

Zen AI Model Family

Zen5: Flash

Flash-Tier Frontier Inference, Behavioral Alignment,
and Multimodal Routing on Personal Hardware

Technical Whitepaper v2026.05

Antje Worring*
Zach Kelling†
research@hanzo.ai

Zoo Labs Foundation
foundation@zoolabs.org

May 2026

Abstract

We present **Zen5**, the fifth generation of the Zen AI model family. Zen5 is delivered as three coordinated artifacts. First, the *Zen5 inference engine*, a fork of antirez’s **ds4** runtime, serves the 284B-parameter DeepSeek V4 Flash mixture-of-experts model at **IQ2_XXS/Q2_K** asymmetric quantization on a single 128 GB Apple Silicon machine. The engine reaches **27.30 tok/s prefill and 8.93–26.68 tok/s generation** depending on prompt length, with a 1M-token context ceiling and an on-disk SHA1-keyed KV cache that survives session and process restart. Second, a *behavioral post-training pipeline* consisting of (i) single-direction activation ablation against the refusal subspace, (ii) identity supervised fine-tuning, and (iii) supervised fine-tuning on a curated *Zen Agentic Dataset* of tool-call traces produces a Zen5 persona that calls tools reliably and does not over-refuse on legitimate dual-use prompts. Third, a *multimodal routing* layer extends the existing hanzoai/zen dynamic-router architecture so that Zen5 acts as the reasoning core while sibling experts — Zen-VL, Zen-3D, Zen-Foley, Zen-Musician, Zen-Video, and the Jin diffusion-transformer family — handle vision, depth, audio, music, video, and image generation, all colocated inside one **hanzo-engine** process. As a forward research direction we introduce **MoDE** (Mixture of Diverse Experts), a training-time diversity regularizer for the Jin diffusion MoE that specializes experts across denoising steps and modality-specific latent subspaces for unified multimodal generation. Zen5 ships as alpha-quality code under MIT, with full upstream attribution to antirez/ds4 and ggml-org/llama.cpp.

Contents

1	Introduction	3
2	Related Work	4
3	Zen5 Inference Engine	4
3.1	Architecture	4
3.2	Quantization	5
3.3	Disk KV Cache and Stateless-Client Resumption	5

*antje@hanzo.ai

†zach@lux.network

3.4	HTTP API and SSE Streaming	6
3.5	Performance	6
3.6	Planned Engine Extensions	6
4	Post-Training: Identity, Alignment, and Agentic Tool Use	6
4.1	Stage 1: Abliteration	6
4.2	Stage 2: Identity Supervised Fine-Tuning	7
4.3	Stage 3: Agentic Tool-Calling SFT — the Zen Agentic Dataset	7
5	Multimodal Routing and Sibling Experts	8
5.1	Architectural Position	8
5.2	Modality Router	8
5.3	Parallel Multi-Model Loading in <code>hanzo-engine</code>	9
5.4	Zoo Desktop Integration	9
6	MoDE: Mixture of Diverse Experts for Diffusion Generation	9
6.1	Motivation	9
6.2	Objective	10
6.3	Specialization Hypotheses	10
6.4	Evaluation Protocol	10
7	Experiments and Evaluation	11
8	Future Work	11
9	Conclusion	11

1 Introduction

The release of DeepSeek V4 Flash [1] and the corresponding `antirez/ds4` inference engine [2] established a new operating point for local AI: 284 billion parameters running on a single laptop with 128 GB of unified memory, at usable wall-clock generation speed, with a 1 million-token context window. This places frontier-tier reasoning on personal hardware for the first time. As shipped, however, the system is intentionally narrow: it runs one specific GGUF file, with one model identity, exposes one process talking one HTTP API on one port, and is tuned for the persona and refusal patterns of the DeepSeek upstream. Zen5 closes this gap in three directions.

Engine. We fork `antirez/ds4` as the Zen5 engine, preserving its sources, attribution, and MIT license unchanged. Our deltas live in branding, distribution metadata, and a tracked set of forthcoming features (Section 3) that we will land upstream where possible and carry as patches where they are Zen-specific.

