Zen AI Model Family

Zen4

Fourth-Generation Language Model

Technical Whitepaper v2026.01

Zach Kelling*
research@hanzo.ai

Zoo Labs Foundation
foundation@zoolabs.org

January 2026

Abstract

We present **Zen4**, our fourth-generation language model at 14B parameters, built on a novel hybrid attention architecture that combines local sliding-window attention with global cross-attention to handle sequences of up to 1 million tokens at constant memory. Zen4 introduces native function-calling support integrated directly into the model's pretraining objective, eliminating the need for post-hoc prompt engineering for tool use. Compared to Zen3, Zen4 achieves a 40% reduction in FLOPs per token at equivalent quality, reaching 85.4% on MMLU, 81.2% on MATH, 88.3% on HumanEval, 67.4% on GPQA, and 83.1% on IFEval. Zen4 represents a fundamental architectural leap enabling frontier capability in a deployment-practical parameter count.

Contents

1 Introduction **3**
1.1 Contributions . . . 3
1.2 Relation to the Zen Family . . . 3

2 Architecture **4**
2.1 Overview . . . 4
2.2 Hierarchical Hybrid Attention . . . 4
2.3 Grouped Query Attention . . . 4
2.4 Native Tool Pretraining . . . 5
2.5 Distilled Fine-Tuning . . . 5

3 Training **5**
3.1 Pretraining Data . . . 5
3.2 Training Infrastructure . . . 6
3.3 Post-Training . . . 6

4 Evaluation **6**
4.1 Standard Benchmarks . . . 6
4.2 Context Length Evaluation . . . 7
4.3 Tool-Use Evaluation . . . 7
4.4 Efficiency Comparison . . . 7

*zach@lux.network

5 Related Work **7**

6 Deployment **8**
6.1 Supported Formats 8
6.2 Hardware Requirements 8
6.3 API Usage 8

7 Safety and Alignment **9**
7.1 Evaluation 9
7.2 Limitations 9

8 Conclusion **9**

A Model Card **10**

B Benchmark Methodology **10**

1 Introduction

The third generation of the Zen model family established that dense transformer architectures with careful training recipes could compete with models several times their size. Zen4 advances this trajectory by addressing the two primary limitations of Zen3: context length and tool-use fidelity.

Prior frontier models have extended context through positional interpolation or sparse-attention approximations, typically at the cost of performance on standard benchmarks. Zen4 instead introduces a **Hierarchical Hybrid Attention** (HHA) mechanism that partitions tokens into local windows processed with full attention and global anchor tokens processed with cross-attention, achieving $O(n \cdot w + n \cdot k)$ complexity where w is the local window size and k the number of anchor tokens. This yields constant memory consumption as context grows, with no degradation on short-context tasks.

Tool use in prior models was grafted on through supervised fine-tuning on tool-call datasets. Zen4 treats tool calls as first-class tokens in pretraining, giving the model a deep statistical prior for when and how to invoke external functions rather than a surface-level imitation signal.

1.1 Contributions

- **Hierarchical Hybrid Attention (HHA)**: 1M token context at $O(n)$ memory, matching full-attention quality on benchmarks through 100K tokens.
- **Native Tool Pretraining**: Function-call tokens included in pretraining corpus; 20-point improvement in complex multi-turn tool-use evaluations versus SFT-only baseline.
- **Distilled Fine-Tuning**: Fourth-generation distilled fine-tuning of a Mixture-of-Experts base produces 14B model with capability profile exceeding prior 30B+ dense models.
- **40% Efficiency Gain**: Architectural and training improvements reduce FLOPs/token by 40% relative to Zen3 at matched quality thresholds.

1.2 Relation to the Zen Family

Zen4 occupies the standard tier of the Zen4 generation alongside Zen4-Mini (4B), Zen4-Pro (72B), Zen4-Max (480B MoE), Zen4-Ultra (1T MoE), and Zen4-Thinking (72B extended reasoning). Each tier shares core architectural principles while optimizing for different deployment constraints.

2 Architecture

2.1 Overview

Component	Specification
Total Parameters	14.3B
Active Parameters	14.3B (dense)
Architecture	Hybrid Transformer
Attention Mechanism	Hierarchical Hybrid Attention (HHA)
Local Window Size	4,096 tokens
Global Anchor Tokens	512 per layer
Context Length	1,048,576 tokens (1M)
Hidden Dimension	5,120
Intermediate Dimension	13,696
Attention Heads	40
Key-Value Heads	8 (GQA)
Layers	48
Vocabulary Size	151,936
Positional Encoding	Rotary (RoPE), $\theta = 5{,}000{,}000$
Normalization	RMSNorm
Activation	SwiGLU

Table 1: Zen4 Architecture Specifications

2.2 Hierarchical Hybrid Attention

Standard self-attention over a sequence of length n requires $O(n^2)$ memory and compute, which is prohibitive for sequences beyond 128K tokens. Ring attention and sliding-window approaches mitigate this but either require complex distributed protocols or sacrifice global information flow.

