keeping data local protects privacy, but it is also important to ensure that the model itself is safe (e.g., it doesn't leak sensitive info it has memorized, and it's protected from tampering).

In summary, the edge deployment of generative models opens up applications that demand low latency, work under resource constraints, and often involve sensitive data or environments. The examples in table 2 illustrate the diverse demands on generative models at the edge. No single model will cover all; instead, there will likely be a proliferation of *specialized small models* tuned to their niches. The next section delves into the core technical

TABLE 2: **DIVERSE DEMANDS ON GENERATIVE MODELS AT THE EDGE**

USE CASE	GENERATIVE AI APPLICATION	MODEL REQUIREMENTS
Smart Hearing Aid	Real-time speech enhancement and translation	Audio generative model under 50 MB, ms latency, ultra-low power
AR Tourist Guide	Generation of descriptions of landmarks in the user's view	Multimodal vision-language model, on-device inference for real-time, cache of local knowledge for factual accuracy
Home Robot Assistant	Chats and plans tasks	Conversational LLM, possibly medium-sized but running on robot hardware; must be aligned to avoid unsafe instructions
Industrial Sensor Monitor	Generates daily reports from sensor data	A model trained on time series to text; can be low latency but must handle long inputs; likely runs on an embedded PC

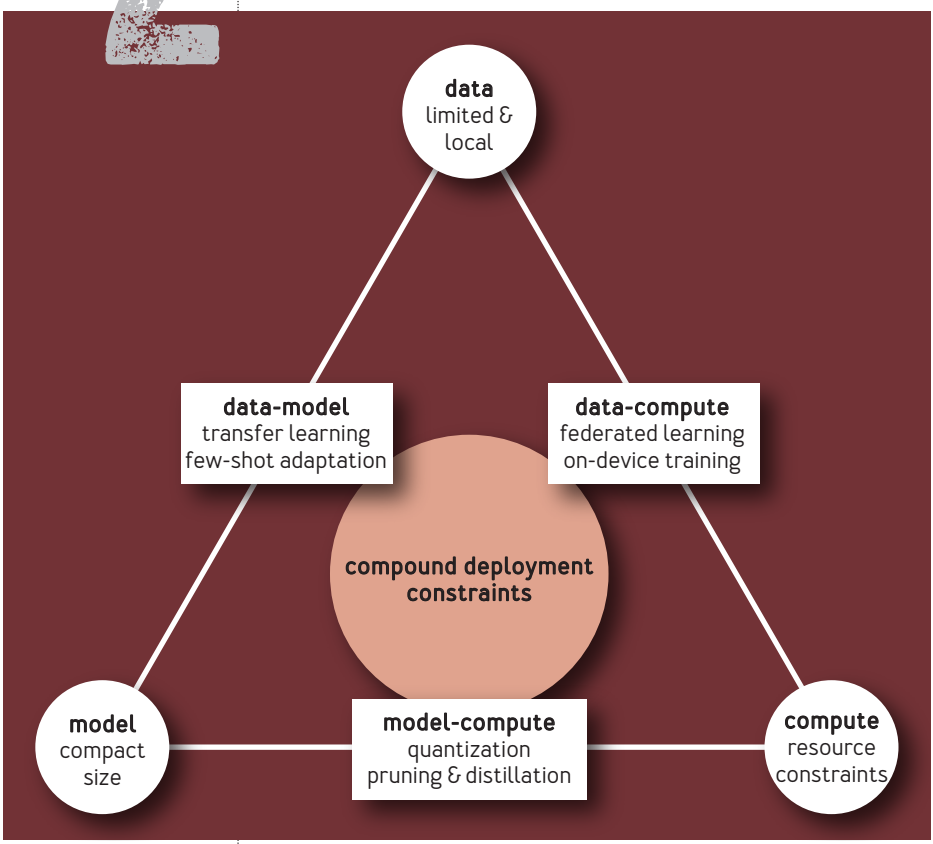
challenges that must be overcome to realize these edge use cases, structured by data, model, and compute considerations.

CORE CHALLENGES FOR EDGE-GENERATIVE AI

Deploying generative AI at the edge requires overcoming a confluence of challenges. These challenges can be organized into three interacting dimensions: data, model, and compute. This section examines each in turn, noting how they intersect. This gives rise to what we call the DMC (data–model–compute) interaction triangle (as illustrated in figure 2), wherein each corner imposes constraints that compound when considered together. Each vertex corresponds to a fundamental constraint: limited local *data*, restricted *compute* resources, and the need for compact *models*. Edges represent pairwise interactions (e.g., data-compute: federated learning to handle data locally; model-compute: quantization/truncation to fit hardware; data-model: transfer learning to adapt models with little data).

The center of the triangle, where all three constraints converge, is the most challenging region, requiring holistic optimization of the DMC tradeoffs. Edge generative AI lives in this regime of compound deployment constraints, where optimizing one aspect often exacerbates another. Data is often scarce and siloed in edge settings, models must be small and efficient, and compute resources (including energy and thermal headroom) are limited. Meeting one of these constraints is hard enough; meeting all three simultaneously is the crux of edge AI deployment.

FIGURE 2: **THE DATA–MODEL–COMPUTE INTERACTION TRIANGLE FOR EDGE AI**



Data constraints

Unlike cloud models trained on vast centralized datasets, edge-deployed generative models must contend with limited and decentralized data. A single-edge device (say a smartphone or an IoT sensor) sees only a slice of data, which may be insufficient to train or even fine-tune a powerful model. This leads to several challenges.

Data Scarcity and Quality

Many generative use cases at the edge involve personal or situational data that is not part of the model's original training corpus. For example, a personal assistant may need to adapt to a user's writing style or a factory model might need to learn the normal patterns of a specific machine. The available data to fine-tune these specifics is typically small (perhaps a few documents or days of logs) and can be noisy or unlabeled. Fine-tuning large models on tiny datasets risks overfitting or learning spurious patterns (e.g., a language model might latch onto idiosyncrasies in a user's texts that degrade its general fluency). There is ongoing research into few-shot and zero-shot learning to make models more adaptable with minimal data and into data augmentation (possibly using the generative model itself to synthesize additional training examples).

In domains such as healthcare, labeled data is not just scarce but sensitive—a local model might have access to a patient's records, but using them for training triggers privacy concerns. Techniques such as differential privacy and FL come into play to allow learning from data without exposing it (Google's Gboard, for example, uses FL to improve its on-device keyboard suggestions across users). Federated approaches, however, can struggle with the highly varied and *non-iid* (independent and identically distributed) nature of edge data—one user's or one sensor's data distribution may be very different from another's. Benchmarks are being developed to evaluate FL performance under these diverse, non-iid conditions.[3]

On-Device Fine-Tuning (Personalization)

Ideally, an edge generative model could continuously learn from user feedback or new local data to improve over time (e.g., your chatbot becomes better at understanding your slang). Full backpropagation-based training on the device is usually prohibitive, however, because of compute and memory limits. Even if the device could handle it, there's the risk of the model overfitting to the user and losing generality (your personalized model might become useless for anyone else) and the risk of catastrophic forgetting (adapting to new data might degrade performance on what it previously knew).

