Electronic Design

KARTHIKEYAN SHANMUGA VADIVEL, Director of AI Architecture,
Technology and Innovation Group, Synaptics Inc.

Moving Small Language Models Out to the Edge

Learn how hardware and software optimizations are making small language models fast, efficient, and practical enough to run directly on edge devices.

The rapid adoption of large language models (LLMs) has reset the bar for intelligent user interfaces, natural-language control, and contextual assistance. While much of the attention has focused LLMs that run in the cloud, a growing number of embedded and industrial systems require AI inference to run locally on the edge. Privacy concerns, regulatory requirements, network reliability, latency constraints, and operating costs are all driving up demand for small language models, or SLMs, that can run on device.

The challenge is that most LLMs have been developed to run inside data centers with abundant compute resources, memory bandwidth, and power budgets. Bringing these AI workloads to the edge, where compute, memory, and energy resources are far more constrained, requires a fundamentally different approach to both hardware architecture and software optimization.

However, researchers are rising to the challenge of shrinking the compute, memory, and power requirements of GenAI. For instance, Google Research and Synaptics recently showed Google's compact Gemma 3 model with 270 million parameters running directly on Synaptics' Coralboard development platform.

Why Smaller Models aren't Enough for Edge AI

Compact models such as Gemma 3 270M are specifically designed to bring the power of LLMs into resource-constrained environments. With approximately 270 million parameters and 18 transformer layers, the SLM is small enough to fit within embedded memory budgets while still supporting conversational interfaces, tool invocation, summarization, translation, and other natural-language tasks.

Transformer architectures rely heavily on attention mechanisms, large matrix multiplications, activation functions, and growing key-value caches and sequence lengths. These characteristics create bottlenecks that are often poorly matched to conventional embedded processors.

As a result, developers frequently discover that simply porting a language model to an embedded CPU produces unacceptable latency, excessive power consumption, or both. Beyond raw performance, running AI workloads on the host CPU also consumes compute capacity needed for the rest of the application stack.

As a result, developers are forced to choose between AI functionality and system responsiveness. This builds a strong case for offloading transformer workloads to dedicated hardware engines and freeing the host processor for control logic and application tasks.

Three Main Bottlenecks in Edge LLM Inference

1. Dynamic Execution Behavior

Traditional embedded accelerators perform best when tensor shapes and execution paths remain fixed. But LLMs introduce a different challenge.

As conversations grow, key-value (KV) caches expand, attention masks change, and sequence lengths evolve dynamically. Inference frameworks running in the data center can accommodate these changes more readily because they operate on highly flexible hardware and software stacks. Edge accelerators, however, often depend on static execution graphs for efficiency. This mismatch between dynamic model behavior and static hardware scheduling can significantly reduce accelerator utilization.

One emerging solution involves transforming dynamic execution graphs into statically scheduled representations during model compilation. By pre-allocating memory structures and converting runtime decisions into compile-time optimizations, developers can achieve more deterministic execution while maintaining model functionality.

2. Activation Function Overhead

A second challenge comes from the complexity of the mathematical operations used throughout transformer networks. Functions such as GELU (Gaussian Error Linear Unit) and softmax appear repeatedly within transformer

layers. Although these functions are computationally manageable in cloud environments, they become surprisingly expensive on embedded platforms because they require exponentiation, division, and other complex operations.

In many edge deployments, activation functions consume a disproportionate share of inference time and energy.

But more efficient approaches are emerging. In Synaptics' implementation with Google, for example, complex activation functions are approximated using hardware-optimized lookup tables (LUTs) and linear interpolation, avoiding repeated exponentials, divisions, and other computationally expensive operations.

These approaches illustrate a broader principle of edge AI design: Maximizing useful inference throughput often matters more than maintaining mathematically exact implementations of every operation.

3. Memory Bandwidth Constraints

A major limitation in LLM deployment at the edge is memory bandwidth.

For transformer inference, memory bandwidth frequently becomes a larger bottleneck than raw compute performance. Large weight matrices must be repeatedly transferred between memory and compute units, causing accelerators to sit idle while waiting for data.

This phenomenon is particularly important in embedded systems, where power-efficient memory subsystems are often preferred over high-bandwidth alternatives. Consequently, many edge AI implementations focus on reducing data movement rather than simply increasing arithmetic throughput.

