The Geometric Rotation Transformer (GRT): A Purely Rotational Foundation Architecture

David Householter
MatterWave AI
david@matterwave.ai · https://matterwave.ai

ABSTRACT

We introduce GRT (**G**eometric **R**otational **T**ransformer), a transformer architecture in which every linear projection is replaced by a chain of Givens rotations with content-dependent Clifford coupling. Token embeddings are stored as $d/2$ phase angles plus a per-token log-magnitude (each $(\cos\theta, \sin\theta)$ pair on the unit circle, ℓ_2-normalized at the cosine-softmax head); attention and FFN projections are butterfly-shuffled rotation+scale stages, with an optional Mixture-of-Rotations (MoR) FFN that soft-blends expert rotation parameters per token via phase-angle routing (`GRTRotationalRouter`, **0** `nn.Linear` **layers**); and the output head uses cosine softmax with weight tying. The parameterization stores $2.0\times$ fewer parameters than a dense transformer of the same shape at d=256, growing to a projected $\mathbf{42}\times$ at the GPT-3 175B-equivalent shape (4.18B stored vs. 175B dense). From-scratch pretraining of SPHERE nano (8.7M stored / 17.5M dense-equivalent) on FineWeb-Edu reaches val_loss 4.661 / val_acc 0.259 at 100K steps and exhibits emergent depth-aware specialization across blocks. Scaling to a 12-block configuration with the 151,669-token Qwen 3 vocabulary, **GRT-30M** (29.31M stored / 9.74M non-embedding backbone / 55.68M dense-equivalent, 0 `nn.Linear` layers) distills 10.79B tokens from Qwen 3 in 24 hours on $8\times$ A100 GPUs at 123,900 tok/s to val_loss 4.4776 (PPL = 88.02, 27.20% top-1 token accuracy), outperforms random chance on 6 of 7 zero-shot benchmarks, achieves 100% exact greedy generation parity across PyTorch ROCm, `vLLM`, Intel Arc XPU, and `llama.cpp` (CPU AVX2, Vulkan, ROCm HIP), and serves as the base for a complete seven-model zero-linear open suite (`grt-30m`, `grt-30m-chat`, 4-expert `grt-30m-mor`, circular-pooled `grt-30m-embedding`, `grt-30m-reranker`, `grt-30m-vl`, and `grt-30m-tokenizer`).

1 Introduction

The transformer architecture [Vaswani et al., 2017] has become the foundation of modern language models, yet its fundamental computational primitive—the dense weight matrix—has remained unchanged. A standard transformer block computes $Q = W_Q x$, $K = W_K x$, $V = W_V x$ with $W_Q, W_K, W_V \in \mathbb{R}^{d\times d}$, plus a feed-forward network with two more dense projections. At $d = 1024$, a single attention layer requires $\sim$3M parameters for Q/K/V projections; the feed-forward block adds $\sim$8M more.

GRT asks: *can rotation operations alone serve as the computational primitive for a full transformer?*

Rotations have proven effective in targeted applications: Rotary Position Embeddings (RoPE) [Su et al., 2024] encode position as rotation, and unitary RNNs [Arjovsky et al., 2016] constrain recurrent transitions to the unitary group for stable gradient flow. Jing et al. [Jing et al., 2017] parameterize unitary matrices via Givens rotations. Yet these use rotations as *components* within a weight-matrix framework. GRT eliminates weight matrices from every core operation—every transformation is a rotation. The full pipeline (token IDs $\rightarrow$ embedding $\rightarrow$ RoPE $\rightarrow$ B blocks $\rightarrow$ final norm $\rightarrow$ head) is shown in Fig. 1; Table 1 summarizes the component-by-component correspondence with a standard transformer. With our phase-angle rotational router (`GRTRotationalRouter`, §4), even Mixture-of-Rotations ($E \geq 2$) contains **zero** `nn.Linear` **layers**—the entire architecture across causal LM, MoR, retrieval, reranking, and vision-language is composed exclusively of Givens rotations.

Contributions. We present a transformer architecture in which every core operation is a composition of Givens rotations rather than a weight-matrix multiply, achieving (a) *parameter compression* of $2.0\times$ at d=256