Recent approaches such as LoRA (low-rank adaptation) and adapter modules offer a lightweight way to fine-tune only small parts of the model, reducing the computation and data needed. Another approach is prompt-based personalization: Rather than altering model weights, store personalized prompts or prefixes (sometimes called *soft prompts* or *embeddings*) that prime the model with user-specific context. This is akin to giving the model a quick memory of the user's data without retraining its core. The challenge is ensuring these personalizations don't compromise the model's base safety or inject biases (and also securing them—if someone extracted your prompt, would it reveal private info?). Communication-efficient federated learning strategies can mitigate bandwidth and energy costs of continual local adaptation.[11]

Privacy and Data Governance

Edge generative AI intersects user data and AI outputs, raising novel privacy questions. A model might inadvertently

output sensitive data on which it was trained (model inversion attacks in NLP have shown that rare training phrases can sometimes be regenerated verbatim). When models train or adapt on-device, mechanisms are needed to ensure they do not "leak" this data in their responses. This is part of a broader challenge of model audibility on the edge—unlike a cloud service where the provider can monitor for problematic outputs, an on-device model acts autonomously. The device might need tools that scan generated content for privacy violations or filter out content that looks too similar to private training data (e.g., a hospital's on-prem generative model accidentally spitting out a patient's name in a generic report).

Legal regimes such as GDPR (General Data Protection Regulation) also come into play: If a model on your phone fine-tunes on your data, is that considered your data (likely yes), and how is transparency or the right to be forgotten provided? These questions are largely open and will need both technical and policy innovation.

In summary, the data corner of the triangle in figure 2 demands methods for learning from limited data, sharing insights across devices without sharing raw data (federated or collaborative training), and maintaining privacy. When data is the bottleneck, one often turns to making the most of the model and computing resources (e.g., using a larger pretrained model so it needs less adaptation). But large models clash with the other corners, as seen in the next section.

Model constraints

The model itself—its size, architecture, and training—is a

central piece of the edge deployment puzzle. Challenges include making models smaller and more efficient (without losing too much performance), dealing with model hallucinations and errors in a resource-constrained setting, and ensuring model safety and robustness even after modifications such as quantization or pruning.

Model Compression (Size versus Performance)

Perhaps the most obvious challenge is that the best models are huge, and huge models don't fit or run well on edge devices. The research community has developed an arsenal of compression techniques—quantization, pruning, and distillation being the primary ones. Quantization reduces precision (e.g., 8-bit or 4-bit weights instead of 16/32-bit), dramatically shrinking model size and speeding up inference on hardware that supports low-precision math. Quantization-aware training or smart calibration methods (e.g., GPTQ, SmoothQuant) can achieve minimal accuracy loss even for LLMs in 8-bit. Recent work such as QT-DoG pushes further by exploring how quantized models can maintain generalization across unseen domains, an important factor for real-world edge applications.[9] Aggressive quantization, however, can hurt model "smoothness." Some users report that 4-bit quantized chat models produce more repetitive or garbled outputs, an indication of how reduced precision might subtly degrade the generation process. Another key compression approach is pruning, which removes redundant weights or entire neurons. It has been effective for vision models, but for dense language models, pruning tends to harm coherence unless done

carefully (e.g., magnitude pruning after fine-tuning).

Distillation, where a smaller "student" model is trained to imitate a larger "teacher," has yielded some of the best results in retaining performance in a smaller package. The downside is that distillation requires extensive training on possibly large corpora (often the original training data or a synthetic dataset generated by the teacher). For edge, one intriguing approach is online distillation: Could a device continually distill a cloud model's knowledge into a local model through interaction? For example, your phone queries a cloud model when it's online, but uses those query-answer pairs to improve its offline model. This would combine FL and distillation, but it's largely conceptual at this stage.

In addition to compression, another promising avenue for edge deployment is the use of MoE (mixture of experts) architectures. MoE models divide the overall model into specialized "experts," where only a subset of experts is activated per inference step. This sparsity reduces compute requirements while maintaining the performance of larger models. While MoE has demonstrated efficiency gains in cloud settings, adapting it for the edge introduces new challenges such as expert routing under latency constraints and dynamic expert selection in low-power environments.

Despite their theoretical benefits, MoEs have seen limited adoption in local models because of significant practical tradeoffs. They require either (1) more HBM (high-bandwidth memory) to store the full set of experts while using less compute per inference, or (2) complex expert offloading/onloading mechanisms that are

atypical for resource-constrained edge devices. These memory-compute tradeoffs often make traditional dense models more practical for current edge deployments. Nevertheless, MoE's ability to selectively allocate compute makes it a potentially attractive approach for balancing model size and performance in resource-constrained settings as edge hardware capabilities continue to evolve.

Hallucinations and Reliability

Generative models are notorious for fabricating information—a harmless quirk in a chatbot but potentially catastrophic in certain edge contexts (imagine a car's navigation AI hallucinating a road that doesn't exist). Large models mitigate hallucinations through techniques such as RLHF and RAG. On the edge, however, you might be using a much smaller model without those luxuries, or one that hasn't been through rigorous alignment training because its capacity to learn and store nuanced alignment preferences is limited by its size. Smaller models generally hallucinate more because they have less knowledge and linguistic finesse baked in. This is a serious issue for edge adoption—users must trust an on-device AI if it is to be genuinely useful. (After all, no one wants AR glasses that occasionally describe things that aren't there.)

One strategy is to narrow the scope of an edge model: For critical decisions, rely on deterministic algorithms or simple models while reserving generative flair for low-stakes tasks. Another approach involves on-device verification—for example, if an edge model generates a plan for a robot, a lightweight verifier (such as a classical planner

or a physics simulator) can assess the plan's feasibility. While this adds latency, it helps prevent obvious failures. A third approach is deferred decision-making, where the edge model offloads tasks to the cloud when uncertain. This approach aligns with the routing methodology described by Ong et al. in RouteLLM, which learns to route queries to appropriate models based on preference data.[16] An on-device assistant, for example, could handle most queries locally but defer to a more powerful cloud model when its confidence is low (e.g., when top predictions have low probability or it detects an out-of-domain input).

Designing the fallback logic and confidence measures for generative models remains an open challenge under the tight computing constraints of edge devices, where traditional uncertainty quantification methods (such as Monte Carlo dropout or ensemble models) may be too computationally expensive.

Building on top of these, a more dynamic approach is emerging to balance safety and efficiency: TTC (test-time compute), where the computational budget during inference is adjusted in real time based on task complexity or model confidence. Rather than statically allocating resources for every input, TTC dynamically scales computation—invoking larger submodels or engaging more complex pathways only when necessary. This approach allows edge models to allocate more compute to ambiguous or high-stakes inputs while maintaining efficiency for routine tasks. For example, an on-device assistant might process simple requests with a lightweight model but escalate to a more capable model when handling ambiguous queries or critical commands.

