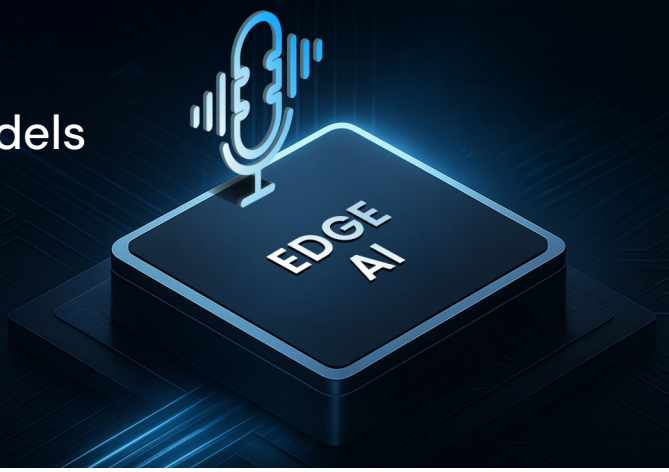
Running Extreme Low-Bit Models on IoT Edge Devices

ENERZAi 1.58bit Speech Recognition Model on the Synaptics Astra SL1680



Introduction

Recent efforts relating to on-device AI have extended well beyond traditional CNN-based vision models to encompass Generative-AI-driven Large Language Models (LLMs). Running such models locally, however, is challenging: LLMs are significantly larger in comparison with CNNs, while edge devices face tight constraints on CPU/GPU cycles, DRAM capacity and bandwidth, power budget, and real-time responsiveness. In the cloud, dozens of gigabytes of memory and many parallel accelerators are readily available; on the edge, by contrast, limited memory and I/O bandwidth make data movement a primary performance bottleneck. Edge devices typically juggle multiple tasks rather than dedicating all resources to a single AI workload, making aggressive model compression techniques and system-level optimizations mandatory.

These constraints have spurred research into sub-4-bit quantization, yet extreme low-bit models have so far seen little real-world adoption. The main barrier has not been due to model accuracy. Even when Post-Training Quantization (PTQ) or Quantization-Aware Training (QAT) has delivered excellent results, acceptance has lagged due to the lack of inference backends capable of efficiently executing such ultra-low-bit formats.

This document presents ENERZAi's 1.58-bit QAT version of OpenAI's Whisper Speech-to-Text model, compiled using the ENERZAi Optimium inference engine as the backend, and deployed on the Synaptics Astra™ Machina (SL1680) platform. The effort highlights extremely efficient edge AI performance through ultra-low-bit quantization and optimized inferencing technology.

Methodology

1.58-bit Quantization

Quantization is a technique that shrinks a neural-network model and its compute footprint by converting weights and activations from high-precision numbers to lower-precision representations. Training and inference are usually performed in 32-bit floating point, but mapping those values to 8-bit integers (int8) or even fewer bits cuts memory usage, speeds up computation, and reduces power consumption—benefits that are particularly valuable for memory-bound LLMs.

Quantization methods fall into two broad categories. Post-Training Quantization (PTQ) applies quantization after training has finished; it is quick to implement but can incur larger accuracy loss. Quantization-Aware Training (QAT) embeds the quantization effects into the training loop, allowing the network to compensate for quantization error and thus preserve accuracy.

1.58-bit quantization is an extreme low-bit scheme that approximates each weight with one of three values {-1, 0, 1}. Whereas int8 and int4 use 256 and 16 discrete levels, respectively, 1.58-bit quantization relies on only three. The resulting compression is maximal, and the arithmetic collapses to simple sign-based operations, easing hardware implementation. The "1.58-bits" label is information-theoretic—$\log_2(3) \approx 1.58$—rather than a literal bit width.

In this work we applied QAT to achieve 1.58-bit quantization. This approach is far more precise than PTQ because the model learns to operate under quantized constraints during training. As confirmed experimentally, PTQ can hold accuracy down to roughly 4 bits, but at 2 bits and below its performance deteriorates sharply. The degradation occurs because the extreme low-precision weights fail to capture the original weight distribution, pushing the solution away from the true optimum. QAT, by contrast, continuously adjusts the parameters while accounting for quantized weights and activations, minimizing this loss. For formats as restrictive as 1.58 bits, QAT is effectively the only way to obtain usable accuracy—an insight we leverage to make SL1680 deployment practical.