Quantization: How It Helps Overcome the Memory Bottleneck

One of the most effective techniques for overcoming memory bottlenecks is quantization.

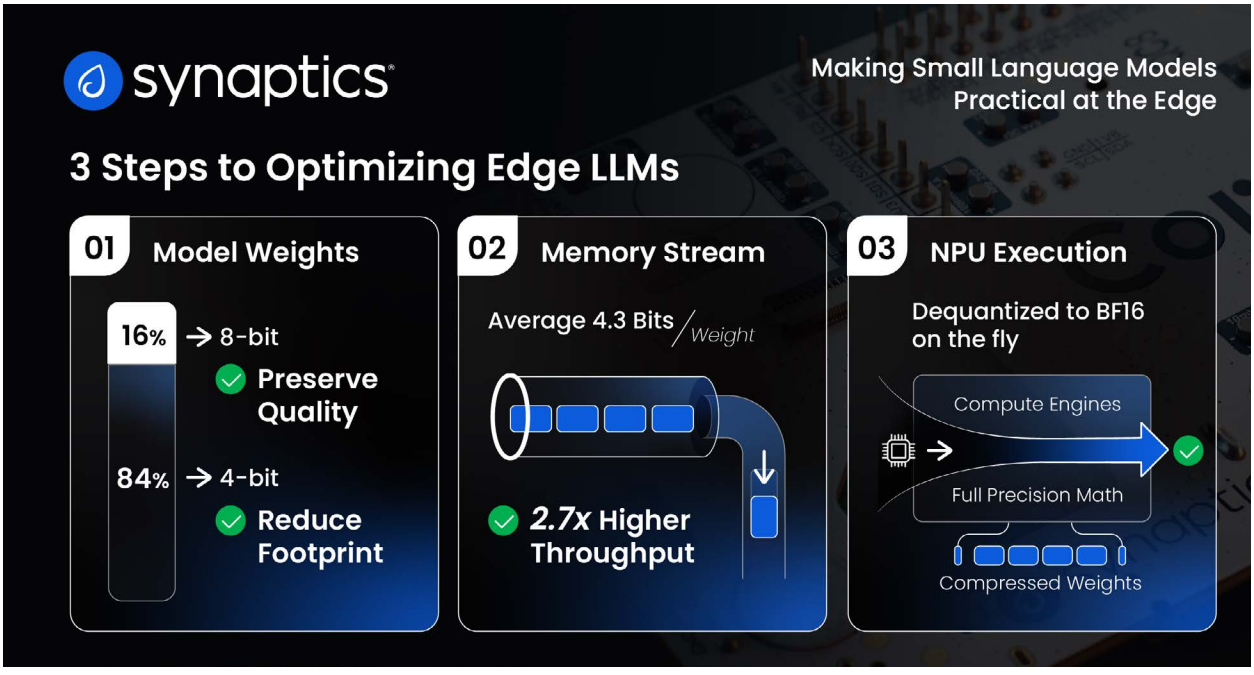
Quantization reduces the number of bits required to represent model weights and activations. Rather than storing parameters in 16-bit floating-point formats, developers can compress portions of a model to 8-bit, 4-bit, or even lower-precision representations.

However, not all model layers respond equally well to aggressive compression. One approach to this problem is sensitivity-guided, mixed-precision weight quantization in which model weights are assigned bit widths according to each layer's tolerance for reduced precision *(Fig. 1)*.

In Synaptics' implementation, 84% of layers are compressed to 4-bit precision, while the remaining 16% — including the language modeling head — are kept at 8-bit to preserve output quality. The result is an average bit width of about 4.3 bits across the full model, with weights dequantized on the fly back to BF16 as they stream into the compute units. This approach significantly reduces memory traffic and storage requirements with negligible loss in model fidelity.

For smaller language models such as Gemma 3 270M, mixed-precision quantization can make the difference be-

1. Mixed-precision quantization reduces edge LLM memory and bandwidth demands while preserving model quality, helping improve inference throughput on resource-constrained devices.



