

Zen3-Nano: A Compact-Deployment Finetune of Qwen3-8B

Technical Whitepaper v2025.06

Zach Kelling
Zen LM Research Team
research@zenlm.org
papers.zenlm.org

June 2025

Abstract

Zen3-Nano is a packaging- and deployment-focused member of the Zen family. Despite the “Nano” name, the released Zen3-Nano weights are *not* a 0.6B-parameter from-scratch model: configuration fingerprinting shows an 8B-class dense transformer matching **Qwen3-8B** [1], the open-weight model released by Alibaba’s Qwen team under the Apache-2.0 license. Earlier versions of this whitepaper described a bespoke “Zen MoDE (Mixture of Distilled Experts)” architecture, a 0.6B parameter count, a from-scratch 2-trillion-token pretraining run, and a hierarchical-distillation pipeline from “Zen3-Mini/Small/Medium” teachers. Those claims do not match the shipped artifact and have been removed. This corrected whitepaper documents the actual Qwen3-8B base, the light instruction tuning and quantization-for-deployment work the Zen team performs, and defers to the upstream Qwen3 technical report for pretraining and benchmark numbers. Quantized GGUF/MLX builds are provided for convenient deployment; the genuine “small” tiers of the Qwen3 family (0.6B, 1.7B, 4B) are noted as the appropriate bases for true edge use.

Contents

1	Introduction	2
2	Architecture (Inherited from Qwen3-8B)	2
2.1	Grouped-Query Attention	3
2.2	Feed-Forward Network	3
3	Post-Training and Quantization	3
3.1	Light Instruction Tuning (Optional)	3
3.2	Quantization for Deployment	3
4	Evaluation	4
5	Related Work	4
6	Deployment and Runtime	4
6.1	Context Length	4
6.2	API Compatibility	4
7	Limitations	5
8	Conclusion	5

1 Introduction

On-device and resource-constrained inference is an important deployment target, and quantization makes even 8B-class models broadly deployable. This whitepaper documents Zen3-Nano honestly, correcting prior inaccurate claims about its size and origin.

Provenance and corrections. The released Zen3-Nano weights fingerprint to **Qwen3-8B**: 36 layers, hidden size 4096, 32 query heads / 8 KV heads (GQA), head dimension 128, FFN intermediate size 12,288, and a 151,936-token vocabulary [2]. This is an 8.2B-parameter dense model released by the Qwen team at Alibaba Cloud under the Apache-2.0 license [1]. The following earlier claims were inaccurate and are withdrawn:

- “0.6B parameters” / “sub-250MB” / “third-generation 0.6B edge model” — the shipped model is 8B-class, not 0.6B.
- “Zen MoDE (Mixture of Distilled Experts)” architecture — Qwen3-8B is a standard dense transformer, not a mixture-of-experts model; there is no “MoDE” routing.
- “Trained from scratch on a 2T-token corpus” and “hierarchical distillation from Zen3-Mini/Small/Medium teachers” — no such from-scratch pretraining or teacher cascade was performed by the Zen team; the base is the released Qwen3-8B.
- Specific benchmark figures (e.g., MMLU 47.3%, GSM8K 52.4%, and the per-hardware tokens/second table) — these were fabricated and are removed.

What Zen3-Nano actually is. Zen3-Nano is a lightly instruction-tuned and quantized repackaging of Qwen3-8B aimed at easy local deployment. Its contributions are (i) optional light SFT for formatting/instruction style on top of Qwen3-8B, and (ii) ready-to-run quantized artifacts (GGUF for llama.cpp, MLX for Apple Silicon, ONNX). For genuinely tiny edge footprints, the Qwen3 family also offers true small dense models—Qwen3-0.6B, Qwen3-1.7B, and Qwen3-4B (all Apache-2.0) [1]—which are the appropriate bases when sub-1B size is a hard requirement.

2 Architecture (Inherited from Qwen3-8B)

Zen3-Nano inherits the Qwen3-8B architecture unchanged: a dense decoder-only transformer with grouped-query attention, SwiGLU feed-forward blocks, RMSNorm, and rotary position embeddings. There is no mixture-of-experts component.

Table 1: Architecture hyperparameters, inherited unchanged from Qwen3-8B [2].

Hyperparameter	Value
Parameters (total)	8.2B
Non-embedding parameters	6.95B
Layers	36
Attention heads (query)	32
KV heads (GQA)	8
Head dimension	128
Hidden dimension	4096
FFN intermediate dimension	12,288
Vocabulary size	151,936
Context length (native)	40,960
Context length (with YaRN)	131,072
Position encoding	RoPE ($\theta = 1,000,000$)
Activation function	SiLU (SwiGLU)
Normalization	RMSNorm
Tied embeddings	No

2.1 Grouped-Query Attention

Qwen3-8B uses GQA [5] with 32 query heads grouped onto 8 key-value heads (head dimension 128), reducing KV-cache memory relative to full multi-head attention. Rotary positional embeddings (RoPE) [6] are used with base frequency $\theta = 1,000,000$; long context up to 131,072 tokens is available via YaRN scaling [8] as configured upstream.

2.2 Feed-Forward Network

Each layer uses a SwiGLU feed-forward block [7]:

$$\text{FFN}(\mathbf{x}) = (\text{SiLU}(\mathbf{x}\mathbf{W}_{\text{gate}}) \odot \mathbf{x}\mathbf{W}_{\text{up}}) \mathbf{W}_{\text{down}} \quad (1)$$

with intermediate dimension 12,288. There is no expert routing; the previously described “MoDE($\mathbf{x}$) = $\sum_i g_i(\mathbf{x})E_i(\mathbf{x})$ ” formulation does not apply to this model.