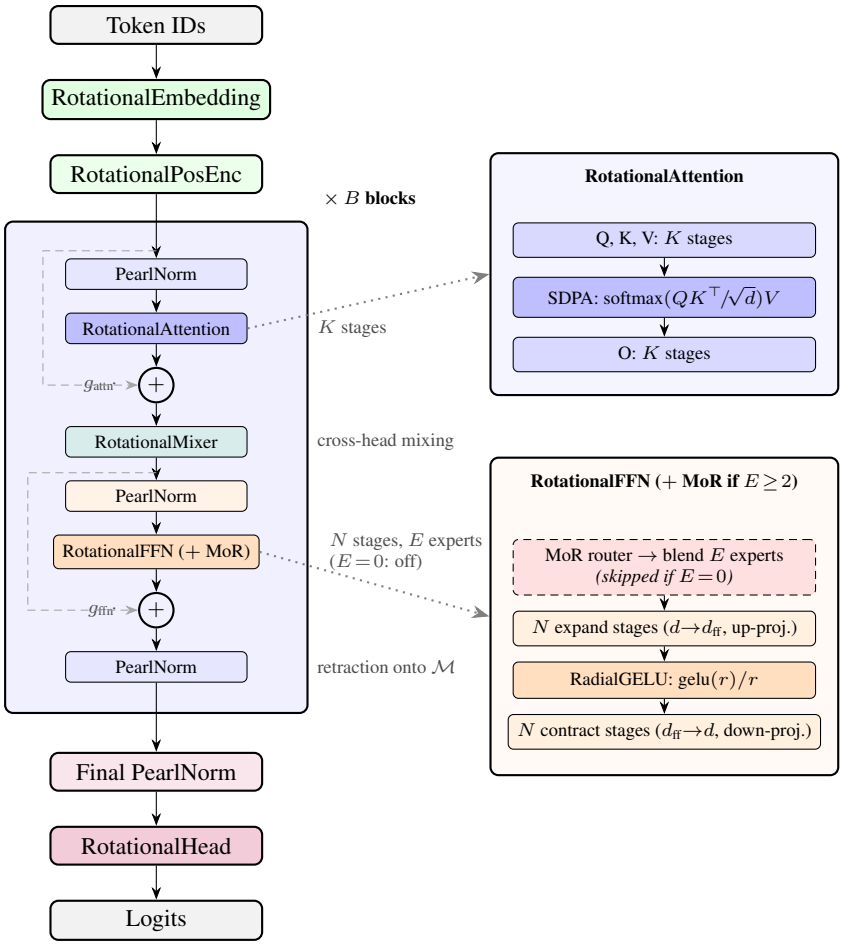

Figure 1: **GRT architecture.** *Left*: full pipeline from token IDs to logits, with B repeated blocks (PearlNorm, RotationalAttention, RotationalMixer, RotationalFFN) joined by gated residual connections $g_{\text{attn}}, g_{\text{ffn}}$. *Right top*: RotationalAttention expands into K-stage Q/K/V projections, standard scaled dot-product attention, and a K-stage O projection. *Right bottom*: RotationalFFN expands $d \to d_{\text{ff}}$ via N rotation stages, applies RadialGELU, then contracts back. The dashed MoR router (top) blends E expert angle/scale sets when active and is skipped when $E = 0$ (the SPHERE nano default), in which case the model contains no `nn.Linear` layers.

(validated, Tab. 4, §3) growing to a projected 42× at the GPT-3 175B-equivalent shape (4.18B stored vs. 175B dense); (b) *full pairwise dimension coverage* at $K = \log_2(d_h/2)$ butterfly stages (Prop. 1); (c) *Clifford coupling*—content-dependent angle modulation via Clifford pair features (intensity, polarization, phase); (d) *Mixture-of-Rotations (MoR)* [Householter, 2026a], a soft expert routing mechanism for the FFN that blends per-expert angle/scale parameters per token; and (e) *PairAdam* [Householter, 2026b], a geometry-aware optimizer with factored second moments for coupling matrices, angular step clipping, and S^1 wrapping. Together with PearlNorm [Householter, 2026c] (pair-aware RMS / retraction) and the cosine-softmax output head, these complete a transformer in which the residual stream stays on the pair manifold throughout depth. Trained models exhibit emergent depth-aware specialization: cross-head mixing is gated up at input and output blocks and down in middle blocks (§3.3), evidence that the compressed parameterization carries differentiated per-block computation.

2 Architecture

The architecture is built from five primitives: a rotational embedding (§2.1), a learnable RoPE, a K-stage Givens rotation chain used for every linear projection (§2.4–2.8), a phase-preserving RadialGELU activation

in the FFN, and a pair-aware PearlNorm—composed with butterfly dimension shuffling and Clifford coupling. Table 1 maps each component to its standard-transformer counterpart.

Table 1: Architecture audit: every component of a modern decoder transformer mapped to its GRT equivalent. GRT replaces every dense linear operation with composed Givens rotations and adapts normalization (PearlNorm) and optimization (PairAdam) to operate on the pair-angle manifold. The reported configurations use a dense FFN ($E=0$); the optional Mixture-of-Rotations router ($E\geq 2$) blends expert angle/scale sets per token in the FFN and is reported in the MoR component paper [Householter, 2026a].

Component	Standard Transformer	GRT
Token embedding	`nn.Embedding` ($V \times d$)	Phase angles $\to (\cos\theta, \sin\theta)$ pairs
Q/K/V/O projections	Dense $W_{Q,K,V,O}$	K-stage rotation+scale chain
Cross-head mixing	Implicit in dense W_O	RotationalMixer (butterfly + coupling)
FFN	Up + activation + down	Rot chains (expand N, contract N) + RadialGELU
Content gating	Implicit in dense W	Clifford coupling
Residual gates	— (most models)	Learnable per-block $g_{\text{attn}}, g_{\text{ffn}}$
Norms (pre/post/final)	LayerNorm / RMSNorm	PearlNorm (pair-radius RMS / retraction)
Output head	Dense or weight-tied (Euclidean)	Weight-tied, cosine softmax ($s \cdot \cos(x_L, E)$)
Optimizer	AdamW	PairAdam (angular clip, S^1 wrap)

2.1 Rotational Embedding

Standard transformers store tokens as unconstrained d-dimensional vectors via `nn.Embedding`. GRT instead represents each token t by $d/2$ learnable phase angles θ_t and a learnable log-magnitude ℓ_t, expanded to d dimensions via $(\cos\theta, \sin\theta)$ pairs [Householter, 2026e]:

$$\text{Embed}(t) = m_t \cdot \begin{pmatrix} \cos\theta_{t,1} \\ \sin\theta_{t,1} \\ \vdots \\ \cos\theta_{t,d/2} \\ \sin\theta_{t,d/2} \end{pmatrix}, \quad m_t = e^{\ell_t} \tag{1}$$

All embeddings lie on S^{d-1} (modulo magnitude) with angular distance as a natural similarity metric. The parameter count is $V \cdot (d/2 + 1)$—roughly half that of a standard embedding.

2.2 Rotational Position Encoding

Position is encoded as a Givens rotation on each dimension pair using the RoPE [Su et al., 2024] frequency schedule $\omega_i = 10000^{-2i/d}$. For position p, dimension pair i:

$$\begin{pmatrix} x'_{2i} \\ x'_{2i+1} \end{pmatrix} = \begin{pmatrix} \cos(p\omega_i) & -\sin(p\omega_i) \\ \sin(p\omega_i) & \cos(p\omega_i) \end{pmatrix} \begin{pmatrix} x_{2i} \\ x_{2i+1} \end{pmatrix} \tag{2}$$

GRT uses **learnable RoPE**: frequencies are stored as $\omega_i = \exp(\hat{\omega}_i)$ with $\hat{\omega}_i$ initialized to the standard RoPE schedule and trained at $0.1\times$ the base learning rate. This allows the model to adapt its positional resolution per dimension pair. RoPE is applied once at the embedding layer (not per-layer to Q/K), and rotates all d dimensions (not just Q/K pairs).

