

The Zen AI Model Family

An Honest Overview: Finetunes and Packaging of Open Models

Technical Overview and Provenance Whitepaper v2.0

Zach Kelling*

Hanzo Industries Lux Industries Zoo Labs Foundation
`research@lux.network`

September 2025

Abstract

The **Zen AI Model Family** is a curated set of models for language, vision, and speech that are *packaged and lightly fine-tuned derivatives of open, permissively licensed upstream models*—predominantly Alibaba’s **Qwen3** series (Apache-2.0), with a few members based on other open releases (e.g., DeepSeek and IBM Granite). Zen is not a from-scratch model program: we do not run large-scale pretraining, and we do not use a proprietary “Zen MoDE” architecture. Earlier versions of this overview claimed homegrown architectures, parameter counts (such as a “480B” code model and “235B” design models), order-of-magnitude efficiency advantages (“10× smaller,” “up to 98% less energy”), and per-model benchmark and energy tables. Those claims were not substantiated by the shipped artifacts and have been removed. This corrected overview states, for each Zen model, the real upstream base and its license, and defers to the upstream technical reports for pretraining details and benchmark numbers. The value Zen provides is selection, instruction/format tuning, and convenient multi-format deployment artifacts (SafeTensors, GGUF, MLX), all under the upstream licenses with attribution.

Contents

1	Introduction	3
1.1	What Zen Is—and Is Not	3
1.2	Corrections to Prior Versions	3
2	Model Family: Bases and Licenses	3
2.1	Provenance Table	3
2.2	The Qwen3 Base Series (for reference)	4
3	Architecture (Inherited)	4
4	What the Zen Team Actually Does	5
4.1	Selection and Curation	5
4.2	Light Instruction / Format Tuning	5
4.3	Packaging and Deployment Artifacts	5
5	Training Methodology (Upstream)	5
6	Evaluation and Benchmarks	5

*Corresponding author: zach@lux.network

7	Deployment and Integration	6
7.1	Deployment Formats	6
7.2	Hardware Footprint	6
7.3	Python Integration	6
7.4	REST API	6
8	Licensing and Attribution	7
9	Safety and Alignment	7
10	Limitations	7
11	Conclusion	7
A	Model Availability	7
B	Citation	8

1 Introduction

The cost of pretraining a competitive foundation model from scratch is out of reach for most teams. The Zen family takes the pragmatic, now-standard route: start from strong, permissively licensed open base models and add value through selection, light instruction/format tuning, and convenient deployment packaging. This overview documents that honestly.

1.1 What Zen Is—and Is Not

- **Is:** a curated catalogue of derivatives of open upstream models (mostly Qwen3, Apache-2.0), repackaged with deployment artifacts and, in some cases, light fine-tuning.
- **Is not:** a from-scratch pretraining effort, a novel “Zen MoDE” architecture, or a set of models with independently verified order-of-magnitude efficiency or accuracy advantages over their own upstreams.

1.2 Corrections to Prior Versions

The following previously published claims were inaccurate and are withdrawn:

- A proprietary “Zen MoDE (Mixture of Distilled Experts)” architecture — the underlying models are standard Qwen3 (dense and Qwen3-30B-A3B MoE) and other open architectures.
- Specific homegrown parameter counts including a “480B” code model and “235B” design models — these do not correspond to the shipped bases.
- “Performance comparable to models 10× their size” and “reducing energy consumption by up to 98%” — unsubstantiated marketing claims; removed.
- Per-model benchmark tables, environmental-impact (energy/CO₂/water) tables, and per-device throughput figures — these were fabricated and are removed in favor of pointers to upstream-reported, attributed numbers.

2 Model Family: Bases and Licenses

2.1 Provenance Table

Table 1 lists each Zen language model with its real upstream base and license. Zen language models are predominantly Qwen3 forks; a small number are based on other open models. All listed Qwen3 bases are released by Alibaba Cloud under Apache-2.0 [1, 2].