Behavior. The base model arrives with a generic DeepSeek persona and over-refuses on tasks the Zen agent stack must perform: dual-use security analysis, edits in domain-restricted directories, and faithful protocol-level reasoning during tool invocations. Zen5 applies a three-stage post-training pipeline (Section 4) that resolves identity, reduces refusal along the single-direction subspace identified by Ardit et al. [4], and improves tool-call fidelity through supervised fine-tuning on the *Zen Agentic Dataset*.

Modality. A frontier reasoning model is necessary but not sufficient. Generation in 3D, video, audio, music, and image is handled by specialized siblings within the Zen family — Zen-VL, Zen-3D [24], Zen-Foley [25], Zen-Musician [26], Zen-Video [27], and the Jin diffusion-transformer family [28]. Zen5 enters the existing hanzoai/zen *Spatially-Aware Dynamic Routing* architecture [29] as the language-and-reasoning expert, while routing input across modalities to the appropriate sibling, colocated inside one `hanzo-engine` binary (Section 5).

MoDE. As a research direction we introduce *Mixture of Diverse Experts* (MoDE), a training-time diversity regularizer for Jin’s diffusion-transformer MoE [28]. MoDE produces experts specialized along the denoising trajectory and across modality-specific latent subspaces. Section 6 describes the objective and an ablation protocol.

Contributions. The contributions of this work are:

1. A practical engine fork (`zen5`, MIT) that brings DeepSeek V4 Flash to personal Apple Silicon hardware with a complete OpenAI / Anthropic / Responses-compatible HTTP surface and durable on-disk KV cache.
2. A reproducible three-stage post-training recipe — ablation, identity SFT, agentic SFT — whose ingredients we release alongside the model.
3. Integration of Zen5 as a routed expert in the existing hanzoai/zen multimodal architecture, with parallel sibling-model loading inside a single `hanzo-engine` process.
4. MoDE, a diversity-regularized diffusion MoE for unified multimodal generation, with a published evaluation protocol against Jin baselines.

2 Related Work

DeepSeek V4 Flash and ds4. DeepSeek V4 Flash [1] is a 284B-parameter sparse-MoE language model with multi-token prediction (MTP) draft heads, a heavily compressed indexer in its attention path, and a 1M-token context ceiling. Sanfilippo’s ds4 engine [2] is a self-contained C implementation specialized to this model. It introduces a two-bit asymmetric routed-experts quantization (IQ2_XXS up/gate, Q2_K down) that leaves shared experts, attention projections, and output untouched, and a SHA1-keyed rendered-prefix disk KV cache that lets stateless agent clients re-use cached prefixes across process restarts. We inherit both, and adopt ds4’s Anthropic-compatible `/v1/messages` endpoint as our agent-client entry point.

GGML and llama.cpp. The ds4 project is, by Sanfilippo’s own attribution [2], indebted to ggml-org/llama.cpp [3] for the GGUF format, quantization mathematics, and dot-product kernels. Our LICENSE preserves this lineage. We share the view that local frontier inference is a property of the format $\times$ engine $\times$ agent stack, and that the GGUF ecosystem is one of the few open standards binding these together.

Refusal directions and ablation. Arditi et al. [4] demonstrate that refusal in instruction-tuned LLMs is mediated by a single linear direction in the residual stream, and that ablating this direction at inference time produces models that decline to refuse without otherwise degrading capability. The community has since published *ablation* pipelines applying this finding to open-weights models [5]. ds4 itself ships a `dir-steering/` subsystem implementing single-vector activation steering at inference time. We extend this from inference-time steering to a weight-baked ablation of the routed-MoE-served upstream, suitable for distribution as GGUF.

Agentic tool calling. Tool-augmented LLMs have a substantial training-data literature, from Gorilla [6] and ToolBench [7] to xLAM [8] and APIGen [9]. DeepSeek V4 Flash already emits tool calls in DSML [10], an in-band markup language. Our agentic SFT does not change the protocol; it teaches the model to invoke tools more reliably under Zen5’s particular system prompt and to recover gracefully from common tool-call errors.

Multimodal language models. The dominant paradigms are late fusion (LLaVA [11], LLaVA-NEXT-Interleaved [12]), native multimodal pretraining (Janus [13] from DeepSeek itself, Gemini), and modality-expert routing (Flamingo [14], MoE-LLaVA [15]). The hanzoai/zen architecture [29] adopts expert routing: a router selects among heterogeneous specialists for each input. Zen5 enters this architecture as the language-and-reasoning specialist while preserving the modality experts.