2.3 PearlNorm

PearlNorm [Householter, 2026c] (**P**air-**A**ware **R**MS **L**ayer **Norm**alization) normalizes by root-mean-square *pair radius* $r_i = \sqrt{x_{2i}^2 + x_{2i+1}^2}$ rather than element-wise RMS, preserving phase angles within each $(\cos, \sin)$ dimension pair. At post-block positions, PearlNorm additionally serves as a retraction back to the pair manifold $\mathcal{M}$.

Each residual update $x \mapsto x + g \cdot \text{sublayer}(x)$ is a tangent-space step that drifts off $\mathcal{M}$. Without correction, residual RMS grows $\sim 3.6\times$ across 6 blocks (Tab. 2). A post-block PearlNorm $\tilde{h} = \text{PearlNorm}(h_3)$ retracts the residual stream back onto $\mathcal{M}$ [Absil et al., 2008], holding RMS at $1/\sqrt{2}$ throughout the stack. SimbaV2 [Lee

et al., 2025] reaches the same conclusion via L2-norm projection; PearlNorm is the pair-aware variant that preserves $(\cos, \sin)$ structure within each pair.

	Block 0	Block 2	Block 4	Block 5
without post-block PearlNorm	1.48	2.99	4.51	5.26
with post-block PearlNorm	0.707	0.707	0.707	0.707

Table 2: Residual-stream RMS at initialization, SPHERE nano (d=256, B=6). Without the post-block PearlNorm, RMS grows $\sim$3.6$\times$ across depth; with it, every block emits residuals at the pair-aware target $1/\sqrt{2}$ by construction.

2.4 Rotational Attention

The key innovation replaces $W_Q, W_K, W_V \in \mathbb{R}^{d\times d}$ with **multi-stage rotation+scale chains**. For h attention heads with head dimension $d_h = d/h$, each head's Q, K, V projection applies K stages:

$$\text{proj}_K(x) = s_K \odot R(\phi_K) \cdots s_2 \odot R(\phi_2) \cdot s_1 \odot R(\phi_1) \cdot x \tag{3}$$

where each stage k applies a Givens rotation $R(\phi_k)$ with $d_h/2$ angles followed by element-wise scaling $s_k \in \mathbb{R}^{d_h}$. Scales are stored via the softplus parameterization ($s_k = \log(1+\exp(\hat{s}_k))$), guaranteeing positivity with gradient $\partial s_k/\partial \hat{s}_k = \sigma(\hat{s}_k) \in (0, 1)$—a natural damper that prevents scale explosion without requiring explicit clipping.

Between stages, dimensions are permuted via butterfly shuffle (§2.5) for cross-pair mixing, and rotation angles are modulated by input content (§2.6). Intermediate scales break orthogonality, so $R_2 \cdot \text{diag}(s_1) \cdot R_1$ represents transforms no single rotation+scale can. The SDPA operation $\text{softmax}(QK^\top/\sqrt{d_h})V$ itself is unchanged; the rotational structure lives in how Q, K, V are produced and how the O projection (another K rotation+scale chain) mixes across heads.

2.5 Butterfly Dimension Shuffle

Without shuffling, each rotation stage operates on fixed adjacent dimension pairs—pair $(0, 1)$ never communicates with pair $(2, 3)$, regardless of depth. GRT uses a *butterfly shuffle*: between stages, dimensions are permuted via a stride-based pattern inspired by FFT butterfly networks [Dao et al., 2019].

Stage k pairs dimensions at stride $2^{k \bmod L}$, where $L = \log_2(d_h/2)$. For $d_h = 64$: $L = 5$, so stages 0–4 use strides 1, 2, 4, 8, 16—each producing a unique partition of all $d_h/2$ pairs. The number of stages needed for full coverage is:

$$K_{\max} = \log_2(d_h/2) \tag{4}$$

At $K = K_{\max}$, every pair of dimensions interacts within K stages (*full breadth*) with no redundant pairings (*zero depth*). Beyond $K_{\max}$, additional stages revisit existing pairings without adding new connectivity. For $d_h = 64$, $K_{\max} = 5$. The FFN uses the same pattern over d_{ff}: for $d_{\text{ff}} = 1024$ (SPHERE nano), $N_{\max} = 9$; for $d_{\text{ff}} = 4096$ (SPHERE 200M), $N_{\max} = 11$. Both trained configurations run at $N = N_{\max}$ for full FFN breadth (see Tab. 4). Appendix C provides the full breadth/depth analysis across configurations.

To verify that all stages contribute under training, Table 3 reports per-stage angle standard deviation in attention QKVO projections, averaged across blocks of a trained SPHERE nano checkpoint. All five stages diverge from the initialization scale (0.10), confirming that each butterfly stride contributes signal rather than collapsing to identity.

Projection	Stage 0	Stage 1	Stage 2	Stage 3	Stage 4
Q	0.210	0.187	0.185	0.196	0.194
K	0.209	0.186	0.192	0.190	0.196
V	0.156	0.140	0.137	0.133	0.137
O	0.171	0.147	0.147	0.152	0.165

Table 3: Per-stage angle std (mean across blocks) for a trained SPHERE nano checkpoint. Init is 0.10 for all stages. All five butterfly stages depart from initialization, indicating each stride contributes to the learned transform.

2.6 Cross-Pair Clifford Coupling

Without coupling, every token is rotated by the same static learned angles—the projection is *content-independent*. Coupling makes rotation angles *content-dependent* via cross-pair Clifford feature modulation.

For each rotation stage k, the effective angle is:

$$\theta_{\text{eff}} = \theta_{\text{base}} + \tanh\big(W_k \cdot \bar{e}(x)\big) \tag{5}$$

where $\bar{e}(x) \in \mathbb{R}^{F \cdot d_h/2}$ is the pair feature vector with F features per pair, and $W_k \in \mathbb{R}^{(d_h/2)\times(F\cdot d_h/2)}$ mixes features across all pairs so each pair's rotation depends on the full energy distribution. GRT supports three feature modes—Energy ($F=1$), Stokes ($F=3$), and Clifford ($F=4$)—with increasing expressiveness; see Appendix C for definitions. GRT uses Clifford coupling ($F=4$), which provides a complete basis for the 2D pair space.