2. Heterogenous edge AI enables faster local response, reduced cloud dependence, and greater control over data by running AI workloads closer to where data is created.

tween a model that barely fits on an edge device and one that operates efficiently in real-time. Compressed weight representation has delivered 2.7X higher effective throughput, which is a meaningful multiplier when bandwidth is the primary constraint.

The Rise of Heterogeneous AI Architectures

These optimization techniques are driving a broader shift toward heterogeneous edge-computing architectures *(Fig. 2)*.

Rather than relying exclusively on CPUs or a single type of accelerator, edge AI systems increasingly distribute workloads across several specialized processing engines. Convolutional-neural-network (CNN) operations, transformer layers, control logic, signal processing, and sensor fusion may each execute on different hardware blocks optimized for their specific requirements.

The collaboration between Synaptics and Google illustrates the trend. The recently introduced Coralboard is powered by the Synaptics' Astra SL2619 SoC and its heterogeneous Torq neural processing unit (NPU) subsystem, which brings together a transformer-capable T1 core and Google Research's RISC-V-based Coral NPU.

The chip can execute Gemma 3 270M locally at the same time as computer vision workloads and multimodal applications. By combining static graph compilation, LUT-based activation approximations, and mixed-precision quantization, it can increase Gemma 3 270M inference speed by as much as 3.5X.

While the specific implementation is vendor-dependent, the principle underpinning the architecture is increasingly common around the industry: Match each portion of the AI workload to the best processing resource available.

Beyond Chatbots: Practical Edge AI Applications

The significance of local language-model execution extends well beyond conversational interfaces.

Many embedded applications use language models primarily as orchestration engines rather than content generators. Natural-language commands can be translated into local actions, API calls, or device controls without requiring cloud connectivity.

Potential applications include:

- Industrial equipment interfaces
- Smart-home automation
- Edge-based voice assistants
- Local document summarization
- On-device translation systems
- Human-machine interfaces for robotics

In these situations, local execution offers more than a performance boost. Keeping inference on the device can improve privacy by reducing the need to transmit sensitive data to external servers. It may also reduce recurring cloud costs and enable operation where connectivity is intermittent or unavailable.

Regulatory considerations are becoming increasingly important as well. Requirements related to cybersecurity, data sovereignty, and operational resilience are encouraging designers to minimize dependence on remote AI infrastructure where practical. Europe's Cyber Resilience Act (CRA), for example, establishes lifecycle cybersecurity requirements for products with digital elements. On-device inference doesn't establish compliance, but reducing unnecessary data transfers can support a broader security and resilience strategy.

The Future of AI at the Edge

The emergence of models such as Gemma 3 270M signals an important transition in the AI industry. Rather than scaling exclusively toward larger and more computationally intensive models, developers are increasingly focusing on compact models that can be specialized for specific tasks and deliver useful capabilities within realistic power, memory, and cost constraints.

Success in this environment depends on far more than model compression alone. Efficient edge AI implementation requires coordinated optimization across software toolchains, compiler technology, memory architecture, quantization strategies, and accelerator design.

The ecosystem is evolving quickly. Platforms such as the Coralboard illustrate one possible path forward. However, the larger industry lesson is that practical edge AI will be enabled by heterogeneous architectures and workload-specific optimizations that address the unique demands of transformer inference.

As generative AI expands from the cloud to include more endpoint devices, cloud resources will remain important for training, orchestration, analytics, and fleet management. The most successful embedded implementations are likely to complement those resources by treating memory movement, execution efficiency, and hardware specialization as first-order design considerations. That shift may ultimately prove as important as the language models themselves.

Karthikeyan Shanmuga Vadivel is Director of AI Architecture in the Technology and Innovation group at Synaptics Inc., focusing on model and compiler optimization efforts for the company's edge AI initiatives. Since joining Synaptics, he has been instrumental in advancing machine learning and biometric technologies across multiple product lines. Karthikeyan began his career at Synaptics following the completion of his PhD in 2014 from the University of California, Santa Barbara, where he built a strong publication record in leading computer vision conferences, including CVPR, ICCV, and ECCV. At Synaptics, he developed the industry-leading Quantum Matcher for capacitive fingerprint sensors and subsequently played a key role in the development of the world's first optical fingerprint sensor, leading image processing and matching innovations.