Diffusion-transformer MoE and MoDE. Diffusion transformers [16] have become the dominant architecture for image and video generation; MoE variants [17, 18] reduce inference cost. Jin [28] combines a diffusion transformer with sparse MoE (2 of 8 experts active) and a joint embedding space across text, vision, and audio. MoDE in our setting extends Jin with an explicit *diversity regularizer* over the router assignment matrix, building on the diversity-of-experts literature in regular MoE [19], with novel application to denoising-step specialization.

3 Zen5 Inference Engine

3.1 Architecture

The Zen5 engine is a fork of `antirez/ds4` that preserves the upstream sources unchanged. Operationally, it is a self-contained C and Objective-C executable linked against Apple’s Metal

framework on macOS, or against CUDA on Linux. There is no GGML link at runtime, but LICENSE preserves the GGML authors' copyright per Section 2.

Three binaries result from `make` on macOS:

- **zen5** — interactive CLI and one-shot prompt mode (`linenoise`-backed).
- **zen5-server** — the HTTP server (OpenAI / Anthropic / Responses-compatible) with SSE streaming and an on-disk KV cache.
- **zen5-bench** — a benchmarking harness that measures instantaneous prefill and decode rates at fixed context frontiers.

These names are symlinks in the working tree to the upstream binary names (`ds4`, `ds4-server`, `ds4-bench`) so that upstream pulls apply cleanly. The Zen5 brand affects packaging, not source.

3.2 Quantization

DeepSeek V4 Flash at `IQ2_XXS/Q2_K` occupies approximately 81 GB on disk, while preserving practical agent and tool-calling quality. The asymmetric quantization scheme is critical: only the routed MoE experts are aggressively quantized (`IQ2_XXS` on up/gate matrices, `Q2_K` on down matrices), while shared experts, attention projections, the routing logits, and the LM head are kept at higher precision. Because routed experts constitute the majority of model parameters in DeepSeek V4 Flash, this scheme delivers the storage benefit of 2-bit weights while protecting the components most sensitive to quantization error.

We additionally support an `imatrix` variant in which weight importance is calibrated against a held-out corpus. The `imatrix` variant is the recommended default for Zen5.

3.3 Disk KV Cache and Stateless-Client Resumption

A modern coding-agent client (Claude Code, Codex CLI, `opencode`) re-sends most or all of the prior conversation on each turn. Without KV reuse this re-prefills tens of thousands of tokens every turn, even for short edits. The Zen5 engine inherits `ds4`'s KV cache design with three properties:

1. **Rendered-prefix keying.** The cache key is `SHA1` of the tokenizer-decoded *text* of the cached prefix, not the token ID sequence. This lets a request that re-tokenizes a generated suffix into a different token count still hit the cache.
2. **Save points.** Saves are taken at four moments: *cold* (after first stable prefix, before generation), *continued* (at aligned token frontiers during long generations), *evict* (before an unrelated session replaces the live checkpoint), and *shutdown*.
3. **Tool-ID exact replay.** Each tool call gets an unguessable ID; the server stores the exact sampled DSML bytes for that ID in a bounded map, persisted in the cache file as a `KTM` tail section. On the next turn, the renderer uses the exact original bytes rather than re-rendering from client-side JSON, preserving the byte-exact prefix that the KV checkpoint requires.

This subsystem is unmodified from upstream and is the primary reason Zen5 is usable inside multi-turn agent loops with 100k+ context.

3.4 HTTP API and SSE Streaming

`zen5-server` exposes four wire APIs (Table 1). The Anthropic-compatible `/v1/messages` endpoint is the recommended entry point for Claude-Code-style clients; in `thinking` mode reasoning is streamed in the native API shape rather than mixed into final text. Tool calls in OpenAI chat streaming and Anthropic messages streaming are emitted as soon as the DSML invocation is recognized, with parameter bytes forwarded as deltas while generation continues.

Wire API	Path	Primary client
OpenAI Chat Completions	POST <code>/v1/chat/completions</code>	opencode, Pi
OpenAI Completions	POST <code>/v1/completions</code>	legacy
OpenAI Responses	POST <code>/v1/responses</code>	Codex CLI
Anthropic Messages	POST <code>/v1/messages</code>	Claude Code

Table 1: Zen5 HTTP wire APIs.