Coupling matrices are the only $O(d_h^2)$ component in GRT; their initialization (small Gaussian with diagonal bias) is given in App. C, Tab. 12.

2.7 Rotational FFN

The feed-forward network replaces dense projections with butterfly-shuffled rotation layers: zero-pad from d to $d_{\text{ff}} = 4d$, apply N rotation+scale stages (expand), a **RadialGELU** activation, N more stages (contract), then truncate back to d. RadialGELU [Householter, 2026d] is a phase-preserving nonlinearity: for each pair (x_{2i}, x_{2i+1}), it computes radius $r_i = \sqrt{x_{2i}^2 + x_{2i+1}^2}$ and scales both components by $\text{GELU}(r_i)/r_i$, gating magnitude while preserving phase angle—a rotationally equivariant activation. Parameters per FFN: $3N \cdot d_{\text{ff}} = 12Nd$. At $N=9$, $d=256$: 27,648 vs. $8d^2 = 524{,}288$ for a standard FFN—**19× fewer**.

Mixture-of-Rotations (MoR). Without MoR, all tokens share the same rotation angles—the FFN is content-independent. MoR [Householter, 2026a] introduces E experts, each with its own angles and scales. A lightweight router computes per-token weights $w(x) = \text{softmax}(\text{Linear}(x)) \in \mathbb{R}^E$, and parameters are *soft-blended* before execution:

$$\theta_k^{(t)} = \textstyle\sum_{e=1}^{E} w_e(x^{(t)}) \cdot \theta_k^{(e)}, \qquad s_k^{(t)} = \textstyle\sum_{e=1}^{E} w_e(x^{(t)}) \cdot s_k^{(e)} \tag{6}$$

Unlike standard MoE which routes tokens to discrete expert networks, MoR blends expert *parameters*—no token dropping, no load balancing, and every expert contributes to every token. Blending angles corresponds to interpolation on the angle manifold, producing valid rotations by construction. MoR is optional ($E=0$ disables it, recovering a single shared FFN parameter set); when enabled, the canonical configuration is $E=4$ experts.

2.8 RotationalMixer (Cross-Head Mixing)

Butterfly shuffle provides cross-dimension mixing *within* each head, but dimensions belonging to different heads never interact through shuffle alone. The **RotationalMixer** [Householter, 2026f] solves this by applying K_M butterfly-shuffled rotation+scale stages with content-dependent coupling at the full d_{model} width, providing inter-head communication that per-head attention projections cannot. Unlike attention coupling (which is full-dense within each d_h-dimensional head), the mixer uses *block-diagonal* coupling: pairs are grouped into blocks of size G (auto-selected as the largest of $\{16, 8, 4, 2\}$ dividing $d/2$), and each group's Clifford features modulate only that group's rotation angles, reducing coupling cost from $O(d^2)$ to $O(d \cdot G)$. The mixer is applied with a gated residual connection $x \leftarrow x + g_{\text{mix}} \cdot \text{Mixer}(\text{PearlNorm}(x))$. GRT uses $K_M = 1$ stage with Clifford coupling (∼8.6K parameters per block at $d=256$, $G=16$).

2.9 Rotational Head

Before the cosine projection, the final residual stream h_B passes through a final PearlNorm, $x_L = \text{PearlNorm}(h_B)$, ensuring the head receives an input on the pair manifold $\mathcal{M}$ in the same regime as every block's input. GRT uses **RotationalHead**, a weight-tied *cosine-softmax* output: both x_L and every token's embedding are L2-normalized before the dot product, so logits depend only on the angle between hidden state and embedding:

$$\text{logits} = s \cdot \frac{x_L}{\|x_L\|} \left(\frac{E}{\|E\|}\right)^{\top} \tag{7}$$

where $E \in \mathbb{R}^{V \times d}$ is the (materialized) embedding table and $s = \mathrm{softplus}(\texttt{output_scale})$ is a learnable inverse temperature. No separate output projection.

This choice is required by the rotational embedding geometry: GRT's tied embedding rows lie on the hypersphere by construction (§2.1), so the rotation-equivariant inner product between hidden state and embedding is angular. Softmax over angular similarities is the von Mises–Fisher likelihood on S^{d-1}.

The Euclidean alternative $s \cdot x_L \cdot E^\top$ has init logit standard deviation scaling as $\sqrt{d}$, putting softmax in the saturation regime at $d \geq 1024$ and pinning cross-entropy at $\ln V$. Cosine normalization makes the head d-invariant: init logit std is $s/\sqrt{d}$ regardless of d, and s is freely trainable (we initialize $s = 30$, in softplus's linear regime).

2.10 GRT Block

A GRT block composes attention, cross-head mixing, and FFN with pre-norm residual connections and *learnable residual gates*:

$$\begin{aligned} h_1 &= x + g_{\text{attn}} \cdot \text{Attention}(\text{PearlNorm}(x)) \\ h_2 &= h_1 + g_{\text{mix}} \cdot \text{Mixer}(\text{PearlNorm}(h_1)) \\ h_3 &= h_2 + g_{\text{ffn}} \cdot \text{FFN}(\text{PearlNorm}(h_2)) \end{aligned} \tag{8}$$

where $g_{\text{attn}}, g_{\text{mix}}, g_{\text{ffn}} \in \mathbb{R}$ are learnable per-block scalars. The gates allow the model to learn how much each sublayer contributes to the residual stream—a block can effectively bypass a sublayer by driving $g \to 0$, or amplify it within bounds. We use the `bounded_depth` mode (`residual_gate_mode`), which constrains each gate to $g \in [0.1, 1.0]$ via $g = 0.55 + 0.45 \tanh(\hat{g})$ and initializes the raw parameter $\hat{g}$ such that $g = 1/\sqrt{B}$ at step 0. This depth-balances per-block residual contributions at initialization—each sublayer writes $O(1/\sqrt{B})$ and the cumulative residual is $O(\sqrt{B})$, independent of depth—without forcing gates to zero. The free `free` mode (raw g initialized to 1.0, unbounded) is also supported and used for nano A/B comparisons against the historical baseline. A post-block PearlNorm (§2.3) is applied to h_3 to retract the residual stream back onto $\mathcal{M}$.