Zen Model	Upstream Base	License	Notes
Zen-Nano	Qwen3-0.6B	Apache-2.0	Smallest dense tier; true edge size
Zen-Eco	Qwen3-4B	Apache-2.0	Consumer/laptop tier
Zen (base)	Qwen3-8B	Apache-2.0	Default instruction model
Zen-Pro	Qwen3-8B	Apache-2.0	Reasoning-tuned (long-CoT/RLVR)
Zen-Omni	Qwen3-30B-A3B (MoE)	Apache-2.0	30.5B total / 3.3B active MoE
Zen-Next	Qwen3-32B	Apache-2.0	Largest dense Qwen3 tier
Zen (DeepSeek tier)	DeepSeek (e.g. V-series)	per upstream	Non-Qwen base; verify upstream license
Zen (Granite tier)	IBM Granite	Apache-2.0	Non-Qwen base (IBM Granite)

Table 1: Zen language models and their real upstream bases. Parameter counts, native context, and benchmark numbers are those of the named upstream; see the corresponding upstream technical report. Non-Qwen tiers must carry the license of their specific upstream base.

On vision, image, and speech members. Zen also ships vision/image and speech models. Those are likewise derivatives of open upstreams (for example, Qwen vision-language and ASR models, and permissively licensed image/video generators) and are documented in their own whitepapers with explicit base-model attribution. A prior license audit found that several Zen media releases had been mislabeled relative to their true upstreams; those have been corrected or removed in the model repositories. This overview focuses on the language family; consult each model’s individual card and whitepaper for authoritative per-model provenance and license.

2.2 The Qwen3 Base Series (for reference)

For context, the upstream Qwen3 series [1] provides open-weight dense models at 0.6B, 1.7B, 4B, 8B, 14B, and 32B, plus two mixture-of-experts models, Qwen3-30B-A3B (30.5B total / 3.3B active) and Qwen3-235B-A22B (235B total / 22B active). All are released under Apache-2.0. The Zen language tiers select from the smaller end of this series (0.6B/4B/8B/32B dense and the 30B-A3B MoE); Zen does not ship a from-scratch model larger than its chosen Qwen3 bases.

3 Architecture (Inherited)

Zen language models inherit the architecture of their Qwen3 bases without modification. In summary, the dense Qwen3 models are decoder-only transformers with:

- Grouped-query attention (GQA) [4];
- SwiGLU feed-forward blocks;
- RMSNorm normalization;
- Rotary position embeddings (RoPE), with long context via YaRN [5];
- A 151,936-token multilingual BPE vocabulary.

Qwen3-30B-A3B additionally uses a sparse mixture-of-experts feed-forward layer (128 experts, 8 activated). There is no proprietary “MoDE” routing; the only MoE in the Zen language family is the upstream Qwen3-30B-A3B design. Per-model hyperparameters (layer counts, hidden sizes, head configurations, native context) are exactly those published in the Qwen3 technical report and model cards [1, 2].

4 What the Zen Team Actually Does

4.1 Selection and Curation

We choose an appropriate open base for each tier (size, license, capability) and document the choice.

4.2 Light Instruction / Format Tuning

For chat-tuned variants we may apply light supervised fine-tuning (response-masked loss, low learning rate, few epochs) and, optionally, preference optimization (DPO) [12] to adjust instruction-following style. This does not constitute pretraining and does not materially change the base model’s knowledge. Where reasoning tuning is applied (e.g., Zen-Pro), it consists of long chain-of-thought SFT and optional reinforcement learning from verifiable rewards on math/code.

4.3 Packaging and Deployment Artifacts

The principal deliverable is convenient, multi-format deployment. We publish, per model:

- BF16 SafeTensors reference weights;
- GGUF quantizations (e.g., Q4_K_M, Q8_0) for llama.cpp on CPU/GPU;
- MLX 4-bit builds for Apple Silicon.

Quantization uses standard, widely used post-training schemes applied to the released weights. Any quality cost of quantization relative to BF16 is measured on the released artifacts and reported alongside, rather than asserted.