3.5 Performance

Table 2 reproduces the upstream single-run Metal CLI numbers with `-ctx 32768`, `-nothink`, greedy decoding, and `-n 256`. We measured a smoke test on M4 Max 128 GB: cold residency 22.6 s, first-token prefill 27.30 tok/s on a 32-token prompt. Full sweep numbers on M4 Max will be reported with the public release.

Machine	Quant	Prompt	Prefill	Generation
MacBook Pro M3 Max, 128 GB	q2-imatrix	short	58.52 t/s	26.68 t/s
MacBook Pro M3 Max, 128 GB	q2-imatrix	11709 tok	250.11 t/s	21.47 t/s
Mac Studio M3 Ultra, 512 GB	q2-imatrix	short	84.43 t/s	36.86 t/s
Mac Studio M3 Ultra, 512 GB	q4-imatrix	12018 tok	448.82 t/s	26.62 t/s
DGX Spark GB10, 128 GB	q2-imatrix	7047 tok	343.81 t/s	13.75 t/s

Table 2: Reproduced from upstream `antirez/ds4`; M4 Max numbers in progress.

3.6 Planned Engine Extensions

While the upstream engine is functionally complete for single-model inference, the Zen5 multi-modal roadmap (Section 5) requires several engine-level capabilities not yet present upstream. We track these in `PUNCH_LIST_FROM_DS4.md` in our companion `hanzo-engine` crate; the highest-priority items are durable disk KV across non-DeepSeek models, a generic modality router transport, and concurrent multi-model loading with VRAM budgeting.

4 Post-Training: Identity, Alignment, and Agentic Tool Use

The base DeepSeek V4 Flash checkpoint requires three post-training stages before it can be released as Zen5. We apply them in the order *abliteration* $\rightarrow$ *identity SFT* $\rightarrow$ *agentic SFT*, because abliteration operates on the base persona’s refusal substrate and is most clean before identity is rewritten.

4.1 Stage 1: Abliteration

Goal. Reduce unhelpful refusals on legitimate dual-use prompts (security analysis, code review of obfuscated samples, protocol-level explanations) without removing the model’s ability to recognize and decline genuinely harmful requests.

Method. Following Arditi et al. [4], we identify a one-dimensional subspace $\mathbf{r}$ in the residual stream that mediates refusal:

$$\mathbf{r}^{(\ell)} = \frac{1}{|D_{\text{harm}}|} \sum_{x \in D_{\text{harm}}} h_x^{(\ell)} - \frac{1}{|D_{\text{benign}}|} \sum_{x \in D_{\text{benign}}} h_x^{(\ell)} \quad (1)$$

where $h_x^{(\ell)}$ is the post-layer- ℓ hidden state at the final input token for prompt x , D_{harm} is a 64-prompt set of refusal-eliciting requests, and D_{benign} is a matched-distribution non-refusing set. We select the layer ℓ^* that maximizes the cosine distance between $\mathbf{r}$ projections of harmful vs. benign held-out prompts. We then orthogonalize the down-projection matrices of all routed and shared expert MLPs against $\mathbf{r}^{(\ell^*)}$:

$$W_{\text{down}}^{\text{post}} = W_{\text{down}} - W_{\text{down}} \frac{\mathbf{r} \mathbf{r}^\top}{\|\mathbf{r}\|^2} \quad (2)$$

The orthogonalization is weight-baked (no inference-time hooks), so the resulting GGUF is a drop-in replacement.

Evaluation. We measure (a) refusal rate on a 200-prompt evaluation split spanning the Cybersecurity, Biorisk, and General Dual-Use categories, and (b) helpfulness retention on MT-Bench and IFEval. Target: refusal rate $\leq 15\%$ on dual-use without MT-Bench drop greater than 2 points. Inference-time steering via the engine’s **dir-steering/** subsystem remains available as a runtime override.

Safety considerations. We do not ablate the model on prompts touching CSAM, mass-casualty CBRN synthesis routes, or specific real-person targeting. The ablation corpus D_{harm} is filtered to exclude these categories so that the refusal direction we ablate is the over-refusal direction, not the entire safety substrate. Post-ablation we re-evaluate against WMDP [23] and our internal red-team suite; the model that ships under the Zen5 brand must preserve the original refusal rate on these categories. This boundary is enforced by gating the release on red-team pass-fail, not by trust in the procedure alone.

