EchoNest: A Proposed Hierarchical Recurrent–Attention Architecture for Efficient Instruction-Following

Frontier LLM, Inc. Design Paper — v0.1-alpha June 2026

Abstract

Since 2023, open-source decoder-only transformer architectures have converged on a stable set of components: pre-normalization with RMSNorm, rotary positional embeddings (RoPE), gated feed-forward networks (SwiGLU or close variants), and grouped-query attention (GQA/MQA). This convergence has produced strong results under the dominant training paradigm of large-scale raw-text pre-training followed by instruction tuning, typically on models with billions of parameters running on dedicated accelerators.

We propose **EchoNest**, a compact 102.8M-parameter hierarchical recurrent–attention model that explicitly separates local sequential processing from global context integration across three nested tiers. Unlike the uniform stack of the converged transformer, EchoNest uses recurrent units for local order-sensitive composition and reserves attention for the global tier. The model is trained directly on instruction–response pairs from random initialization, with no separate raw-text pre-training stage. It is designed for efficient inference on commodity CPUs rather than accelerators.

This paper characterizes the converged transformer stack and its underlying assumptions, presents the EchoNest architecture and training regime, and articulates the design priorities that motivate this alternative approach. Empirical results are not yet available; training of v0.1-alpha is in progress. This document serves as a design and position paper. A full technical report with reproducible specifications and controlled experiments will follow completion of training.

1. Introduction

The dominant approach in large language model development has emphasized scaling parameters, data volume, and accelerator compute. Under this regime, open-source architectures have rapidly converged on a small set of well-engineered components that perform well when trained at scale on GPUs/TPUs. While this convergence represents a significant engineering achievement, it also reflects a particular set of deployment assumptions: abundant accelerator memory, very large parameter counts, and a two-stage lifecycle of massive raw-text pre-training followed by instruction tuning.

We argue that a meaningfully different deployment context — small models running on commodity CPUs at the edge, with constrained memory and no dedicated accelerators — rewards a different set of architectural and training choices. EchoNest is our first attempt to design explicitly for that context.

This paper makes three contributions:

1. A concise characterization of the converged open-source transformer stack and the assumptions it encodes (Section 2).
2. A description of the proposed EchoNest hierarchical recurrent–attention architecture and its training regime (Section 3).
3. A comparative analysis and explicit statement of design priorities, including acknowledged trade-offs and limitations (Sections 4–6).

2. Background: The Converged Transformer Stack

By 2024–2025, transformer architectures for open-source LLMs had undergone rapid exploration followed by striking convergence. A practitioner examining the architecture sections of Llama 3/3.1/3.2, Mistral, Gemma 2, Qwen2/Qwen2.5, and Phi-3/Phi-4 families will find highly consistent building blocks. This uniformity is supported by mature kernel ecosystems (FlashAttention, fused RMSNorm/SwiGLU operations) that make the stack efficient on current accelerators.

Core components of the converged stack include:

- **Decoder-only uniform blocks** with self-attention and feed-forward sublayers connected by residual pathways. The removal of recurrence in favor of parallel attention (Vaswani et al., 2017) enabled scalable training and effective modeling of long-range dependencies.
- **Pre-norm with RMSNorm.** Normalization is applied before each sublayer. RMSNorm retains re-scaling while dropping re-centering, providing training stability at lower computational cost than LayerNorm.
- **Rotary Positional Embeddings (RoPE).** Relative position information is encoded directly into the attention dot product by rotating query and key vectors according to their positions. This approach extends more gracefully to longer contexts than absolute positional embeddings.
- **Gated feed-forward networks (SwiGLU).** Introduced by Shazeer (2020) and widely adopted following PaLM and Llama, these replace standard ReLU or GELU activations with a gated mechanism. The MLP intermediate dimension is typically ~2.67× the model hidden size in Llama-style architectures and accounts for a substantial portion of parameters and compute.
- **Grouped-Query Attention (GQA/MQA).** Query heads remain numerous while key/value heads are shared across groups. This significantly reduces KV cache size and improves inference throughput. For example, TinyLlama (Zhang et al., 2024) uses 32 query heads with only 4 KV heads.