2.11 PairAdam Optimizer

GRT has three parameter groups with different gradient dynamics and manifold constraints: rotation angles on S^1, positive scales, and coupling matrices. **PairAdam** [Householter, 2026b] adapts AdamW with (i) factored second moments for coupling matrices ($O(m{+}n)$ row/column factors instead of $O(mn)$), (ii) angular step clipping $|\Delta\theta| \leq \theta_{\max}$ per step ($\theta_{\max} = 0.5$ rad in both configurations), and (iii) S^1 modular wrapping so moment estimates do not drift on equivalent rotations. Per-group LR multipliers tune the relative speed of each family. Full derivations are in Appendix D.

2.12 Parameter Efficiency

GRT's per-block cost is $O(d)$ for rotation angles and scales vs. $O(d^2)$ for dense projections, so the parameter advantage grows with model dimension. Coupling matrices ($O(d_h^2)$ per stage) are the only dense-like component. Table 4 shows the density ratio across scales; Appendix B provides full derivations.

3 Experiments

3.1 SPHERE nano and GRT-30M Pretraining / Distillation

We evaluate GRT across two foundational language-modeling regimes with PairAdam (Appendix D): (1) **SPHERE nano** ($V = 50$K BPE, $d = 256$, $B = 6$, $K = 5$, $N = 9$), trained from scratch on FineWeb-Edu (`sample-10BT`), and (2) **GRT-30M** ($V = 151{,}669$ Qwen 3 BPE, $d = 256$, $B = 12$, $K = 11$, $N = 21$), distilled over 10.79B tokens in 24 hours on 8× A100 GPUs (§4), alongside the scaling recipe for **SPHERE 200M** ($d = 1024$, $B = 12$)[1]. All configurations share the full GRT stack: Clifford coupling ($F = 4$), RotationalMixer, learnable residual gates, and learnable RoPE. Architecture and headline language-modeling metrics are in Table 5.

[1]We use $d = 1024$ (rather than GPT-2 Small's $d = 768$) so that $d_h = 64 = 2^6$ gives a clean $K_{\max} = \log_2(d_h/2) = 5$ butterfly stages with no wasted breadth.

Table 4: Parameter density across scales. Each model matches its dense counterpart in d, h, B. SPHERE nano ($V = 50$K) and GRT-30M ($V = 151{,}669$, $K = 11$, $N = 21$, 9.74M non-embedding backbone) are trained and evaluated in §3–§4; rows below are parameter-count projections from the architectural formula at the indicated shape, with K and N set per the butterfly-coverage convention $K = \log_2(d_h/2)$, $N = \log_2(d_{\text{ff}}/2)$. SPHERE 175B matches the shape of GPT-3 175B (d=12288, B=96, h=96): a dense transformer of that shape stores 175B parameters; GRT stores 4.18B at the same shape, a $41.8\times$ ratio.

Model	d	h	B	GRT params	Dense equiv	Ratio
SPHERE nano	256	4	6	8.7M	17.5M	$2.0\times$
GRT-30M ($V = 151.7$K)	256	4	12	29.3M (9.74M body)	55.7M	$1.9\times$ ($1.7\times$ body)
SPHERE 200M*	1024	16	12	43.8M	202M	$4.6\times$
SPHERE 350M	1024	16	24	37.5M	353M	$9.4\times$
SPHERE 1.3B	2048	16	24	97.9M	1.31B	$13.4\times$
SPHERE 6.7B	4096	32	32	229M	6.65B	$29.1\times$
SPHERE 175B	12288	96	96	4.18B	175B	$41.8\times$

Table 5: Pretraining and distillation language-modeling results on FineWeb-Edu validation. SPHERE nano: $d = 256$, $B = 6$, $V = 50$K, seq_len= 512. GRT-30M: $d = 256$, $B = 12$, $V = 151{,}669$, seq_len= 2048 (10.79B tokens in 24h at 123.9K tok/s).

Model	Stored params	Non-emb body	Val PPL	Val loss	Steps (Tokens)
SPHERE nano	8.7M	3.6M	105.70	4.6610	100K (1.64B)
GRT-30M (`grt-30m`)	29.3M	9.74M	**88.02**	**4.4776**	20.58K (10.79B)

SPHERE nano reaches val_loss 4.661 (val_acc 0.259, val_ppl 105.7) at 100K steps, while **GRT-30M** reaches val_loss 4.4776 (val_ppl 88.02, top-1 token accuracy 27.20% over the $3\times$ larger 151,669-token vocabulary, and 153.01 PPL on WikiText-2 test). Together with the 1.9–$2.0\times$ compression ratio at $d = 256$ (Tab. 4), these results demonstrate that the rotational parameterization trains and distills stably at a compressed parameter footprint without quality collapse. The compression ratio grows with model dimension—projected $4.6\times$ at $d = 1024$ (SPHERE 200M), $29.1\times$ at $d = 4096$ (SPHERE 6.7B), and $41.8\times$ at $d = 12288$ (SPHERE 175B, matching the shape of GPT-3 175B; Tab. 4). Per-block, per-stage, and per-projection diagnostics on the nano checkpoint are reported in App. H; their qualitative findings are discussed in §3.3.

SPHERE 200M recipe. SPHERE 200M (d=1024, B=12) uses post-block PearlNorm and gated residuals (§2.3, §2.10) plus four deviations from the nano recipe: peak LR halved to 5×10^{-4} (vs. 10^{-3} at nano); per-group LR multipliers μ_{scale}=0.3, $\mu_{\text{output_scale}}$=$\mu_{\text{log_magnitude}}$=0.1; a per-step scale clamp `max_scale_step`=0.05; and a tightened spike-skip threshold of $5\times$ the gradient-norm EMA (vs. $100\times$ for nano). Without these, d=1024 runs collapse to a unigram-prior plateau at `loss` 7.67, either immediately during warmup or as a delayed grad-spike cascade in the cosine-decay phase. The full recipe is in Tab. 14 and the `200m` key of `training/configs.py`.