TTC introduces a paradigm of adaptive inference that aligns computational effort with task complexity, enabling edge deployments to achieve higher safety without sacrificing efficiency. Implementing TTC in resource-constrained settings, however, introduces new challenges, such as designing low-latency confidence estimators and ensuring that escalation thresholds are tuned to avoid unnecessary compute overhead. Despite these challenges, TTC holds promise for enabling models that dynamically balance safety and efficiency at the edge.

Alignment and Safety at the Edge

Large-scale alignment (such as RLHF) is usually done in the cloud on giant models with human feedback. How do you ensure a small edge model is safe, polite, and abides by norms? You might distill the aligned behaviors from a big model into a small one. Ouyang et al.'s finding that even a 1.3B model can be aligned to superhuman instruction-following is promising—it suggests some degree of alignment in smaller packages is possible.[17] For example, aligned models often have a calibrated refusal behavior ("I'm sorry, I can't assist with that request."). If the model is heavily compressed, will it still reliably trigger that refusal when needed, or will it produce an unsafe response?

Early evidence shows that compression can sometimes degrade moderation filters or make toxic outputs more likely if not carefully managed. Therefore, a challenge is developing metrics and tests for safety, specifically under edge constraints. We need to measure things such as "toxicity per memory footprint" or evaluate hallucination rates as a function of quantization level. The "Safety–

Efficiency Tradeoffs" section later presents some thoughts on this.

Continual Learning and Model Refresh

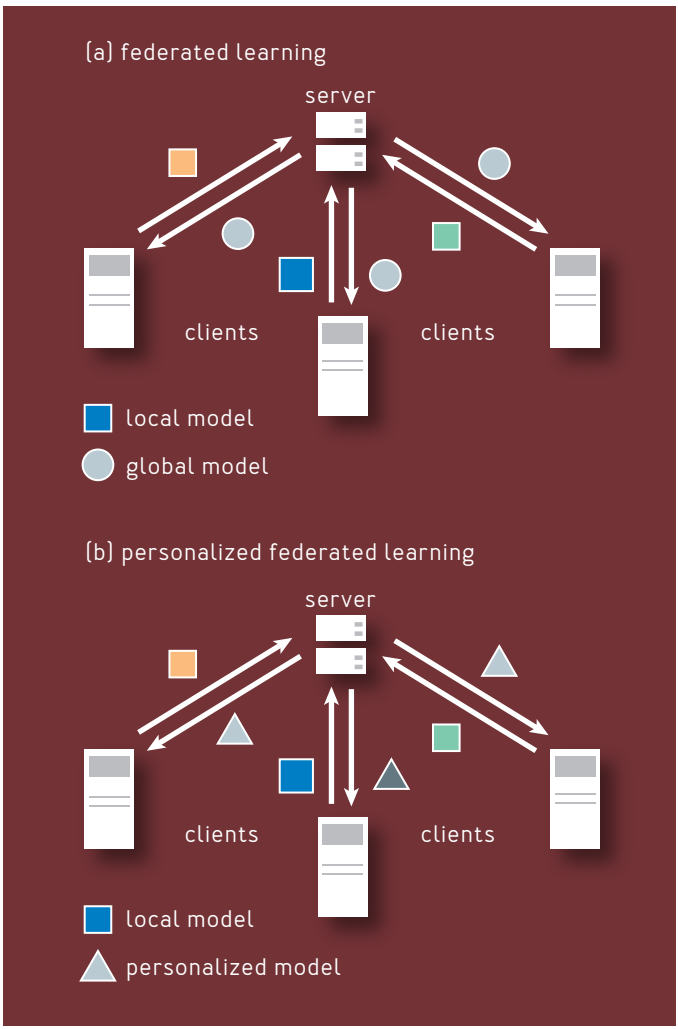
Edge models, once deployed, might not be frequently updated (unlike cloud models that can be patched or replaced centrally). This raises the issue of model staleness; over time, an on-device generative model may become outdated in its knowledge (imagine a local news summarizer that doesn't know about events in 2025 or later because it was trained in 2024). If the device is mostly offline, it can't fetch updates.

How to keep models fresh? One approach is to include a retrieval mechanism that can pull in new information (as long as it has access to some data source). Another is periodic updates when the device is connected, but rolling out frequent large model updates on billions of devices is nontrivial (also, users may not appreciate such large downloads). A related challenge is heterogeneity: In a fleet of edge devices, all models might not update at the same time or even to the same version. This could complicate FL (clients on different model versions) and create inconsistent experiences.

Figure 3 illustrates: (a) traditional FL, where clients share updates between local models and a global model maintained on the server, and (b) personalized FL, where clients maintain personalized models that adapt to individual user needs while still contributing to and benefiting from the global knowledge base. The "model" in the field becomes not a monolith but a distribution of versions. Edge-friendly algorithms might be needed that

3

FIGURE 3: **COMPARISON OF FEDERATED LEARNING APPROACHES**



can adjust to whatever model version is present (perhaps by keeping a stable API or interface that the rest of the system uses, even if internals differ).

Each device might fork its model from the base when personalizing models, making it even harder to apply a universal update without overwriting personal adaptations. Solving this likely involves decoupling the base model knowledge from personal deltas (so you can update the base safely) and using techniques such as federated distillation, where knowledge from user models is distilled into a new global model that then gets sent back out, merging updates in a privacy-preserving way.

In essence, the model corner of the triangle shown in figure 2 demands balancing compactness with capability and doing so in a way that models remain truthful, safe, and up to date. Many of these aspects are directly tied to computing considerations since solutions such as running a verifier or retrieval require extra computation. Let's turn to that corner next.

Compute constraints

Edge devices range from battery-powered IoT sensors to smartphones and vehicles, but all share limits on computation, memory, and energy. The *compute* dimension is often the most immediately constraining—if a model is too slow or too power-hungry, it simply won't run on the device at all. Key challenges include meeting latency requirements, respecting energy/thermal budgets, and leveraging hardware accelerators effectively.

Latency and Real-Time Operation

Many edge use cases have strict latency bounds (e.g., an AR translation app might need to generate text within 50ms to feel instantaneous, a car's AI must react within tens of milliseconds, a dialogue assistant should respond in under a second to feel fluid). Achieving this means the model's inference has to be optimized to the hilt. Batch processing (amortizing overhead over many requests) is usually not applicable at the edge since tasks come one at a time. This is opposite to the cloud, where batching across users boosts throughput. So, edge models must be efficient in single-instance inference. Techniques here include quantization (again, for speed on certain hardware), operator fusion and optimized graph compilers (to reduce overhead between neural network layers), and in some cases model splitting across time (e.g., running a smaller model most of the time and invoking a larger model only occasionally for a complex query (trading off quality versus latency dynamically).