Underlying assumptions. These choices optimize quality-per-FLOP on current accelerator hardware and benefit from highly optimized kernels. The stack also assumes a two-stage training lifecycle (massive raw-text pre-training followed by instruction tuning) and parameter counts in the billions. These assumptions are reasonable for frontier-scale development but are not universal. Recent work has begun exploring hybrid architectures that combine attention with recurrent or state-space components for improved efficiency or long-context behavior (e.g., Jamba and its successors).

3. EchoNest: A Proposed Hierarchical Recurrent–Attention Architecture

EchoNest is a compact instruction-following model organized into a three-tier nested structure rather than a flat stack of identical blocks. The design explicitly separates local sequential processing from global context integration instead of expecting this separation to emerge implicitly across many uniform layers.

Three-tier structure:

- **EchoUnit (local tier):** Recurrent units (LSTM-style) handle short-range, order-sensitive composition where sequential structure is most informative.
- **MicroNest (mid tier):** An intermediate aggregation stage that combines local representations into phrase- and clause-scale features.
- **MacroNest (global tier):** Attention operates only at the top tier, performing context integration where global mixing provides the greatest value.

This hybrid approach uses recurrence for compact local state and order encoding while confining the quadratic cost of attention to the tier where it is most useful. The motivation is efficiency-driven specialization for CPU inference and short-to-medium context lengths.

Model card (v0.1-alpha)

Property	Value
Parameters	102.8M
Architecture	Hierarchical recurrent–attention (proposed EchoNest)
Context window	512 tokens
Vocabulary	50,257 (GPT-2 tokenizer)
Embedding	768-dim, weight-tied output projection
Training objective	Next-token prediction on response tokens only (instruction tuning)
Target hardware	Commodity CPUs

Training regime (in progress). EchoNest v0.1-alpha is being trained directly on instruction–response pairs from random initialization, without a separate raw-text pre-training stage. The current mixture comprises approximately 100,000 prompt–response pairs drawn from Stanford Alpaca (general instruction following), Python code-instruction data, and CodeSearchNet (multilingual code). Optimization uses AdamW (weight decay 0.01), a peak learning rate of 1e-4 with linear warmup and cosine decay, effective batch size of 16, label smoothing of 0.1, and EMA decay of 0.999. All training is performed on CPU.

This regime deliberately trades the broad world knowledge acquired through trillion-token pre-training for a smaller, faster, and more controllable training process targeted at instruction-following behavior. The viability of this approach at the 100M scale remains an open empirical question and is a central focus of ongoing work.

Note on specifications. The public v0.1-alpha model card does not yet fully specify normalization scheme, activation functions within the recurrent and attention modules, or the precise mechanisms coupling the recurrent tiers to the attention tier. These details will be provided in a forthcoming technical report to enable independent reproduction.

4. Comparative Analysis

Dimension	Converged Transformer Stack	EchoNest (Proposed)
Core mechanism	Pure self-attention, no recurrence	Recurrence (local) + attention (global) hybrid
Macro-structure	Flat stack of identical decoder blocks	Three-tier nested hierarchy
Specialization	Largely emergent across uniform layers	Explicit, by tier
Positional information	RoPE (explicit, relative)	Implicit via recurrence order
Attention cost	Full self-attention at every layer	Reserved for global tier only
Parameters	Typically 1B–70B+	102.8M
Context window	2k → 128k+ tokens	512 tokens
Training lifecycle	Raw-text pre-training → instruction tuning	Instruction tuning from initialization (no pre-training)
Training compute	GPU/TPU clusters, trillions of tokens	CPU, ~100k instruction pairs
Target deployment	Dedicated accelerators	Commodity CPUs / edge

Three conceptual distinctions stand out:

1. **Recurrence vs. attention.** The transformer eliminated recurrence for parallelism and long-range modeling. EchoNest reintroduces recurrence at the local tier, trading some training parallelism for a compact hidden state and cheaper local processing. This trade-off becomes attractive when sequences are bounded (512 tokens) and the target hardware is CPU rather than GPU.
2. **Hierarchy vs. uniformity.** Standard transformers rely on depth and scale to induce functional specialization implicitly. EchoNest builds an explicit separation between local, mid-range, and global processing. The hypothesis is that an explicit hierarchy can achieve comparable structure with fewer parameters at small scale.
3. **Behavior vs. broad knowledge.** Pre-trained transformers acquire world knowledge from raw text before learning to follow instructions. EchoNest learns instruction-following behavior directly on a curated mixture. This choice accepts limited world knowledge in exchange for a cheaper, more targeted, and potentially more controllable training process.