Optimium

Running AI models on edge devices requires a dedicated inference engine. Frameworks such as PyTorch and TensorFlow are optimized for training; they pull in numerous third-party dependencies, consume large amounts of memory, and rely on a full Python runtime—making them unusable or highly inefficient on mobile and embedded targets. In production, therefore, developers turn to lightweight, inference-only runtimes such as TensorFlow Lite, which compress trained networks and apply execution-time optimizations so that models can run reliably under tight memory and compute budgets.

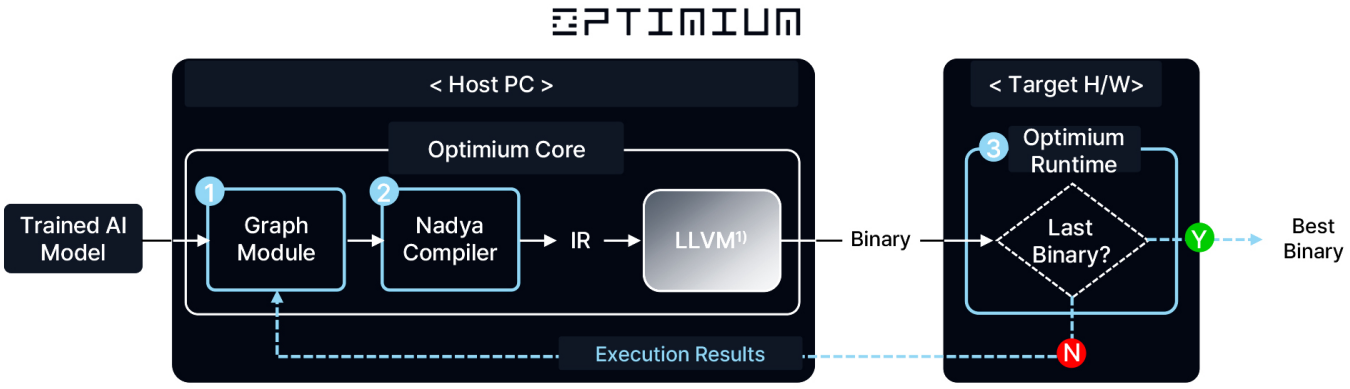


Figure 1: Overview of Optimium pipeline

In this work we deploy the high-performance Optimium runtime from ENERZAi, to execute a 1.58-bit-quantized model directly on the Synaptics Astra platform. Optimium ingests models exported from PyTorch or TensorFlow Lite, performs extensive operator fusion and graph-level optimizations, and emits a target-specific, highly optimized output. It is built around ENERZAi Nadya, a domain-specific language, and its MLIR-based compiler stack that produces the model as a shared library (.so). Nadya's metaprogramming facilities allow the code generator to adapt dynamically to the performance profile of the target device, yielding optimal execution speed.

Even among Arm® edge devices that share a standard CPU core subsystem (ex. Cortex-A73), SIMD support (FP16, NEON, etc.), memory bandwidth, and cache hierarchy can vary widely by vendor. Hand-tuning kernels for every variant is impractical. Optimium's abstraction layer and automatic code-generation pipeline make the same kernel portable across heterogeneous hardware, while an internal explore-exploit search algorithm selects the fastest variant from a large pool of candidates. Crucially, the 1.58-bit kernels used in this work are unsupported by other runtime options, yet Optimium's flexible architecture let us define, compile, and deploy them with ease. The result is a platform capable of highly optimized inference—even under the severe constraints typical of edge devices.

Results

The target device 1.58-bit QAT was applied to Whisper, the transformer-based Large Language Model (LLM) designed to transcribe spoken language into text, and available in various sizes—such as large, medium, small, base, and tiny(Table 1).

Model Version by Size	Number of Parameters
Tiny	39M
Base	74M
Small	244M
Medium	769M
Large-v3	1550M

Table 1: Various Whisper versions by size(Source: https://huggingface.co/openai/whisper-large-v3)

Among these, the Whisper small model was chosen as the primary target due to its strong balance between performance and efficiency. It represents a practical upper bound for on-device speech recognition within embedded environments. As speech recognition is one of the most prominent real-world use cases for edge AI, Whisper's high accuracy and transformer architecture make it a compelling candidate for quantization and model compression.