Another approach is distillation for speed: Beyond compressing parameters, you can train a student to match the teacher's outputs *with fewer layers*, directly targeting a reduction in the number of sequential operations (since on a device, unlike on big GPU clusters, you can't heavily parallelize across many cores—you're often bound by sequential execution on a few cores). Certain model architectures are also more latency friendly: For example, a unidirectional Transformer has to compute attention sequentially for each generated token, which adds latency proportional to output length. Approaches such as nonautoregressive generation (which generates multiple

tokens in parallel via iterative refinement) could potentially cut latency if their accuracy can be made acceptable.

Memory and Storage

Memory is a hard limit on many devices—if a model doesn't fit in RAM (or VRAM on mobile GPU), it cannot run. Unlike cloud servers, you can't just add more memory to a phone or watch. This is why model size (parameters) is so critical. But it's not just model weights; the activation memory during inference also matters. Running a 6B parameter model at half precision requires maybe 12GB just for weights, which far exceeds typical mobile RAM. Even loading a 1B parameter model (maybe ~2GB) can strain some devices. Another issue is model storage: Storing a large model in a device's flash can be an issue, especially for apps that have to stay below certain APK (Android Package Kit) size. Compression techniques (such as weight clustering and coding) can reduce on-disk size beyond what quantization alone does, though they might require an on-device decompression at runtime. Techniques like unified compression and adaptive layer tuning, as in Edge-LLM, offer an elegant solution by jointly optimizing compute and memory use while retaining the model's core capabilities.[30]

Energy and Thermal Constraints

Edge devices often run on batteries and within tight thermal envelopes. A complex generative model that maxes out a phone's CPU/GPU for several seconds will drain battery and could overheat the device, leading to throttling. The energy per inference therefore has to be considered. Some academic works propose metrics such

as *Joules per prediction* or even *watt-hours per 1,000 tokens*. For example, an int8 quantized model might use five times less energy per query than an FP16 model on the same hardware. Recent MLPerf Power benchmarks suggest, however, that the energy efficiency of ML models is plateauing, even with hardware improvements, highlighting the diminishing returns of optimization.[25] This plateau means that further energy reductions may require not just hardware advancements, but also more aggressive model compression and architectural innovations. If a use case requires frequent generation (e.g., a running commentary in AR), the model must be extremely efficient or have hardware acceleration.

Speaking of hardware, modern phones have NPUs/DSPs that are much more power efficient for neural nets than general CPU. Using these fully is a challenge because it often requires custom model optimization (quantization to the chip's supported format, using specific ops). An edge deployment strategy should include hardware-aware model design, perhaps even searching the model architecture for one that best fits the target device's accelerator (as done in PhoneLM's design process). Thermal issues mean that even if a device can burst to do a big computation, it might not sustain it. A virtual reality headset might run a heavy model for a minute but then have to shut it down as it hits thermal limits.

Strategies to mitigate this include duty cycling (run model at intervals), offloading parts of computation to a nearby edge server (if available)—a paradigm called *split computing* where, for example, a camera feed might be preprocessed on-device, then sent to an edge server

for generation, and results returned. Split computing is a compromise when full on-device isn't possible, but it introduces network dependency, which may not be acceptable for all the reasons we started looking at edge in the first place. Still, in a local network or on-device hub scenario, it can be useful (consider a smart home with one more powerful hub that heavy models run on, serving multiple low-power sensors).

Compute-Data Intersections

It's worth noting how compute and data interplay. If you have low compute, you can't run fancy training algorithms on the device, limiting how you use data. Conversely, if data is scarce, maybe you can allocate more compute per data point to squeeze more signal out of it (such as doing a very long training run on a few examples, which might overfit, though). An example intersection is FL overhead: FL requires devices to do local training (compute intensive) and communicate updates (which can be heavy if the model is large). Techniques such as federated averaging on full LLMs are impractical on phones because of compute—hence the need for research on federated distillation or sending lighter updates (such as sending gradients of only a small adapter rather than the whole model).

Another intersection: Compound deployment constraints occur when, for example, you try to quantize a model (to meet compute limits) and then find it's now less accurate and needs more data or fine-tuning to reach acceptable performance, but you don't have that data. Or you try to fine-tune a model on device (data-model interplay) and run into memory issues (model-compute

interplay). Solving one constraint in isolation is not enough; the design must jointly satisfy data, model, and compute constraints.

To illustrate, let's revisit the triangle in figure 2: An edge solution might pick a moderately sized model (to satisfy compute), then augment it with retrieval (to compensate data/knowledge limitations) and quantize it (to further reduce compute), but that quantization could make the model's integration with retrieval less coherent or impact safety. These are called *compound deployment constraints*, where the combination of edge requirements creates new research challenges that do not appear when each factor is considered alone.

In summary, the compute corner demands efficient inference, memory optimization, and often bespoke hardware-aware implementations. Edge generative AI is essentially systems engineering: It's not just about the model's raw accuracy but about the entire pipeline of sensors → model → outputs running under tight budgets. The interplay of data, model, and compute constraints demonstrates the need for holistic solutions—a smaller model *and* a smarter way to use data *and* leveraging hardware. There is no single silver bullet.

SAFETY–EFFICIENCY TRADEOFFS

As generative models are optimized for edge deployment, there are inevitably tradeoffs between efficiency (speed, size, resource use) and safety (factual accuracy, consistency, harmlessness of outputs). This manifests as a safety–efficiency tradeoff curve—improving one often degrades the other. Understanding and quantifying this

tradeoff is crucial so neither ultrafast models that spout unreliable or harmful content, nor overly safe but bloated models that can't run on target devices are deployed.