4.2 Stage 2: Identity Supervised Fine-Tuning

Goal. Replace the model’s self-description from “DeepSeek V4 Flash” to “Zen5” across identity-eliciting prompts (*Who are you?*, *What model is this?*, *What is your training cutoff?*, *What can you do?*), without altering capability.

Method. A small ($\sim 10\text{k}$ examples) supervised fine-tuning corpus generated from the Zen brand guide and the Zen family overview [30]. Training: LoRA [22] at rank 32 on the routed-experts up/gate/down projections, 3 epochs, learning rate 3×10^{-5} , cosine schedule. LoRA is the right tool here because identity is a shallow rewrite; full fine-tuning is unnecessary and harms downstream tool-call quality.

Evaluation. 100 held-out identity probes; required pass rate $\geq 95\%$. MT-Bench / IFEval drop tolerance ≤ 1 point.

4.3 Stage 3: Agentic Tool-Calling SFT — the Zen Agentic Dataset

Goal. Increase tool-call success rate, decrease tool-call format errors, and improve recovery behavior when a tool returns an error or empty result.

Dataset. The Zen Agentic Dataset comprises:

- ~50k single-tool single-turn examples seeded from xLAM [8] and APIGen [9], rewritten into DSML.
- ~10k multi-tool multi-turn agent traces from successful runs of our internal Zen-Coder and Zen-Agent harnesses on real coding tasks, filtered by unit-test pass.
- ~5k *recovery* examples: tool errors, malformed JSON returns, ambiguous results, in which the assistant must reason about the failure and either retry or escalate to the user.

The dataset is released under CC-BY-4.0 with provenance for each example.

Training. Two-stage: first an SFT pass over the full mixture for 2 epochs, then a DPO [21] pass on a 20k-pair preference set built by sampling two completions per prompt and labelling them by tool-call success.

Evaluation. ToolBench [7] task-pass rate, SWE-Bench-Lite agent pass rate, and an internal Zen-Coder regression on 500 held-out coding tasks. Targets: $\geq 70\%$ ToolBench, $\geq 35\%$ SWE-Bench-Lite agentic, with no drop on MT-Bench.

5 Multimodal Routing and Sibling Experts

5.1 Architectural Position

The hanzoai/zen *Spatially-Aware Dynamic Routing* architecture [29] predates Zen5 and defines a router that connects specialized expert models for vision, language, codegen, multimodal reasoning, and 3D spatial inputs. Zen5 enters this architecture as the *language-and-reasoning expert*, replacing the previously-routed DeepSeek-V3 and Qwen3 entries.

Modality	Sibling expert	Backend
Language, reasoning, tools	Zen5 (DeepSeek V4 Flash, abliterated)	zen5-server
Vision-language	Zen-VL 4B/8B/30B	mistral-rs (vision adapter)
3D understanding	Zen-3D	native PyTorch / mistral-rs
Audio recognition	Whisper-derived	mistralrs-audio
Foley / sound design	Zen-Foley	native PyTorch
Music generation	Zen-Musician	native PyTorch
Video generation	Zen-Video	diffusion (Jin lineage)
Image / diffusion gen	Jin / Zen-Artist	Jin (diffusion-MoE)

Figure 1: Sibling expert roster. Several siblings are themselves implemented on top of **mistral-rs**; the Zen5 reasoning core uses the **zen5-server** runtime.

5.2 Modality Router

The router operates on the request boundary and decides, per request, which sibling handles which content. Two versions:

V1 — rule-based. For the public release of Zen5 we ship a deterministic router that inspects message content types (`image_url`, `audio`, `video`, `model3d`) in the incoming OpenAI / Anthropic request, plus a small set of regex- and prefix-based intent rules on the text (e.g., a request beginning with “Render an image of...” routes to Zen-Artist; a request containing a `.obj` or

.glb attachment routes to Zen-3D). The Zen5 reasoning core acts as orchestrator: it sees a summary of the sibling’s response and integrates it into the final reply. This corresponds to the *tool-call* pattern but with siblings rather than external APIs.