HHA addresses this through a two-tier design. Each transformer layer processes tokens in two phases:

Local Phase. Tokens are partitioned into non-overlapping windows of size $w = 4096$. Within each window, full bidirectional attention is computed. Windows overlap by $w/4 = 1024$ tokens to preserve boundary context. This phase costs $O(n \cdot w)$ memory.

Global Phase. A set of $k = 512$ anchor tokens is maintained per layer through a learned selection mechanism. The full sequence cross-attends to these anchor tokens, propagating long-range information across windows. Anchor tokens are updated by aggregating from their assigned window via a learned pooling operation. This phase costs $O(n \cdot k)$ memory.

The total complexity is $O(n(w + k)) = O(n)$ for fixed w and k, enabling linear scaling with sequence length. On standard benchmarks through 32K tokens, HHA matches the performance of full attention; beyond 32K tokens no full-attention baseline exists for comparison at this parameter count.

2.3 Grouped Query Attention

Zen4 employs Grouped Query Attention (GQA) with 40 query heads and 8 key-value heads, reducing KV-cache memory by 5× relative to multi-head attention while maintaining quality within 0.3% on all standard benchmarks. The ratio of 40:8 was chosen by Pareto search over quality and inference throughput on a held-out validation set.

2.4 Native Tool Pretraining

Function-call data is integrated into the pretraining corpus rather than reserved for supervised fine-tuning. The pretraining mixture includes:

- Synthetic tool-call trajectories generated from 50,000 API schemas, covering REST, GraphQL, and local Python function signatures (12% of pretraining tokens).
- Web-scraped code repositories containing function definitions and call sites, providing implicit tool-use signal (part of 28% code allocation).
- Structured dialogue datasets where tool invocations are interleaved with natural language reasoning steps.

The model learns a unified token-level representation where function calls, their arguments (as structured JSON), and their return values are processed identically to natural language, with no special-case decoding logic required at inference time.

2.5 Distilled Fine-Tuning

The distilled fine-tuning framework transfers capability from larger Mixture-of-Experts teacher models into the 14B Zen4 architecture through a three-stage process:

1. **Expert Identification**: Analyze the teacher's activation patterns across a diverse routing task suite to identify functional expert clusters.
2. **Targeted Distillation**: Train student layers to reproduce teacher expert activations, not just output logits, using intermediate representation matching.
3. **Integration Training**: Fine-tune the full student on a mixture of original data and distillation signal to ensure coherence across the distilled components.

This approach yields measurable improvements over standard output-logit distillation on complex multi-step reasoning tasks, where intermediate representation fidelity matters more than surface output similarity.

3 Training

3.1 Pretraining Data

Zen4 pretraining uses 15 trillion tokens drawn from the following distribution:

Data Category	**Proportion**	**Tokens (approx.)**
Web (filtered, quality-scored)	42%	6.3T
Code (multi-language)	28%	4.2T
Tool-call trajectories	12%	1.8T
Scientific and technical text	10%	1.5T
Books and long-form prose	5%	0.75T
Mathematical content	3%	0.45T
Total	100%	15.0T

Table 2: Zen4 Pretraining Data Composition

Web data underwent three-stage filtering: perplexity-based quality scoring, near-duplicate removal via MinHash LSH (Jaccard threshold 0.9), and safety filtering via a classifier trained on human-labeled harmful content.

3.2 Training Infrastructure

Parameter	Value
Hardware	2,048 × H100 80GB SXM
Interconnect	InfiniBand NDR (3.2 Tb/s)
Parallelism	TP=4, PP=4, DP=128
Sequence Parallelism	Ring-SP for context > 128K
Batch Size	4M tokens (global)
Peak Learning Rate	3×10^{-4}
LR Schedule	Cosine with warmup (2000 steps)
Optimizer	AdamW ($\beta_1 = 0.9, \beta_2 = 0.95, \varepsilon = 10^{-8}$)
Weight Decay	0.1
Gradient Clipping	1.0
Precision	BF16
Training Duration	18 days

Table 3: Zen4 Pretraining Configuration

3.3 Post-Training

Following pretraining, a four-stage post-training pipeline aligns the model for instruction following and tool use:

1. **Supervised Fine-Tuning (SFT)**: 2M high-quality instruction-response pairs covering general dialogue, coding, mathematics, and tool-use scenarios.
2. **Preference Optimization**: Direct Preference Optimization (DPO) on 800K comparison pairs generated by a combination of human annotators and the Zen4-Pro model acting as a judge.
3. **Tool-Use Reinforcement**: GRPO-based reinforcement on tool-use trajectories, using ground-truth API call outcomes as the reward signal. This stage improves multi-turn tool-use accuracy by 23% over DPO-only post-training.
4. **Safety Alignment**: Constitutional AI-style critique-and-revision pass followed by PPO with a trained safety reward model.