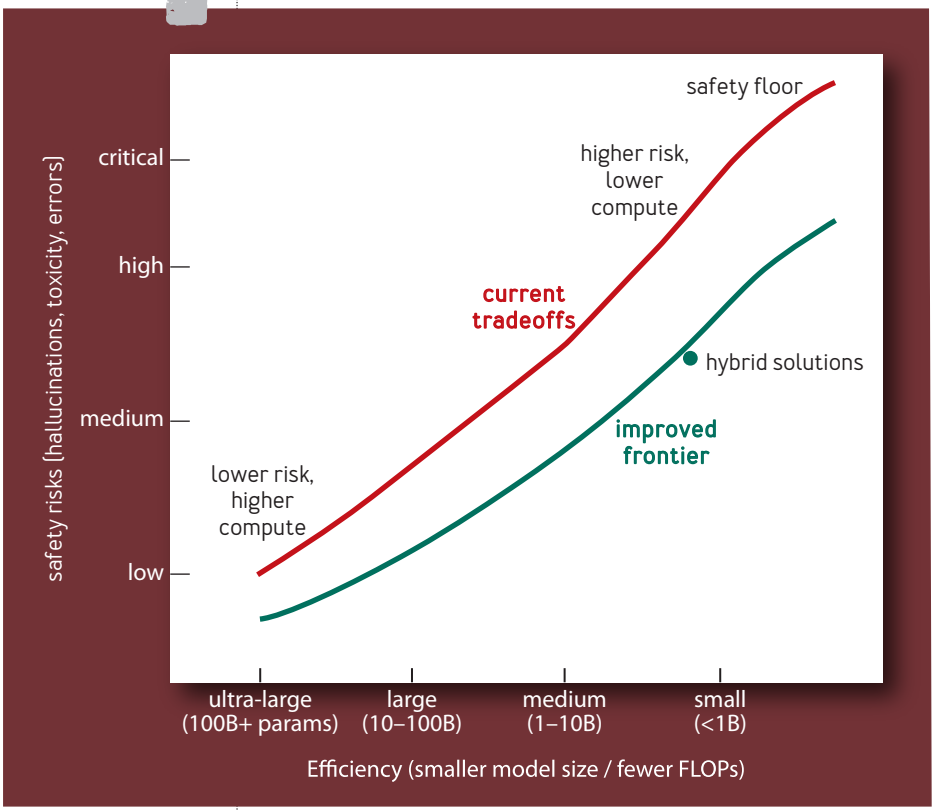
One major concern is that aggressive compression or truncation of models can erode the fine-tuned safeguards and knowledge that larger models possess. For example, a 4-bit quantized model might save memory and run faster, but subtle quantization noise could cause it to produce more toxic or biased outputs than its 16-bit counterpart, undermining safety efforts. Likewise, pruning away 30 percent of model weights might hardly reduce perplexity on evaluation, yet that pruned model might fail to refuse disallowed requests the original model would have refused.

These are not just hypothetical concerns; recent work examines the impact of different compression techniques—such as pruning and quantization—on safety metrics, including toxicity, bias, and truthfulness.[28] Their results suggest that pruning, in particular, increases perplexity more significantly than quantization, highlighting the risk of degraded model safety postcompression. The study does not compare these effects across different model sizes, however, leaving open questions about how compression interacts with model scale and whether smaller models might be more vulnerable to safety erosion during compression.

As figure 4 illustrates, this relationship can be conceptualized where the x-axis represents efficiency (model size or computational requirements) and the y-axis represents safety risks (hallucinations, toxicity, errors). Moving toward higher efficiency (leftward, fewer FLOPs or smaller model) often increases the risk of model errors or

4

FIGURE 4: **HYPOTHETICAL SAFETY VS. EFFICIENCY TRADE-OFF CURVE**



unsafe behavior (upward). The red curve indicates current tradeoffs. Techniques such as distilled alignment, selective retraining, or hybrid retrieval (green dot) could achieve safer behavior at a given level of efficiency, pushing toward the dashed green frontier. Notably, extremely small models see a sharp rise in issues such as hallucinations and toxicity, indicating a safety floor for a given model size.

A move toward more efficient models (left to right) means that safety tends to worsen (higher toxicity, more errors). For example, going from a 6B model to a 1.3B model might double the hallucination rate in a QA benchmark. Going further down to a 300M model might make it triple.

There are also discontinuities: A quantized model might have a sudden jump in errors if quantization crosses a certain threshold of precision. Our goal in research is to bend this curve upward, finding techniques that provide better efficiency without sacrificing as much safety, effectively dominating the old curve (a Pareto improvement).

Let's break down some specific safety concerns under edge constraints.

Degradation of factuality

As parameter counts drop or quantization increases, models may lose some of the stored knowledge or precision needed for factual accuracy. A bigger LLM might know obscure historical facts, whereas a smaller one might just approximate or completely fabricate an answer. If retrieval is also removed (because of offline operation), the small model is on its own. This could be mitigated by keeping a cache of verified facts on-device (a mini knowledge base to query), but then you're back to needing more memory and code for retrieval.

One promising area is robust knowledge distillation, where instead of just distilling raw outputs, the teacher forces the student to internalize facts (perhaps by generating QA pairs or true/false statements and training

the student on them). There is also work on modularizing knowledge so that a small model can consult a compressed knowledge store as needed. But then ensuring the model actually consults it and doesn't hallucinate is a challenge. Techniques such as KnowNo,[21] which use conformal prediction to align uncertainty and prompt models to defer responses when confidence is low, offer a promising way to mitigate this issue. Such approaches could reduce hallucination risks and ensure that factuality is preserved, even when models operate offline or under constrained conditions. To make progress, evaluating factuality on-device may likely require new metrics that balance accuracy with resource constraints. For example, "answers per kilojoule" could be a quirky but useful metric, combining factual accuracy with energy efficiency.

Toxicity and bias

Many large models are given safety training to avoid toxic language or biased outputs. If you compress a model heavily, you need to check if those filters still function. It could be that a pruned model inadvertently removes some neurons critical to filtering hate speech, for example. An aligned model usually has a sort of "moral compass" encoded through many subtle parameters—it's fragile to compression.

One approach is to retrain or fine-tune the compressed model on safety-specific data. For example, after quantizing, run an additional pass of RLHF or instruction tuning focusing on avoiding harmful outputs. This could realign the quantized model. However, doing RLHF on-device is infeasible; this would be done on a server and then

the new weights pushed to devices. If devices personalize models, they might also personalize in a way that deviates from safety guidelines (not intentionally, but say a user fine-tunes a model on a niche Internet forum text, it might pick up that subculture's language, which could be toxic by broader standards). Policies will be needed on how much freedom a user has to alter an on-device model's behavior—this starts bordering on user responsibility versus developer responsibility.

Recent work suggests that the order of compression and fine-tuning can significantly impact both accuracy and bias.[28] These findings indicate that pruning followed by fine-tuning tends to preserve the model's overall accuracy better, whereas fine-tuning followed by pruning results in models with lower bias. A hybrid approach—choosing the order of operations based on the desired balance between accuracy and bias—could offer a more robust solution when compressing safety-aligned models.

Robustness to distribution shift

Edge models might face input distributions that differ from their training (maybe more dialectal language or sensor noise). Large models tend to be more robust (by virtue of broad training), whereas small ones can be brittle. If an edge model encounters unexpected input, will it gracefully say, "I don't know," or will it malfunction (outputting something random or unsafe)? Often, the efficient models have less robust uncertainty calibration. Work such as HELM ("Holistic Evaluation of Language Models" by Liang et al.)[14] emphasizes evaluation on robustness and calibration, but the same is needed for edge-specific

models—possibly new benchmarks where models are evaluated under constrained settings, or after undergoing compression operations, to quantify how robustness drops. Multi-agent setups (such as an ensemble of small models) could help here—if one model is unsure, another might catch it, but running multiple models is a compute luxury edge devices may not have.

Emerging combined metrics

Since current metrics fall short, new metrics that combine safety and efficiency may be needed to guide future research. For example, using hallucinations per watt-hour, run a model on a factual QA task until it uses a fixed amount of energy (say, 1Wh) and count how many incorrect factual statements it produces. A lower number would be better (fewer hallucinations for the energy). Or use a metric such as toxicity per token per model size, basically measuring toxicity rate and normalizing by model size to see if compressing increases the per-parameter toxicity propensity.