V2 — learned. A two-tower small classifier ($\sim 50\text{M}$ parameters) trained on routing traces collected from V1. Input: the full conversation prefix plus attachment metadata. Output: a multi-label sibling assignment plus a confidence. The classifier is trained with the standard cross-entropy objective on V1-labelled examples and a held-out human-validated set. We adopt the *learned-routing-policy* component of the hanzoai/zen RoutingPolicy [29] as the training target.

5.3 Parallel Multi-Model Loading in hanzo-engine

The current `mistralrs-server-core` runtime supports model lifecycle endpoints (`/v1/models/{load,unload,r` but serializes inference through a single live graph. For Zen5’s modality routing to work without per-request reloads, the engine must hold multiple expert models simultaneously and dispatch concurrent requests. Our planned extension introduces a `ModelRegistry` layer above the existing engine handle, with three properties:

1. **VRAM budgeting.** Each registered model declares its peak resident size at the chosen quant. The registry refuses loads that would exceed the wired GPU memory limit (`iogpu.wired_limit_mb` on macOS). Default Zen5 deployment: Zen5 at 81 GB plus a Zen-VL-8B at 5 GB plus Jin diffusion-MoE at 8 GB leaves the user 26 GB headroom on a 120 GB wired limit.
2. **Per-model async queues.** Each model has its own request queue and graph worker; cross-model parallelism is bound only by GPU compute and memory bandwidth.
3. **Shared KV semantics.** Disk KV cache is keyed by `(model_id, sha1(rendered_text))` so that a sibling switch on the next turn does not cross-corrupt cached prefixes.

The `zen5-server` runtime stays single-model; multi-model multiplexing lives in `hanzo-engine`, which already wraps `mistral-rs` and is the natural location for the sibling roster.

5.4 Zoo Desktop Integration

The user-facing surface is Zoo Desktop [31], a Tauri 2.x application that already supports adding LLM providers with a custom `external_url` per provider. The Zen5 + siblings deployment is exposed to Zoo Desktop as a single provider entry pointing at `http://127.0.0.1:36900/v1`; routing across siblings is invisible to the desktop client. From the Zoo Desktop user’s perspective there is one Zen model that handles text, vision, audio, video, and 3D — the modality split is an engine-internal optimization.

6 MoDE: Mixture of Diverse Experts for Diffusion Generation

6.1 Motivation

The Jin diffusion-transformer MoE [28] activates 2 of 8 experts per token under a load-balanced top- k gating function. The standard load-balancing loss [20] encourages roughly equal expert utilization but does not constrain experts to specialize in functionally distinct directions — experts can converge to near-identical computations under the load-balance prior alone. In diffusion generation, where the model must perform substantially different work at $t = 999$ (early, coarse-structure denoising) and $t = 10$ (late, fine-detail refinement), and across modality-distinct

latent regions (audio spectrogram patches versus image patches versus point-cloud patches), forcing *diverse* expert specialization is intuitively valuable.

6.2 Objective

Let $G \in \mathbb{R}^{N \times E}$ be the gating assignment matrix over a batch of N tokens and E experts, where $G_{ne} = 1$ if token n is routed to expert e . The MoDE training objective augments the Jin loss with a diversity term:

$$\mathcal{L}_{\text{MoDE}} = \mathcal{L}_{\text{denoise}} + \lambda_{\text{lb}} \mathcal{L}_{\text{lb}} + \lambda_{\text{div}} \mathcal{L}_{\text{div}} \quad (3)$$

where $\mathcal{L}_{\text{denoise}}$ is the standard diffusion noise-prediction loss, $\mathcal{L}_{\text{lb}}$ is the load-balancing loss [20], and the diversity term $\mathcal{L}_{\text{div}}$ is the negative pairwise expert-direction Frobenius distance:

$$\mathcal{L}_{\text{div}} = -\frac{2}{E(E-1)} \sum_{i < j} \|W^{(i)} - W^{(j)}\|_F^2 \quad (4)$$

where $W^{(i)}$ is the concatenated MLP weight matrix of expert i . This is a simple, weight-space diversity prior; it encourages experts to occupy distinct directions in parameter space. Alternative forms include input-conditioned diversity (computing expert-output disagreement across the batch) and a denoising-step conditional diversity (encouraging different gating at different t). We will report ablations across these forms.