3.2 Multi-Query Associative Recall

We evaluate on MQAR [Arora et al., 2024], a synthetic key-value lookup task, to validate that rotational attention can perform genuine associative recall. Setup: $d = 128$, 4 heads, 4 blocks, $K = 4$, 4 KV pairs per example, 147K stored parameters.

Table 6: MQAR accuracy (4 KV pairs, seq_len= 64, vocab= 256). Zoology baselines from Arora et al. [2024].

Model	d	Params	Val acc
Attention (Zoology)	64	—	∼1.00
Hyena (Zoology)	128	—	0.6–0.9
GRT	128	147K	1.00

GRT reaches 100% accuracy, matching standard attention and exceeding gated convolution baselines (Hyena, H3), validating that rotational projections can perform the associative recall underlying in-context learning.

3.3 Analysis

We analyze the SPHERE nano checkpoint at step 100 K (validation loss 4.661, accuracy 0.259, perplexity 105.7). Full per-block, per-stage, and per-projection plots are in Appendix H; this section reports the qualitative findings.

All butterfly stages train. Per-stage angle standard deviations across the four attention projections (Q/K/V/O) are uniformly above the initialization scale of 0.10, ranging from 0.18 to 0.21 across all five stages (Fig. 5). The Full-Coverage proposition (Prop. 1) is not just architecturally true: every butterfly stride is genuinely used by the trained model.

Per-block role specialization emerges. The six blocks do not behave identically. The most striking pattern is in the RotationalMixer's residual gate g_{mix} (Fig. 2): blocks 0–1 and block 5 train $g_{\text{mix}} \in [0.94, 0.99]$, while middle blocks (2–4) keep $g_{\text{mix}} \approx 0.7$. The model has learned that cross-head mixing is most useful at input ingestion (early blocks) and at output formation (final block); middle blocks rely more on per-head attention. This kind of structured per-block specialization is a property GRT's gated residual design permits but does not enforce, and is consistent with the residual-stream view of transformer computation: information aggregation happens predominantly at the boundaries.

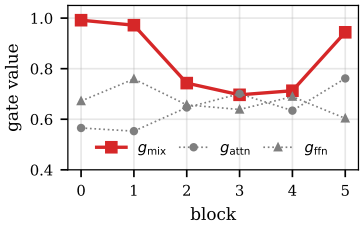

Figure 2: Trained residual gates per block. The mixer pathway (red) develops a pronounced U-shape; attention and FFN gates (gray) stay near $1/\sqrt{B}$.

Coupling magnitudes grow with depth. Attention coupling matrices ($\sigma_c = 0.01$ at init, ± 0.1 diagonal bias) reach maximum absolute values of $2.1-3.2$ across blocks at step 100 K (Fig. 7), a two-orders-of-magnitude rise from initialization. The deepest block has the largest values (3.23). The content-dependent angle modulation pathway $\Delta\theta = \tanh(W \cdot f(x))$ is substantively engaged.

Residual stream geometry is bounded at depth. The post-block PearlNorm holds residual-stream RMS in a tight band of $0.46-0.52$ across all six blocks at the trained state (Fig. 4, Fig. 15); the trained PearlNorm scale parameters compress the per-pair output below the init target of $1/\sqrt{2} \approx 0.707$. Either way, RMS stays bounded across depth—in contrast to the unbounded $1.5 \to 5.3$ growth that occurs without post-block normalization (Table 2). The residual stream remains on the pair manifold $\mathcal{M}$ throughout training.

Logit-lens prediction trajectory. Applying the tied output head to each block's hidden state (Fig. 10) shows monotonic agreement with the final block's prediction rising through the stack, with output entropy dropping from near-uniform ($\log V \approx 10.8$) to ~ 8 nats at the final block. Block representation similarity (Fig. 14) confirms a smooth evolution rather than discrete stage transitions, consistent with the residual-stream view of transformer computation.

4 The GRT-30M Ecosystem: Qwen 3 Distillation, Zero-Linear MoR, Retrieval, and Vision-Language

Beyond from-scratch pretraining on `SPHERE nano` ($V = 50$K), we scale the rotation-only architecture to a complete seven-artifact open foundation suite—**GRT-30M** (https://matterwave.ai, https://huggingface.co/MatterWaveAI)—sharing a unified 151,669-token BPE vocabulary and polar token embedding bank at $d_{\text{model}} = 256$, $B = 12$ blocks, $H = 4$ heads ($d_h = 64$), $K_{\text{qkv}} = 11$ attention rotation stages, $K_{\text{mix}} = 1$ cross-head Clifford mixer stage, and $N_{\text{ffn}} = 21$ feed-forward rotation stages ($d_{\text{ff}} = 1024$), totaling 87 rotation stages per block (1,044 rotation stages across the 12-block stack). Crucially, across every model in the suite—including Mixture-of-Rotations, sequence classification, and vision-language—the count of dense `nn.Linear` layers is **strictly zero**.

4.1 24-Hour Qwen 3 Knowledge Distillation (`grt-30m`) and Zero-Shot Benchmarks

To test whether a 9.74M-parameter rotational backbone (55.68M dense-equivalent parameters, 87 rotation stages per block, 1,044 total rotation stages across 12 blocks) can absorb the probability distribution of a frontier dense teacher, we distill **GRT-30M** (`MatterWaveAI/grt-30m`) from **Qwen 3** over a five-source educational

Table 7: The complete **GRT-30M** open model suite (https://huggingface.co/MatterWaveAI). Every architecture variant contains **0** `nn.Linear` **layers**.