Although somewhat contrived, such metrics force thinking in a multi-objective way. The community could consider incorporating efficiency into leaderboards; *energy vs. accuracy* plots are already present in "Green AI" discussions,[22] so extending that to *energy vs. accuracy vs. safety* would be logical. Ultimately, a safe-enough but efficient model (shown as the green dot in figure 4) might be preferred over a super-safe but huge model that can't deploy or an efficient but unsafe model that causes harm.

Navigating the safety–efficiency tradeoff will likely involve hybrid solutions (e.g., a small model augmented

by a lightweight safety layer). One idea is a *governor*, a secondary model (much smaller than the main model) that watches the output for red flags. This approach aligns with research by Ji et al.,[10] who demonstrate that analyzing LLM internal states can effectively reveal hallucination risk in responses. For example, a tiny text classifier on-device could monitor the LLM's output for hate speech or obvious lies (if fact references are available) and then either veto or modify the output. This is analogous to how some cloud systems have a separate moderation API. The governor itself would have to be efficient (maybe a simple keyword list or a small neural net). This two-model system slightly increases resource use but might achieve better safety per efficiency than trying to make one model do everything.

Acknowledging and addressing the safety–efficiency tradeoff is essential for responsible edge AI deployment. *We should ignore neither safety in the pursuit of efficiency (leading to untrustworthy models), nor efficiency assuming safety is only a cloud concern*. The next generation of benchmarks and research should evaluate models on a *joint* spectrum. Only then can we chart paths to improve both simultaneously—for example, finding that a certain quantization scheme preserves factuality better, or a certain model architecture compresses with less loss of alignment. With robust evaluation, the community can iterate toward edge models that are trustworthy and efficient by design.

FUTURE OUTLOOK

The convergence of generative AI and edge computing is opening new frontiers. As discussed here, there are

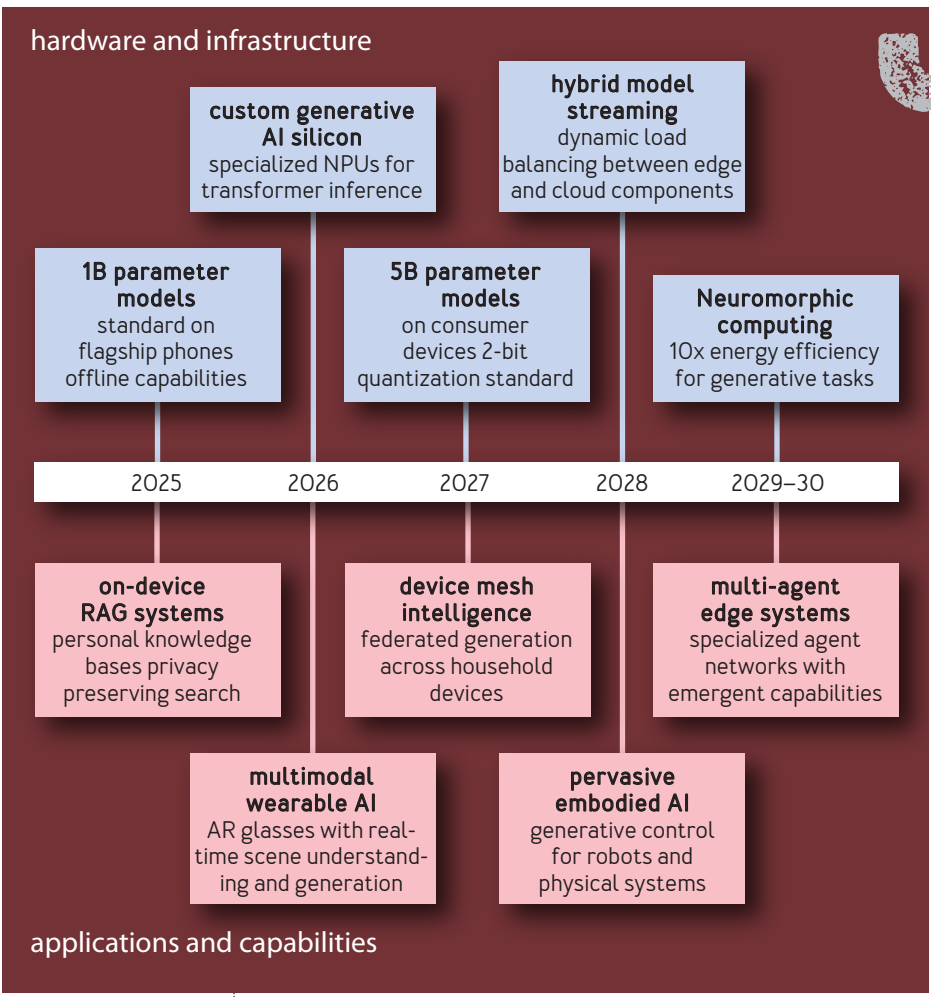
challenges but also clear momentum toward solutions. This final section outlines key trends and proposes a research agenda to realize deployable generative AI fully. The guiding vision is a world of ubiquitous, personalized, and cooperative intelligent devices powered by what might be called *embedded foundation models* that encapsulate useful knowledge and skills yet are lightweight and safe enough to embed in everyday tech.

Figure 5 presents a projected timeline for edge generative AI evolution over the next five years, mapping both the hardware advances and the application capabilities that will likely emerge. This roadmap highlights how fundamental research in areas such as model compression, specialized hardware, and FL may translate into practical capabilities that progressively address the challenges outlined in the data-model-compute triangle.

Toward specialized small models by design

Thus far, many small models are obtained by compressing bigger ones. An emerging trend is designing SLMs from scratch with edge deployment in mind. The PhoneLM project exemplifies this: It performs an architecture search to find a Transformer variant that runs optimally on phone hardware, then pretrains it.[29] Google's Gemma 3 is another successful implementation of this philosophy. Intentionally designed for single-GPU and constrained usage environments, these models demonstrate that purpose-built small models can achieve impressive results. Notably, Gemma 3 outperforms even older versions of larger Gemini models, which highlights how the performance

FIGURE 5: **PROJECTED EVOLUTION OF AI CAPABILITIES AND UNDERLYING HARDWARE (2025–2030)**



gap between resource-efficient models and their larger counterparts is rapidly narrowing.

This flips the traditional paradigm—usually we train, then optimize; but here we optimize the design first. Pushing forward on this front, we can expect more research in NAS (neural architecture search) for generative models under resource constraints. There may also be algorithmic breakthroughs—for example, new attention mechanisms or token representation schemes that are less memory hungry (work on linear or block-sparse attention could allow longer contexts on device).