5 Training Methodology (Upstream)

Large-scale pretraining of the underlying models was performed by the upstream teams (Alibaba’s Qwen team for Qwen3; the respective teams for any non-Qwen bases). The Zen team does not run pretraining. For the pretraining corpus, scale, data composition, and pretraining procedure, see the upstream technical reports—principally the Qwen3 technical report [1]. Post-training that the Zen team performs is limited to the light fine-tuning described above and is documented per model.

6 Evaluation and Benchmarks

No fabricated tables. This overview deliberately does not reproduce per-model benchmark tables. The capabilities of each Zen model are, to first order, those of its upstream base. For rigorous, independently reproducible benchmark results—MMLU, GSM8K, MATH, HumanEval, multilingual evaluation, long-context retrieval, and reasoning suites—we refer the reader to the upstream technical reports and model cards, principally the Qwen3 technical report [1] and the per-model Qwen3 cards [2]. Where a specific Zen release reports numbers, they are measured on the released artifact under a stated harness and presented alongside the corresponding upstream baseline so that the effect of any Zen fine-tuning is transparent.

No fabricated efficiency claims. The previously stated efficiency advantages (“10× smaller,” “up to 98% energy reduction”) and the environmental-impact tables were not derived from measurements and are removed. The genuine efficiency story is simply that smaller models in the family (e.g., the 0.6B/4B tiers) are cheaper to run than larger ones, and that 4-bit quantization

reduces memory and increases throughput at a measurable accuracy cost—facts that hold for the upstream models as well.

7 Deployment and Integration

7.1 Deployment Formats

Format	Typical Precision	Platform / Runtime
SafeTensors	BF16/FP16	Hugging Face Transformers, vLLM (reference weights)
GGUF	Q4_K_M, Q8_0	llama.cpp (CPU/GPU)
MLX	4-bit	Apple Silicon
ONNX	INT8	Cross-platform runtimes

Table 2: Deployment formats shipped for Zen language models. Memory footprint and throughput depend on the base model size, quantization, and hardware; we report measured figures per artifact rather than fixed percentages.

7.2 Hardware Footprint

Memory requirements scale with the base model size and quantization in the usual way (roughly ~ 2 bytes/parameter in BF16, ~ 0.5 bytes/parameter at 4-bit, plus KV cache). For example, an 8B-class model (Zen / Zen-Pro) runs in BF16 on a single 24 GB GPU and in 4-bit on commodity hardware; the 0.6B tier (Zen-Nano) is genuinely small enough for edge devices. We do not publish a fixed memory table here because the values are simply those implied by the chosen Qwen3 base size.

7.3 Python Integration

```
# Zen models load via the standard Transformers API,
# because they are Qwen3-architecture models.
from transformers import AutoModelForCausalLM, AutoTokenizer

model = AutoModelForCausalLM.from_pretrained("zenlm/zen-eco-4b-instruct")
tokenizer = AutoTokenizer.from_pretrained("zenlm/zen-eco-4b-instruct")

messages = [{"role": "user", "content": "Solve_this_problem_step_by_step."}]
inputs = tokenizer.apply_chat_template(messages, return_tensors="pt")
outputs = model.generate(inputs, max_new_tokens=2000)
```

7.4 REST API

```
# Serve via any Qwen3-compatible OpenAI-style server (e.g., vLLM).
# Query with an OpenAI-compatible request:
curl -X POST http://localhost:8080/v1/completions \
  -H "Content-Type: application/json" \
  -d '{"model": "zen-eco-4b-instruct", "prompt": "Hello , world!", "max_tokens": 100}'
```

8 Licensing and Attribution

Most Zen language models derive from Qwen3, released by Alibaba Cloud under Apache-2.0, which permits commercial use, modification, and redistribution subject to attribution and the terms of the license. Zen distributes these derivatives under the same Apache-2.0 terms with the required upstream attribution and NOTICE, and records the `base_model` for each release. **Non-Qwen members must carry the license of their specific upstream base** (for example, IBM Granite is Apache-2.0; DeepSeek-based members must follow the applicable DeepSeek model license). Relabeling upstream weights does not extinguish upstream license terms; each model card states its authoritative license. Users should consult the upstream model cards and LICENSE files for definitive terms.