3 Post-Training and Quantization

The Zen team does not pretrain Zen3-Nano. Pretraining of the underlying Qwen3-8B was performed by the Qwen team; see the upstream technical report [1] for the corpus, scale, and procedure. Our work on top of the released weights is limited to:

3.1 Light Instruction Tuning (Optional)

Where a chat-tuned variant is shipped, we apply light supervised fine-tuning on curated instruction-response data (response-masked loss, low learning rate, few epochs) to adjust formatting and instruction-following style. This does not constitute pretraining and does not materially change the base model’s knowledge.

3.2 Quantization for Deployment

The deployment artifacts use standard post-training quantization (e.g., llama.cpp’s Q4_K_M and Q8_0 schemes, and MLX 4-bit) applied to the released weights. We do not perform the

“quantization-aware training from epoch 1” described in earlier versions of this whitepaper; the shipped artifact is the released Qwen3-8B quantized post hoc with widely used GGUF/MLX recipes. References to a proprietary “BitDelta (ZIP-007)” quantization-aware objective have been removed as they were not used to produce this model.

4 Evaluation

On benchmark numbers. This whitepaper no longer reports fabricated benchmark tables or per-device throughput figures. Because Zen3-Nano is the Qwen3-8B model (optionally lightly tuned and quantized), its capabilities are those of Qwen3-8B, subject to the usual accuracy cost of 4-bit quantization. For rigorous, independently reproducible results on Qwen3 models (MMLU, GSM8K, MATH, HumanEval, multilingual, and long-context), see the official Qwen3 technical report [1] and the Qwen3-8B model card [2]. Any quantization-vs-BF16 quality deltas we publish are measured on the released GGUF/MLX artifacts under a stated harness; we do not quote numbers we have not measured.

Right-sizing for edge. If a true sub-1B footprint is required, the appropriate choice is a genuinely small Qwen3 base (0.6B/1.7B/4B), not an 8B model relabeled “Nano.” Published Qwen3 results for those tiers [1] should be used to set expectations rather than the fabricated figures previously printed here.

5 Related Work

Knowledge distillation [3] and teacher-assistant cascades [4] are well-established techniques for producing smaller models; however, Zen3-Nano as shipped is not the product of such a pipeline—it is the Qwen3-8B base, optionally lightly tuned and quantized. Post-training quantization for efficient deployment is standard practice (e.g., the GGUF schemes used by llama.cpp). The underlying base model belongs to the Qwen3 series [1], which provides a coherent family of dense (0.6B–32B) and MoE models under Apache-2.0.

6 Deployment and Runtime

Because Zen3-Nano is architecturally Qwen3-8B, it runs on any Qwen3-compatible runtime:

- **llama.cpp** — GGUF format (Q4_K_M, Q8_0).
- **MLX** — Apple Silicon, 4-bit.
- **ONNX Runtime** — ARM and x86 SIMD.
- **Hugging Face Transformers / vLLM** — BF16 reference weights.

6.1 Context Length

Native context is 40,960 tokens (Qwen3-8B default), extensible to 131,072 tokens via YaRN at the cost of additional KV-cache memory. Memory footprint depends on quantization: an 8B model in 4-bit requires on the order of several gigabytes, which is larger than the sub-250MB figure incorrectly cited in earlier versions.

6.2 API Compatibility

When served via a Qwen3-compatible inference server, Zen3-Nano exposes an OpenAI-compatible API, enabling drop-in replacement for OpenAI SDK clients.

7 Limitations

Zen3-Nano inherits the limitations of Qwen3-8B and of autoregressive models generally: factual hallucination, degraded performance on long multi-step reasoning, and incomplete safety alignment. Post-training quantization to 4-bit introduces an additional, measurable accuracy cost relative to BF16. The “Nano” name is a legacy product label and does not indicate a sub-1B model; deployments with hard memory limits should select a genuinely small Qwen3 base instead.

8 Conclusion

Zen3-Nano, as released, is the Apache-2.0 Qwen3-8B model—optionally lightly instruction-tuned and quantized for convenient local deployment—not a 0.6B from-scratch “Zen MoDE” model. This corrected whitepaper documents the inherited Qwen3-8B architecture, the limited post-training and standard quantization actually performed, and the correct guidance to use a true small Qwen3 base (0.6B/1.7B/4B) when a sub-1B edge footprint is required. We defer to the upstream Qwen3 technical report for pretraining details and rigorous, attributed benchmark numbers.

Acknowledgments

We thank the Qwen team at Alibaba Cloud for releasing the Qwen3 models under the Apache-2.0 license, and the maintainers of llama.cpp and MLX for the quantization and runtime tooling used to produce the deployment artifacts.

References

- [1] Qwen Team, Alibaba Cloud, “Qwen3 Technical Report,” *arXiv:2505.09388*, 2025.
- [2] Qwen Team, “Qwen3-8B model card and configuration,” Hugging Face, <https://huggingface.co/Qwen/Qwen3-8B>, 2025.
- [3] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network,” *NIPS Deep Learning Workshop*, 2015.
- [4] S. I. Mirzadeh et al., “Improved Knowledge Distillation via Teacher Assistant,” *AAAI*, 2020.
- [5] J. Ainslie et al., “GQA: Training Generalized Multi-Query Transformer Models,” *EMNLP*, 2023.
- [6] J. Su et al., “RoFormer: Enhanced Transformer with Rotary Position Embedding,” *arXiv:2104.09864*, 2021.
- [7] N. Shazeer, “GLU Variants Improve Transformer,” *arXiv:2002.05202*, 2020.
- [8] B. Peng et al., “YaRN: Efficient Context Window Extension of Large Language Models,” *arXiv:2309.00071*, 2023.