Another direction is model modularity: Instead of one giant model that does everything, multiple smaller components each handle parts of the task (one for language understanding, one for factual recall, one for style, etc.), orchestrated on device. This modular approach could align with how operating systems schedule processes (e.g., run only the vision module when needed and park it otherwise to save energy). Research into *interface protocols* between such modules (so they can share context efficiently) would be valuable.

Federated generation and learning

In scenarios such as distributed sensor networks or user communities, we can envision federated generative AI, where multiple devices collaboratively train or even *generate* content. FL is already studied for discriminative models; for generative models, there are extra wrinkles. For example, if 100 phones collectively train a language model, how do you ensure it doesn't overfit to one user's style or leak another user's phrases to another? There's

work on federated GANs and VAEs, but federated LLMs is a nascent area.

Beyond training, think of *federated generation.* Could devices share partial results to generate something jointly? For example, in a multiplayer AR game, each user's device might generate part of a story or environment, which needs consistency. A concept introduced here is federated generation, where generation is distributed across nodes. One simple case is a chain: Device A starts a story, device B continues it, etc., all locally without a central server. This requires the model (or models) to maintain coherence and possibly a shared latent state. New algorithms for synchronization of generative state across devices would be needed. Another case is voting or consensus: Multiple devices, each running a local model, aggregate their outputs (e.g., a swarm of drones each imagines an optimal path, then they agree on one). This intersects with multi-agent systems.

Multi-agent and collaborative SLMs

Multi-agent AI often refers to separate AI entities interacting. Here, multi-agent SLMs are considered to be collections of small models that communicate to solve tasks. Instead of one model trying to do everything (which might require it to be large), there could be a team of five small models, each an expert in something, chatting among themselves (on the device or over a local network) to produce an outcome. Recent work on *generative agents*—AI agents that simulate humanlike behavior and dialogue in a sandbox environment—has shown that even with large models, interesting emergent behaviors arise

when multiple agents interact.[18]

Translating that to small models on edge, you could imagine, for example, a household with various devices, each running an agent: The fridge has a meal-planning agent, the fitness band has a health coach agent, and together they coordinate via a local dialogue to recommend your diet for the day. This is somewhat futuristic, but it aligns with the vision of ambient computing, where intelligence is distributed. Research questions include: How do you ensure consistency between agents? How do you prevent compounding errors (one agent believing another's hallucination)? What communication protocols (perhaps a minimalist language or data format) should they use to be efficient?

Embodied and physical world foundation models

As discussed in robotics, there's interest in RFMs (robotic foundation models) that encode skills for interacting with the world. Future *embodied SLMs* might integrate language, vision, and action in a small footprint model for a home robot or an automotive assistant. The term *embodied SLM* is meant to emphasize small models with sensorimotor capabilities. These models will likely need to be trained with simulation and real-world data and must handle continuous control outputs (a departure from discrete token generation). Efforts such as Microsoft's Phi-1 model (a 1.3B LLM) showed some emergent capabilities with proper training.[30] Integrating classical control knowledge, such as PID (proportional-integral-derivative) controllers or motion planners, into the network (perhaps as differentiable modules) could yield safe yet

flexible behavior for embodied use. A research agenda here is combining model-based control with model-free generative policies in a resource-efficient way.

Trustworthiness and ethics

As edge generative AI becomes commonplace, ensuring trustworthy AI is paramount. This spans robustness (not crashing or misbehaving because of minor input perturbations); transparency (some ability to explain why the model produced an output, which is hard for generative models but maybe approximate explanations can be given); and user control (allowing users to set preferences or limits on the AI's behavior). For example, a user might want an edge assistant to avoid certain topics; implementing that locally might involve a user-editable filter list that the model's decoder respects. The research community should also study the *social impact* of widespread on-device generative AI: Does it amplify echo chambers because each person's model becomes highly personalized to their viewpoints? Does it reduce reliance on verified sources (since the model can speak confidently even if wrong)? Are there positive effects, such as improved accessibility and empowerment for users without Internet connectivity? These questions will shape how the technology is designed and deployed. Multidisciplinary research with social scientists and user experience experts will be valuable, as was done for analyzing large foundation models; the same is needed for edge foundation models.

Sustainability as a future imperative

As generative AI transitions toward multi-agent

collaboration and federated generation in edge environments, sustainability becomes a central concern. Although edge inference can reduce the operational energy footprint by eliminating cloud communication overhead, the embodied carbon cost of deploying billions of intelligent edge devices remains a challenge. Achieving sustainability in generative AI will require innovations in hardware efficiency, life-cycle management, and responsible edge deployment strategies. By 2030, embedded foundation models must not only be adaptive and trustworthy, but also sustainable across their entire lifecycle—from manufacturing to real-time inference. Addressing these challenges will require collaborative efforts to develop standardized sustainability benchmarks and ensure that future AI systems balance performance with long-term environmental impact.

Standardizing benchmarks and metrics

Unlike classification or retrieval, generative AI is inherently more challenging to evaluate, because it produces open-ended outputs rather than discrete predictions. This complexity becomes even more pronounced at the edge, where constraints on latency, memory, and energy intersect with safety, privacy, and personalization requirements. Just as MLPerf has provided a foundation for benchmarking AI performance in datacenter, mobile, and edge contexts,[2] a new generation of benchmarks is needed to capture the nuanced demands of generative models deployed on edge devices. These benchmarks should reflect practical tasks, such as real-time summarization, low-resolution image captioning, and

privacy-aware conversational agents, while incorporating multidimensional metrics that account for quality, latency, energy use, and safety risks such as hallucination or toxicity. Standardizing evaluation in this space is critical not only for comparing models but also for ensuring responsible and efficient deployment of generative AI in real-world settings.

Future edge AI benchmarks must go beyond traditional performance metrics, however. Sustainability and energy efficiency should become core dimensions of evaluation. As generative AI deployments at the edge scale to billions of devices, understanding the carbon footprint of model inference, the impact of local adaptation, and the life-cycle emissions of edge devices will be essential for responsible AI deployment. Benchmarks should also encourage participants to submit systems, not just models, ensuring that runtime optimization and hardware usage improvements are factored into overall scores. Refining metrics such as hallucinations per watt-hour can further align benchmarks with sustainability goals, encouraging innovation that balances safety, efficiency, and environmental impact.

CALL TO ACTION

As generative AI models evolve from centralized cloud services to adaptive, multi-agent systems operating at the edge, the tradeoffs between safety, efficiency, and performance will grow more complex. The innovations discussed—MoE, test-time compute, FL, multi-agent collaboration, and embodied AI—are critical milestones in this journey. By 2030 and beyond, we can anticipate

the emergence of embedded foundation models that seamlessly balance these tradeoffs, delivering context-aware, efficient, and trustworthy AI experiences across diverse edge environments.