9 Safety and Alignment

Zen models inherit the safety properties and limitations of their upstream bases. Any additional safety behavior comes from the light instruction/preference tuning described above; we do not claim a distinct “Constitutional AI” training program or specific red-teaming hour counts that were not performed. Safety alignment is incomplete, as it is for the upstream models, and adversarial prompts can elicit policy-violating outputs.

10 Limitations

Because Zen models are derivatives, they inherit the limitations of their upstreams: factual hallucination, degraded performance on long multi-step reasoning, incomplete safety alignment, and the accuracy cost of low-bit quantization. Light fine-tuning can also regress some base-model capabilities; any claimed improvement should be verified against the upstream baseline. The Zen family does not provide capabilities beyond what its chosen open bases provide.

11 Conclusion

The Zen AI Model Family is best understood as a curated, well-packaged distribution of open upstream models—predominantly Apache-2.0 Qwen3, with a few non-Qwen members—rather than a from-scratch model program. Its value lies in selection, light instruction/format tuning, and convenient multi-format deployment artifacts, all under the upstream licenses with attribution. This corrected overview removes the prior fabricated architecture, parameter-count, efficiency, and benchmark claims and points readers to the upstream technical reports for authoritative details. Honest provenance is, we believe, more useful to adopters than inflated marketing.

Acknowledgments

We thank the upstream teams whose open models make the Zen family possible—principally the Qwen team at Alibaba Cloud for the Apache-2.0 Qwen3 series—and the open-source communities behind Transformers, llama.cpp/GGML, and MLX.

A Model Availability

Zen models and their per-model cards (with `base_model` attribution and license) are available at:

- HuggingFace: <https://huggingface.co/zenlm>

- **GitHub:** <https://github.com/zenlm/zen>
- **Documentation:** <https://docs.hanzo.ai/zen>

B Citation

```
@misc{zenfamily2025,
  title = {The Zen AI Model Family: An Honest Overview of Finetunes
    and Packaging of Open Models},
  author = {Zen LM Research Team (Hanzo / Lux / Zoo Labs)},
  note = {Derivatives of open upstream models, principally
    Apache-2.0 Qwen3 (Alibaba Cloud)},
  year = {2025}
}
```

References

- [1] Qwen Team, Alibaba Cloud, “Qwen3 Technical Report,” *arXiv:2505.09388*, 2025.
- [2] Qwen Team, “Qwen3 model cards and configurations,” Hugging Face, <https://huggingface.co/Qwen>, 2025.
- [3] A. Vaswani et al., “Attention is All You Need,” *NeurIPS*, 2017.
- [4] J. Ainslie et al., “GQA: Training Generalized Multi-Query Transformer Models,” *EMNLP*, 2023.
- [5] B. Peng et al., “YaRN: Efficient Context Window Extension of Large Language Models,” *arXiv:2309.00071*, 2023.
- [6] N. Shazeer et al., “Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer,” *ICLR*, 2017.
- [7] E. J. Hu et al., “LoRA: Low-Rank Adaptation of Large Language Models,” *arXiv:2106.09685*, 2021.
- [8] T. Dettmers et al., “QLoRA: Efficient Finetuning of Quantized LLMs,” *NeurIPS*, 2023.
- [9] E. Frantar et al., “GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers,” *arXiv:2210.17323*, 2022.
- [10] L. Ouyang et al., “Training language models to follow instructions with human feedback,” *NeurIPS*, 2022.
- [11] R. Rafailov et al., “Direct Preference Optimization,” *NeurIPS*, 2023.
- [12] R. Rafailov et al., “Direct Preference Optimization: Your Language Model is Secretly a Reward Model,” *NeurIPS*, 2023.