The target device was the Synaptics Astra SL1680 IoT AI processor, which features a quad-core ARM Cortex-A73 processor. For comparison, we used three model variants: the baseline float16 Whisper small, a 4-bit PTQ model, and the 1.58-bit QAT model proposed in this work. As for the float16 and 4-bit benchmark models, inference was executed using whisper.cpp, the most widely adopted backend for running Whisper on edge devices. The 1.58-bit QAT model was trained on approximately 40,000 hours of speech data, using public datasets such as LibriSpeech and Common Voice, and utilized the Optimium backend.

Model accuracy was evaluated using Word Error Rate (WER) on the LibriSpeech dataset. WER measures the rate of incorrect word transcriptions; even a single word mistake is counted as an error, so lower values indicate better performance. Experimental results showed that our 1.58-bit QAT model incurred only

approximately 0.3% degradation in terms of WER, a difference small enough to be imperceptible in real-world use. These findings confirm that QAT-based extreme quantization can significantly reduce model size while maintaining high transcription accuracy, making it highly suitable for practical on-device deployment.

Model	WER(%)
Whisper Small FP16	5.99
Whisper Small Q4	5.86
1.58-bit Whisper	**6.38**

Table 2: WER comparison using LibriSpeech test dataset

Below are the profiling results of inference performance on SL1680 CPU(Cortex-A73) for the Whisper small model under different quantization schemes. The experiment used a 9-second audio input. Peak memory usage during inference was measured using the Linux `time` command, and latency was calculated with the C++ `chrono` library.

Model	Peak Memory(MB)	Latency(s)
Whisper Small FP16	629	16.98
Whisper Small Q4	309	9.38
1.58-bit Whisper	**143**	**6.91**

Table 3: Comparison of peak memory and latency for each model type

The 1.58-bit QAT model achieved up to a 4x reduction in memory usage compared to the float16 baseline and nearly 2x improvement in latency. While the 4-bit PTQ model also delivered meaningful latency improvements, it still consumed more than twice the memory of the 1.58-bit model.

In on-device AI environments, multiple AI workloads often need to run in parallel, and most devices operate under strict memory constraints—typically just a few gigabytes or less. As such, memory and performance optimizations have a direct impact on system stability and the end-user experience.

While the Whisper model is available in various sizes, including the smaller base variant, our experiments revealed that the float16 base model was less memory-efficient than the small model, and the 4-bit PTQ base model exhibited excessively high WER, making it unsuitable for real-world applications. These findings validate the effectiveness of the 1.58-bit QAT-based Whisper small model proposed in this work, as it offers an optimal balance between accuracy and performance, making it a compelling solution for practical on-device AI.

Model	Peak Memory(MB)	Latency(s)
Whisper Base FP16	225	7.53
Whisper Base Q4	132	8.25

Table 4: Peak memory and WER of FP16 & Q4 Whisper Base models

Conclusion

It's meaningful to enable extreme low-bit STT models on the edge, as speech functions could be widely used as a powerful modality for a broad set of Edge AI applications. However, this marks only the first step of many more 1.58-bit projects we expect to become available, considering high-performance, AI-native edge hardware platforms like the Synaptics Astra SL1680.

Especially, SL1680 is a SoC comprised of stable and balanced CPU, GPU and NPU subsystems which means it could not only provide diverse scenarios in terms of AI model utilization but also could enable heterogeneous execution scenarios to maximize Edge AI performance. Furthermore, we can also consider splitting a single model into pieces to allocate some operations to CPU after Optimium-optimization and the rest to NPU or GPU for optimum latency and throughput. Additionally, the NPU can be utilized in parallel to run vision or NLU models alongside other tasks, enabling seamless end-to-end AI applications on the edge. For instance, while the CPU handles real-time speech recognition or lightweight control logic, the NPU can simultaneously execute a hand gesture recognition model or intent classification model, powering use cases such as multimodal human-machine interfaces or smart assistants. This parallelism empowers developers to build responsive, resource-efficient systems by fully tapping into the heterogeneous compute potential of the SL1680 SoC.

This convergence of powerful, reliable hardware platforms like the Synaptics Astra SL1680 with ENERZAi's advanced inference optimization technology underscores how the combined strengths unlock AI potential at the edge.

ENERZAi is an AI solution provider specializing in inference optimization utilizing the custom-built, high-performance 'Optimium' inference engine. ENERZAi provides extreme low-bit models built to run efficiently on resource-constrained edge devices.

Discover the full potential of Synaptics Astra AI-native compute solutions.