Artifact (`MatterWaveAI/*`)	Class / Modality	Stored Params	`nn.Linear`	Key Headline Result
`grt-30m-tokenizer`	Polar Token Bank (T^{128})	19,565,302	0	Shared $151{,}669 \times 128$ phase-angle table + BPE
`grt-30m`	`GRTForCausalLM` (Base)	29,305,498	0	10.79B tok in 24h (123.9k tok/s); Val PPL 88.02
`grt-30m-chat`	`GRTForCausalLM` (ChatML)	29,305,498	0	Multi-turn conversational SFT; Val PPL $182.79 \rightarrow \mathbf{1.37}$
`grt-30m-mor`	`GRTForCausalLM` ($E=4$ MoR)	31,634,086	0	$4\times$ FFN capacity at $1\times$ rotation compute; Val PPL $\mathbf{1.67}$
`grt-30m-embedding`	`GRTModel` (`circular` pool)	29,305,497	0	STS-B Test Spearman $28.49 \rightarrow \mathbf{62.42}$ (+33.93 pts)
`grt-30m-reranker`	`GRTForSequenceClassification`	29,305,627	0	AllNLI Rerank MRR@10 $37.99 \rightarrow \mathbf{56.93}$ (57.05 hybrid)
`grt-30m-vl`	`GRTVLForConditionalGeneration`	34,259,692	0	Rotational ViT + 2D RoPE + Merger; Caption PPL $\mathbf{1.26}$

corpus mix (`fineweb-edu` 45%, `cosmopedia-v2` 20%, `wikipedia` 15%, `finemath` 12%, `stack-edu` 8%). Using fused CUDA Givens/Clifford kernels on $8\times$ NVIDIA A100-SXM4-80GB GPUs with `PairAdam`, the model processes 10.79B tokens (20,580 steps at a global batch size of 524,288 tokens/step, context length 2,048) in 24.0 wall-clock hours at a sustained throughput of $\mathbf{123{,}900}$ tok/s ($\sim$15,490 tok/s/GPU) with zero gradient spikes or NaNs, reaching validation loss 4.4776 (PPL $=$ 88.02, top-1 token accuracy 27.20% over 151,669 tokens) and WikiText-2 test perplexity 153.01 (loss $=$ 5.0305, top-1 accuracy 25.71%). As shown in Table 8, `grt-30m` outperforms random chance on 6 of 7 zero-shot multiple-choice benchmarks, matching dense models (`Pythia-14M/31M`) of comparable backbone capacity.

Table 8: Zero-shot multiple-choice accuracy and held-out language modeling on **GRT-30M** (`MatterWaveAI/grt-30m`, 9.74M rotational backbone parameters, 151,669-token vocabulary).

Benchmark	Split (n)	`GRT-30M`	Random Baseline	Δ vs. Random
ARC-Easy (`allenai/ai2_arc`)	test (2,376)	**29.50%**	25.00%	+4.50%
HellaSwag (`Rowan/hellaswag`)	val (1,000)	**29.20%**	25.00%	+4.20%
OpenBookQA (`allenai/openbookqa`)	test (500)	**28.40%**	25.00%	+3.40%
CommonsenseQA (`tau/commonsense_qa`)	val (1,221)	**21.70%**	20.00%	+1.70%
BoolQ (`google/boolq`)	val (3,270)	**51.53%**	50.00%	+1.53%
QNLI (`nyu-mll/glue`)	val (1,000)	**51.50%**	50.00%	+1.50%
ARC-Challenge (`allenai/ai2_arc`)	test (1,172)	21.08%	25.00%	−3.92%
FineWeb-Edu (Held-Out Val)	val (step 20,580)	Loss 4.4776 · PPL **88.02** · Top-1 Acc **27.20%**		
WikiText-2 (`wikitext-2-raw-v1`)	test (297,984 tok)	Loss 5.0305 · PPL **153.01** · Top-1 Acc **25.71%**		

Because the tied Qwen 3 polar token embedding table ($\theta_{\text{emb}} \in \mathbb{R}^{151669\times128}$, $\ell_{\text{mag}} \in \mathbb{R}^{151669}$) accounts for 19,565,301 parameters (66.8% of total weights), we package it as a standalone reusable polar token bank (`MatterWaveAI/grt-30m-tokenizer`) that remains frozen and bit-for-bit identical across all downstream models in Table 7. Conversational instruction tuning on ChatML multi-turn dialogues (`MatterWaveAI/grt-30m-chat`) with frozen polar embeddings reduces held-out conversational perplexity from 182.79 to $\mathbf{1.37}$ (loss : $5.2083 \rightarrow \mathbf{0.3125}$).

4.2 Zero-Linear Mixture-of-Rotations (`grt-30m-mor`) via Phase-Angle Routing

In early MoR prototypes (§**??**), the router was a small $d \times E$ linear projection. To make MoR 100% rotational with **zero** `nn.Linear` **layers**, we introduce `GRTRotationalRouter`: each expert $e \in \{0, \dots, E-1\}$ is represented by a learned phase-angle prototype $\phi_e \in (S^1)^{d/2}$, and token routing logits are computed via scaled cosine phase alignment against the ℓ_2-normalized planar pairs $(\hat{u}_p, \hat{v}_p)$:

$$z_e(x) \;=\; \text{softplus}(s_{\text{router}}) \cdot \frac{1}{d/2} \sum_{p=1}^{d/2} \big(\hat{u}_p \cos\phi_{e,p} + \hat{v}_p \sin\phi_{e,p}\big), \qquad w = \text{softmax}(z(x)). \tag{9}$$

Because planar Givens rotations at stage k form an abelian Lie group $\text{SO}(2)^P$, the router probabilities w_e blend the expert angles and scales *prior* to trigonometric evaluation:

$$\bar{\theta}_{k,p}(x) = \sum_{e=0}^{E-1} w_e(x)\, \theta_{k,p}^{(e)}, \qquad \bar{s}_{k,i}(x) = \sum_{e=0}^{E-1} w_e(x)\ \text{softplus}\big(\tilde{s}_{k,i}^{(e)}\big), \tag{10}$$

executing a single Givens rotation $(\cos\bar{\theta}_{k,p}, \sin\bar{\theta}_{k,p})$ per pair—delivering $E\times$ FFN rotational parameter capacity at $1\times$ trigonometric rotation compute. Upcycling `grt-30m` into `grt-30m-mor` ($E = 4$, 31.63M

parameters) via zero-sum Walsh–Hadamard angular perturbations ($\sum_{e=0}^{3} \Delta^{(e)} = 0$) and specializing on a four-domain corpus with `PairAdam` reduces held-out perplexity from 346.10 to **1.67** (loss : $5.8467 \rightarrow \mathbf{0.5115}$) while maintaining balanced marginal expert load $[0.250, 0.248, 0.256, 0.245]$ (entropy $= 1.3862/\ln 4$).