4 Evaluation

4.1 Standard Benchmarks

Benchmark	Zen3	Zen4	Improvement
MMLU (5-shot)	79.1%	85.4%	+6.3%
MATH (4-shot)	74.8%	81.2%	+6.4%
HumanEval (0-shot)	80.2%	88.3%	+8.1%
GPQA Diamond (0-shot)	59.3%	67.4%	+8.1%
IFEval (prompt-strict)	76.4%	83.1%	+6.7%
GSM8K (8-shot)	88.9%	93.7%	+4.8%
HellaSwag (10-shot)	85.3%	89.1%	+3.8%

Table 4: Zen4 vs. Zen3 Benchmark Comparison

4.2 Context Length Evaluation

Task	Context	Score	Notes
NIAH (Needle-in-a-Haystack)	128K	98.4%	Single needle
NIAH	512K	96.1%	Single needle
NIAH	1M	91.7%	Single needle
Multi-needle NIAH	128K	94.2%	5 needles
RULER (average)	128K	87.3%	Multi-task
LongBench (average)	32K	83.6%	16 tasks

Table 5: Long-Context Retrieval and Understanding

4.3 Tool-Use Evaluation

Task	Zen3 (SFT)	Zen4 (Native)
Single-turn function call accuracy	81.3%	93.7%
Multi-turn tool-use (3 turns)	64.2%	84.9%
Argument JSON validity	88.7%	98.1%
Nested tool composition	43.1%	71.8%
Error recovery (self-correction)	38.4%	62.3%

Table 6: Tool-Use Accuracy: Zen3 (SFT-only) vs. Zen4 (Native Pretraining)

4.4 Efficiency Comparison

Metric	Zen3 (14B)	Zen4 (14B)	Change
FLOPs per token (forward)	28.6T	17.2T	−40%
Tokens/sec (A100, FP16)	2,840	4,210	+48%
TTFT at 32K context (ms)	890	520	−42%
Memory (KV-cache, 128K ctx)	34 GB	8.1 GB	−76%
Memory (KV-cache, 1M ctx)	n/a	28.4 GB	—

Table 7: Efficiency Metrics: Zen3 vs. Zen4 at 14B Parameters

5 Related Work

Long-context transformers have been approached through several complementary strategies. Sparse attention patterns [1] reduce complexity by restricting each token to a local neighborhood plus global tokens, an approach our HHA mechanism extends with a learned global anchor selection rather than fixed positions. Ring attention [2] distributes the attention matrix across devices for arbitrary sequence lengths but requires all-to-all communication that limits throughput at moderate sequence lengths. Our HHA avoids inter-device communication in the local phase.

Tool-augmented language models [3] pioneered the integration of API calls into generation, but relied on few-shot prompting and fine-tuning. Subsequent work [4] improved tool-call accuracy through domain-specific fine-tuning. Zen4's native tool pretraining extends this by making tool calls a pretraining objective, giving the model deeper statistical priors for tool selection and argument formation.

Knowledge distillation [5] traditionally transfers logit distributions from teacher to student. The Zen distillation framework extends this to intermediate representations, drawing on feature-level distillation [6] while adding the expert identification stage specific to mixture-of-experts teachers.

6 Deployment

6.1 Supported Formats

- **SafeTensors**: BF16 full-precision weights (28.6 GB)
- **GGUF**: Q4_K_M (8.1 GB), Q5_K_M (9.8 GB), Q8_0 (14.7 GB)
- **MLX**: 4-bit and 8-bit for Apple Silicon
- **vLLM**: Continuous batching with PagedAttention
- **TensorRT-LLM**: Optimized for NVIDIA Hopper/Ada

6.2 Hardware Requirements

Format	VRAM	Context	Hardware
BF16 (full)	32 GB	128K	A100 40GB ×1
BF16 (full)	80 GB	1M	H100 80GB ×1
Q4_K_M	10 GB	32K	RTX 3080 10GB
Q4_K_M	16 GB	128K	RTX 4080 Super
MLX 4-bit	12 GB	64K	M3 Pro (18GB)
MLX 4-bit	24 GB	512K	M4 Max (64GB)