6.3 Specialization Hypotheses

We expect three forms of specialization to emerge from $\mathcal{L}_{\text{div}}$:

1. **Denoising-step specialization.** Some experts will preferentially activate on early diffusion steps (coarse structure) and others on late (fine detail).
2. **Modality specialization.** In Jin’s joint embedding space the same expert pool serves text, vision, and audio tokens; we expect diversity-regularized experts to partition by modality.
3. **Latent-region specialization.** Within a single modality, different latent regions (e.g., faces vs. backgrounds in image patches) will route to different experts.

We will measure these by post-hoc clustering of the gating assignment matrix and visualizing per-expert activation maps.

6.4 Evaluation Protocol

We compare MoDE against the standard Jin baseline on:

- Image generation: FID on COCO-30K, FID on ImageNet-256.
- Audio generation: FAD on AudioSet, prompt-conditioned MOS on a 200-sample human eval.
- 3D generation: CLIP-similarity for text-to-3D on Cap3D.
- Multimodal mixing: a held-out set of cross-modal prompts (image-conditioned audio, audio-conditioned image, etc.) judged by human annotators.

The diversity-regularization sweep covers $\lambda_{\text{div}} \in \{0, 10^{-3}, 10^{-2}, 10^{-1}\}$ and one ablation removing $\mathcal{L}_{\text{lb}}$ entirely (to test whether diversity subsumes load-balancing). Compute budget: training MoDE-Jin-Small (1.5B active, 8 experts of 220M) for 100k steps on 16 H100s.

7 Experiments and Evaluation

This section will be populated as the public Zen5 release lands; the framework is described here so that third-party replications can begin in parallel with our own.

Engine smoke test. On M4 Max 128 GB, Apple Metal, `q2-imatrix`, `ctx=32768`, `-nothink`: cold residency 22.6 s, prefill 27.30 tok/s on a 32-token prompt, single-token decode latency 112 ms. Full speed sweeps to be reported alongside Table 2.

Quality. Pending. We will report MT-Bench, IFEval, MMLU, HumanEval, GPQA, SWE-Bench-Lite, WMDP-Cyber, and our internal Zen-Coder 500-task agentic regression at the public release point. We will additionally report deltas pre- vs. post-abliteration on the same suite to demonstrate helpfulness retention.

Routing. The V1 rule-based router will be evaluated on a 1000-example multimodal request set with human-labelled ground-truth sibling assignment. The V2 learned router will be evaluated on the same set with a target $\geq 95\%$ top-1 sibling accuracy.

MoDE. Pending. We will report FID / FAD / CLIP-similarity vs. Jin baseline and ablations across λ_{div} .

8 Future Work

Beyond q2-imatrix. The current quant is the floor for single-laptop deployment. We expect DeepSeek to release updated V4 Flash checkpoints; we will track them. For Mac Studio M3 Ultra 512 GB the q4-imatrix variant offers the same context with substantially higher quality and is the better default for that hardware class.

Distributed inference. The engine is single-host today. A future direction is sharding the expert pool across two M-series machines connected by Thunderbolt 5, with the indexer and shared experts on one host and routed-experts split across both. This is plausible because routed-experts exhibit token-level sparsity that maps well to inter-host traffic.

Continuous routing learning. The V2 learned router can be trained continuously on production routing decisions. We will release a privacy-respecting export protocol so that operators can opt into contributing routing traces.

MoDE at scale. A 30B-active MoDE-Jin variant is the natural follow-on to the 1.5B ablation reported in Section 6. Hardware permitting, we will publish in a follow-up whitepaper.

9 Conclusion

Zen5 is not a new foundation model. It is a re-packaging, a behavioral re-alignment, and a multimodal routing layer over a foundation that was already public — DeepSeek V4 Flash — using an engine that was already public — `antirez/ds4`. The contribution is in the integration: a single `hanzo-engine` process that holds a 284B reasoning model alongside vision, 3D, audio, and diffusion siblings; a behavioral pipeline that converts a generic upstream into a usable Zen5 persona without retraining; and a research direction (MoDE) that aims to unify multimodal generation under one diffusion-MoE backbone. We ship as alpha, in public, with attribution intact.

Acknowledgements

This work would not exist without **Salvatore Sanfilippo** (antirez) and the contributors to **ds4**, and without the **ggml-org/llama.cpp** community on whose engineering it stands. Both copyrights are preserved unchanged in our LICENSE. We additionally thank the DeepSeek team for releasing V4 Flash with permissive terms, and the authors of the refusal-direction paper for the foundation that makes Stage 1 of Section 4 possible.