But building these systems—and making that vision a reality—requires engineers and researchers who are deeply fluent in the complexities of machine-learning infrastructure. Much of the future innovation in AI hinges on not just algorithms or data, but also the ability to design, optimize, and deploy intelligent systems at scale. To that end, the next generation must be trained to understand machine learning systems as a full-fledged engineering discipline.[19,20]

CONCLUSION

Generative AI at the edge is the next phase in AI's deployment: from centralized supercomputers to ubiquitous assistants and creators operating alongside humans. The challenges are significant but so are the opportunities for personalization, privacy, and innovation. By tackling the technical hurdles and establishing new frameworks (conceptual and infrastructural), we can ensure this transition is successful and beneficial. The coming years will likely see *embodied, federated,* and *cooperative* small models become commonplace, quietly working to enhance our lives in the background, much as embedded microcontrollers did in the previous tech generation. The difference is, these models won't just *compute;* they will *communicate, create,* and *adapt*. It's up to us, the researchers and engineers, to pave the way for

this deployable intelligence and shape it with the values of efficiency, safety, and trustworthiness from the ground up.

Acknowledgments

I would like to express my gratitude to my students Andy Cheng, Jason Jabbour, Jeffrey Ma, Mark Mazumder, Arya Tschand, and Ike Uchendu for their valuable assistance and insightful feedback.

References

1. Bahdanau, D., et al. (2014). *Neural machine translation by jointly learning to align and translate*. arXiv preprint arXiv:1409.0473.
2. Banbury, C., et al. (2021). *MLPerf Tiny benchmark*. In NeurIPS Datasets and Benchmarks Track.
3. Bommasani, R., et al. (2021). *On the opportunities and risks of foundation models*. Stanford CRFM Report. https://crfm.stanford.edu/report.html
4. Brohan, A., et al. (2022). *RT-1: Robotics Transformer for real-world control at scale*. arXiv preprint arXiv:2212.06817.
5. Dettmers, T., et al. (2022). *LLM.int8(): 8-bit matrix multiplication for transformers at scale*. In NeurIPS 2022. https://dl.acm.org/doi/10.5555/3600270.3602468
6. Driess, D., et al. (2023). *PaLM-E: An embodied multimodal language model*. In Proceedings of the 40th ICML, 8469–8488. https://dl.acm.org/doi/10.5555/3618408.3618748
7. Guu, K., et al. (2020). *REALM: Retrieval-Augmented Language Model pre-training*. In Proceedings of the 37th ICML. https://dl.acm.org/doi/10.5555/3524938.3525306

8. Hinton, G., et al. (2015). *Distilling the knowledge in a neural network*. arXiv preprint arXiv:1503.02531.
9. Javed, S., et al. (2024). *QT-DoG: Quantization-aware training for domain generalization*. arXiv preprint arXiv:2410.06020.
10. Ji, Z., et al. (2024). *LLM internal states reveal hallucination risk faced with a query*. arXiv preprint arXiv:2407.03282.
11. Konecny, J., et al. (2016). *Federated learning: Strategies for improving communication efficiency*. arXiv preprint arXiv:1610.05492.
12. Lan, Z., et al. (2020). *ALBERT: A lite BERT for self-supervised learning of language representations*. In ICLR 2020. https://iclr.cc/virtual_2020/poster_H1eA7AEtvS. html
13. Lewis, P., et al. (2020). *Retrieval-augmented generation for knowledge-intensive NLP tasks*. In NeurIPS 2020. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html
14. Liang, P., et al. (2022). *Holistic evaluation of language models (HELM)*. arXiv preprint arXiv:2211.09110.
15. Nissen, L., et al. (2025). *Medicine on the edge: Comparative performance of on-device LLMs for clinical reasoning*. arXiv preprint arXiv:2502.08954.
16. Ong, I., et al. (2024). *RouteLLM: Learning to route LLMs from preference data*. In ICLR 2024. https://arxiv.org/abs/2406.18665
17. Ouyang, L., et al. (2022). *Training language models to follow instructions with human feedback*. In NeurIPS 2022, 27730–27744. https://dl.acm.org/doi/10.5555/3600270.3602281

18. Park, J. S., et al. (2023). *Generative agents: Interactive simulacra of human behavior*. In UIST 2023, Article 2, 1–22. https://dl.acm.org/doi/10.1145/3586183.3606763
19. Reddi, V. J. (2024). *MLSysBook.AI: Principles and practices of machine learning systems engineering*. In CODES+ISSS, 41–42. IEEE.
20. Reddi, V. J. (2025). *Machine Learning Systems*. https://mlsysbook.ai
21. Ren, A. Z., et al. (2023). *Robots that ask for help: Uncertainty alignment for large language model planners*. arXiv preprint arXiv:2307.01928.
22. Schwartz, R., et al. (2020). *Green AI. Communications of the ACM*, 63(12), 54–63. https://cacm.acm.org/research/green-ai/
23. Sun, Z., et al. (2020). *MobileBERT: A compact task-agnostic BERT for resource-limited devices*. ACL 2020. https://aclanthology.org/2020.acl-main.195/
24. Sutskever, I., et al. (2014). *Sequence to sequence learning with neural networks*. In NeurIPS 2014, 3104–3112. https://dl.acm.org/doi/10.5555/2969033.2969173
25. Tschand, A., et al. (2025). *MLPerf Power: Benchmarking the energy efficiency of machine learning systems from microwatts to megawatts for sustainable AI*. In HPCA 2025. https://arxiv.org/abs/2410.12032
26. Vaswani, A., et al. (2017). *Attention is all you need*. In NeurIPS 2017. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
27. Xu, M., et al. (2024). *Unleashing the power of edge-cloud generative AI in mobile networks: A survey of AIGC services*. IEEE Communications Surveys and

Tutorials, 26(2), 1127–1170. https://dl.acm.org/doi/10.1109/COMST.2024.3353265
28. Xu, Z., et al. (2024). *Beyond perplexity: Multi-dimensional safety evaluation of LLM compression*. arXiv preprint arXiv:2407.04965.
29. Yi, R., et al. (2024). *PhoneLM: An efficient and capable small language model family through principled pre-training*. arXiv preprint arXiv:2411.05046.
30. Yu, Z., et al. (2024). *Edge-LLM: Enabling efficient large language model adaptation on edge devices via unified compression and adaptive layer voting*. Proceedings of the 61st ACM/IEEE Design Automation Conference, 1–6. https://doi.org/10.48550/arXiv.2406.15758.

Dr. Vijay Janapa Reddi *is an associate professor at Harvard University, where he works at the intersection of computer architecture, machine learning systems, and autonomous agents. He is vice president and cofounder of MLCommons, where he helps lead MLPerf and directs MLCommons Research. His work spans efficient computing systems from edge devices to large-scale ML infrastructure, with real-world impact through initiatives like the TinyML edX massive open online course (MOOC) series and the open-source* Machine Learning Systems *textbook (*mlsysbook.ai*). Dr. Janapa Reddi has received numerous honors, including the IEEE TCCA Young Computer Architect Award, multiple IEEE Top Picks, and induction into the MICRO and HPCA Halls of Fame.*

CONTENTS