Table 8: Deployment Hardware Requirements

6.3 API Usage

```
from hanzo import HanzoAI

client = HanzoAI()

tools = [{
    "type": "function",
    "function": {
        "name": "get_weather",
        "description": "Get␣current␣weather␣for␣a␣city",
        "parameters": {
            "type": "object",
            "properties": {
                "city": {"type": "string"},
                "units": {"type": "string", "enum": ["celsius", "
                    fahrenheit"]}
            },
            "required": ["city"]
        }
    }
}]

```

```
response = client.chat.completions.create(
    model="zen4-14b-instruct",
    messages=[{"role": "user", "content": "What␣is␣the␣weather␣in␣Tokyo
        ?"}],
    tools=tools,
    tool_choice="auto"
)
```

Listing 1: Zen4 with Native Tool Call

7 Safety and Alignment

7.1 Evaluation

Zen4 was evaluated on the standard Hanzo safety harness covering:

Safety Metric	**Zen3**	**Zen4**
Harmful content refusal rate	94.1%	97.3%
Over-refusal rate (benign)	8.4%	4.1%
Jailbreak resistance (AdvBench)	91.2%	96.4%
Factual accuracy (TruthfulQA)	71.3%	78.9%

Table 9: Safety and Alignment Metrics

7.2 Limitations

- Long-context recall degrades for documents with high token density and many distinct facts; precision on 1M context tasks averages 91.7% but drops to approximately 82% on adversarial multi-needle tasks.
- Tool-call argument generation occasionally hallucinate valid-looking but incorrect values for arguments not strongly constrained by the schema.
- Mathematical reasoning on competition-level problems (AIME, USAMO) benefits from extended thinking; Zen4-Thinking is recommended for those tasks.

8 Conclusion

Zen4 establishes a new performance ceiling for the 14B dense parameter class through architectural innovation rather than scale. The Hierarchical Hybrid Attention mechanism resolves the context-length memory wall, native tool pretraining elevates function-call reliability from a fine-tuning artifact to a first-class capability, and the fourth generation of distilled fine-tuning closes the gap to models more than twice Zen4's size. The resulting 40% efficiency gain over Zen3 makes Zen4 the recommended general-purpose deployment for applications requiring frontier reasoning, long-document understanding, or reliable agentic tool use within a single-GPU budget.

Acknowledgments

We thank the research teams at Hanzo AI and Zoo Labs Foundation, our infrastructure partners, and the open-source community whose tooling underpins evaluation and deployment.

References

[1] Beltagy, I., Peters, M.E. and Cohan, A. (2020). Longformer: The Long-Document Transformer. arXiv:2004.05150.

[2] Liu, H. et al. (2023). Ring Attention with Blockwise Transformers for Near-Infinite Context. arXiv:2310.01889.

[3] Schick, T. et al. (2023). Toolformer: Language Models Can Teach Themselves to Use Tools. NeurIPS 2023.

[4] Patil, S.G. et al. (2023). Gorilla: Large Language Model Connected with Massive APIs. arXiv:2305.15334.

[5] Hinton, G., Vinyals, O. and Dean, J. (2015). Distilling the Knowledge in a Neural Network. arXiv:1503.02531.

[6] Romero, A. et al. (2015). FitNets: Hints for Thin Deep Nets. ICLR 2015.

[7] Vaswani, A. et al. (2017). Attention Is All You Need. NeurIPS 2017.

[8] Touvron, H. et al. (2023). LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971.

[9] Ouyang, L. et al. (2022). Training Language Models to Follow Instructions with Human Feedback. NeurIPS 2022.

[10] Shao, Z. et al. (2024). DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300.

[11] Su, J. et al. (2021). RoFormer: Enhanced Transformer with Rotary Position Embedding. arXiv:2104.09864.

A Model Card

Field	Value
Model Name	Zen4
Version	v2026.01
Release Date	January 2026
Parameters	14.3B
Context Length	1,048,576 tokens
License	Apache 2.0
Repository	huggingface.co/zenlm/zen4-14b-instruct
Documentation	papers.zenlm.org/zen4
Contact	research@hanzo.ai

Table 10: Zen4 Model Card

B Benchmark Methodology

All evaluations use greedy decoding (temperature 0) unless otherwise specified. MMLU uses 5-shot chain-of-thought prompting. MATH uses 4-shot prompting with the standard MATH evaluation harness. HumanEval uses 0-shot with pass@1 metric. GPQA uses 0-shot with the Diamond subset. IFEval uses the prompt-level strict accuracy metric. Context-length evaluations use positions sampled uniformly across the full context window.