5. Design Priorities

EchoNest is not positioned as a competitor to frontier-scale models on raw capability. It targets a different and underserved region of the design space. Three principles guide the approach:

1. **Design for the deployment context from the start.** Most "small" models are scaled-down versions of large-model architectures. EchoNest is architected from the ground up for CPU inference and edge deployment, placing the expensive global attention operation only where it contributes most.
2. **Structural efficiency over brute-force scale.** Rather than relying solely on parameter count and data volume to induce useful structure, the architecture builds in explicit specialization. If this reduces the parameter budget required for a given level of behavioral competence, the savings are particularly valuable in the small-model regime.
3. **Behavior-first, controllable training.** Instruction tuning from initialization on a modest curated dataset yields a fast, inexpensive, and tightly scoped training process. This is attractive for teams that need predictable, domain-focused assistants they can train and iterate on without large pre-training budgets.

In short, while the converged transformer stack answers "how do we build the most capable model given large-scale compute?", EchoNest explores "how do we build a genuinely useful model given commodity hardware and a modest training budget?"

6. Limitations and Current Status

Intellectual honesty requires stating constraints plainly:

- **Stage:** EchoNest v0.1-alpha is a research release. Training is in progress (early epochs). It is not production-ready.
- **Evaluation:** No quantitative results are available. Validation loss, perplexity, code-generation metrics, and instruction-following evaluations will be reported upon completion of training.
- **Context length:** 512 tokens limits applicability to long documents or extended multi-turn interactions.
- **Language and domain:** Primarily English with Python-focused code generation in v0.1. Broader multilingual code support is planned.
- **Scale and knowledge:** At 102.8M parameters with no raw-text pre-training, world knowledge and open-ended general capability are limited by design.
- **Architectural specification:** Full details of normalization, activation functions, and inter-tier coupling mechanisms are not yet public. A complete technical report will provide these specifications.

A rigorous evaluation requires controlled experiments: EchoNest compared against a parameter-matched standard transformer (~100M parameters, 512 context) trained on the identical instruction mixture, with measurements of CPU latency, memory usage, and task performance. Until those results exist, the case for EchoNest remains primarily architectural and philosophical.

7. Conclusion and Future Work

The open-source LLM field has converged on a strong but assumption-laden architecture: the decoder-only transformer with RMSNorm, RoPE, gated FFNs, and GQA, trained via large-scale pre-training. EchoNest steps outside this convergence with an explicit hierarchical recurrent–attention design and instruction tuning from initialization, optimized for commodity CPU deployment.

The immediate roadmap is to complete v0.1 training, release an evaluation suite and inference-optimized weights (v0.2), and follow with safety review and API integration (v1.0). The most critical near-term work is empirical: a full technical report with reproducible specifications and the controlled, parameter-matched comparisons outlined above.

If the explicit hierarchy delivers competitive instruction-following behavior at this scale and footprint, it will support the broader thesis that, for a growing class of deployments, the right architecture is not merely a smaller version of a frontier model, but a different one.

References

1. Vaswani, A. et al. (2017). *Attention Is All You Need.* NeurIPS.
2. Shazeer, N. (2020). *GLU Variants Improve Transformer.* arXiv:2002.05202.
3. Zhang, P. et al. (2024). *TinyLlama: An Open-Source Small Language Model.* arXiv:2401.02385.
4. Shao, M. et al. (2024). *Survey of different Large Language Model Architectures: Trends, Benchmarks, and Challenges.* arXiv:2412.03220.
5. Lieber, O. et al. (2024). *Jamba: A Hybrid Transformer-Mamba Language Model.* arXiv:2403.19887.
6. Wang, G. et al. (2025). *Hierarchical Reasoning Model.* arXiv:2506.21734.
7. Frontier LLM, Inc. (2026). *EchoNest v0.1-alpha Model Card* (internal).