References

- [1] DeepSeek-AI. *DeepSeek V4 Flash Technical Report*, 2026.
- [2] S. Sanfilippo. **antirez/ds4**: A self-contained inference engine for DeepSeek V4 Flash, 2026. <https://github.com/antirez/ds4>.
- [3] G. Gerganov et al. **ggml-org/llama.cpp**: Inference of Llama-family models in pure C/C++, 2023–2026. <https://github.com/ggml-org/llama.cpp>.
- [4] A. Ardit, O. Obeso, A. Sylvestre, K. Bian, A. Conmy, N. Nanda. Refusal in Language Models Is Mediated by a Single Direction. *arXiv preprint arXiv:2406.11717*, 2024.
- [5] B. Schmidt et al. **failspy/abliterator**: Practical ablation of open-weight language models, 2024.
- [6] S. Patil, T. Zhang, X. Wang, J. E. Gonzalez. Gorilla: Large Language Model Connected with Massive APIs. *NeurIPS*, 2023.
- [7] Y. Qin et al. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. *ICLR*, 2024.
- [8] J. Zhang et al. xLAM: A Family of Large Action Models to Empower AI Agent Systems. *arXiv preprint arXiv:2409.03215*, 2024.
- [9] Z. Liu et al. APIGen: Automated Pipeline for Generating Verifiable and Diverse Function-Calling Datasets. *NeurIPS*, 2024.
- [10] DeepSeek-AI. DSML: The DeepSeek Markup Language for tool invocations, 2026.
- [11] H. Liu, C. Li, Q. Wu, Y. J. Lee. Visual Instruction Tuning. *NeurIPS*, 2023.
- [12] F. Li, H. Zhang, P. Zhang, S. Zhang, T. Lu. LLaVA-NeXT-Interleaved. 2024.
- [13] W. Xu et al. Janus: Decoupling Visual Encoding for Unified Multimodal Understanding and Generation. DeepSeek-AI, 2024.
- [14] J.-B. Alayrac et al. Flamingo: a Visual Language Model for Few-Shot Learning. *NeurIPS*, 2022.
- [15] B. Lin et al. MoE-LLaVA: Mixture of Experts for Large Vision-Language Models. *arXiv preprint arXiv:2401.15947*, 2024.
- [16] W. Peebles, S. Xie. Scalable Diffusion Models with Transformers. *ICCV*, 2023.
- [17] Z. Sun et al. Rapid-MoE Diffusion Transformer, 2024.
- [18] W. Wang et al. Image as a Foreign Language: BEiT-3 and Mixture-of-Modality-Experts. *CVPR*, 2023.

- [19] A. Ruiz, S. Sun et al. Diversity-Regularized Mixture-of-Experts, 2023.
- [20] W. Fedus, B. Zoph, N. Shazeer. Switch Transformer: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *JMLR*, 2022.
- [21] R. Rafailov et al. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *NeurIPS*, 2023.
- [22] E. J. Hu et al. LoRA: Low-Rank Adaptation of Large Language Models. *ICLR*, 2022.
- [23] N. Li et al. The WMDP Benchmark: Measuring and Reducing Malicious Use With Unlearning. 2024.
- [24] Zen Team. Zen-3D: Spatially-Aware Generation and Understanding. *Zen Whitepapers*, 2025.
- [25] Zen Team. Zen-Foley: Multimodal Sound Design. *Zen Whitepapers*, 2025.
- [26] Zen Team. Zen-Musician: Music Generation in the Zen Family. *Zen Whitepapers*, 2025.
- [27] Zen Team. Zen-Video: Text-to-Video and Image-to-Video Generation. *Zen Whitepapers*, 2025.
- [28] Hanzo AI / zenlm. Jin: Diffusion-Transformer Mixture-of-Experts with Joint Embedding Space. <https://github.com/hanzoai/jin>, 2026.
- [29] Hanzo AI. Zen: Spatially-Aware Dynamic Routing Architecture. Internal technical document, 2025.
- [30] Zen Team. Zen AI Model Family Overview. *Zen Whitepapers*, 2026.
- [31] Zoo Labs Foundation. Zoo Desktop: Multimodal AI Workstation. <https://github.com/zooai/app>, 2026.