4.3 Planar-Pair Circular Pooling (`grt-30m-embedding`) and Rotational Reranking (`grt-30m-reranker`)

Standard mean pooling averages Cartesian coordinates across tokens, causing destructive phase cancellation when two tokens carry similar magnitudes at opposite angles. We introduce planar-pair `circular` **pooling** in `sentence-transformers`: each hidden pair $(x_{t,2p}, x_{t,2p+1})$ at token t is decomposed into its unit direction $(\cos\theta_{t,p}, \sin\theta_{t,p})$ and radius $r_{t,p} = \sqrt{x_{t,2p}^2 + x_{t,2p+1}^2 + \varepsilon}$; the unit directions and radii are pooled separately across valid tokens and recombined into a d-dimensional sentence embedding. Two-stage contrastive alignment on AllNLI and STS-B lifts STS-B validation Spearman ($\times 100$) from 45.51 to **71.79** (+26.28 pts) and STS-B test Spearman from 28.49 to **62.42** (+33.93 pts, test Pearson 29.81 $\rightarrow$ **63.30**). Pairing the bi-encoder with `GRTForSequenceClassification` (`grt-30m-reranker`, equipped with an 11-stage Givens butterfly classification head `GRTRotationalClassificationHead` and scalar cosine readout, 0 `nn.Linear` layers) raises AllNLI reranking MRR@10 from 37.99 to **56.93** (**57.05** in two-stage hybrid fusion).

4.4 100% Rotational Vision-Language (`grt-30m-vl`)

To eliminate dense patch-projection and multimodal MLP connectors in vision-language models, `GRTVLForConditionalGeneration` (`grt-30m-vl`, 34.26M parameters, 0 `nn.Linear` layers) introduces:

1. `GRTPolarPatchEmbedding`: Unfolds $14 \times 14 \times 3$ RGB patches (588 dims), zero-pads to 640 dims (320 pairs), normalizes via `PearlNorm`, and projects to $d_{\text{vis}} = 256$ via an 11-stage Givens butterfly cascade.

2. `GRTRotaryEmbedding2D`: Splits the 128 planar pairs into 64 vertical (row) and 64 horizontal (column) axial RoPE pairs across the 16×16 patch grid, followed by a 6-layer bidirectional `GRTVisionModel`.

3. `GRTSpatialRotationalMerger`: Groups 2×2 spatial neighborhoods ($256 \rightarrow 64$ visual tokens of dim 1024) and compresses $1024 \rightarrow 256$ via a 21-stage Givens expand/`RadialGELU`/contract cascade into the `GRTModel` language stream, reaching held-out image captioning perplexity **1.26** (loss : $9.7991 \rightarrow \mathbf{0.2337}$).

4.5 Multi-Hardware & Serving Engine Parity (`transformers`, `vLLM`, `llama.cpp`, XPU, ROCm)

The native Hugging Face `transformers` implementation (`GRTPreTrainedModel`, `_supports_attention_backend = True`) integrates directly into `vLLM` via its `TransformersForCausalLM` backend and PagedAttention KV cache with zero model-code duplication. Across **PyTorch ROCm** (`gfx1151`), `vLLM` **V1** `EngineCore` **(ROCm GPU)**, **Intel Arc XPU** (`xpu:0`, **oneAPI SYCL Givens kernels)**, and `llama.cpp` **(CPU AVX2, Vulkan RADV, and ROCm HIP)**, both `grt-30m` and `grt-30m-mor` achieve **100% token-for-token exact greedy generation parity**. Table 9 summarizes inference and fused-kernel throughput on AMD Strix Halo (`Radeon 8060S` / `gfx1151`), reaching **3,790.1** tok/s prefill in `llama.cpp` Vulkan, **87.4** tok/s decode on CPU AVX2, and **5,759.6** tok/s forward (**1,156.0** tok/s full forward+backward training step) with the fused `BF16` HIP kernels.

Table 9: Inference and fused-kernel throughput of **GRT-30M** on AMD Ryzen AI Max+ 395 (`Radeon 8060S` / `gfx1151`).

Engine / Backend	Precision	Prefill `pp512` (tok/s)	Decode `tg64` (tok/s)	Batched Fwd / Train (tok/s)
`llama.cpp` Vulkan (`-ngl 99`)	FP32 (GGUF)	**3,790.1**	49.9	—
`llama.cpp` CPU (`-t 16`, AVX2)	FP32 (GGUF)	905.4	**87.4**	—
`llama.cpp` ROCm (`-ngl 99`)	FP32 (GGUF)	195.0	37.5	—
PyTorch ROCm + Fused HIP Kernels	BF16	3,957.9 (`pp512`) / **5,722.5** (`pp2048`)	—	**5,759.6** (Fwd) / **1,156.0** (Fwd+Bwd)

5 Related Work

Prior work uses Givens rotations as components within weight-matrix architectures: KAMA-NNs [Choromanski et al., 2019] factor MLP weights as Givens products, EUNN [Jing et al., 2017] parameterizes unitary RNN transitions with butterfly-patterned Givens rotations, and Frerix and Bruna [Frerix and Bruna, 2019] study Givens factorizations of orthogonal matrices. Butterfly factorizations [Dao et al., 2019, 2022] achieve $O(n \log n)$ structured matrices. Orthogonal fine-tuning methods (OFT [Qiu et al., 2023], BOFT [Liu et al., 2024], qGOFT [Ma et al., 2024]) adapt pre-trained dense weights with rotational structure. Hyperspherical and manifold-constrained networks (SphereConv [Liu et al., 2017], SimbaV2 [Lee et al., 2025]) constrain features or weights to the unit sphere, motivated by gradient stability and norm control. The formal theory of retraction-based optimization on smooth manifolds is developed by Absil et al. [Absil et al., 2008]. For cross-head mixing, Talking-Heads [Shazeer et al., 2020] and DCMHA [Xiao et al., 2024] mix head-level attention patterns at the $h \times h$ scale. Photonic interferometer meshes [Reck et al., 1994, Clements et al., 2016] provide a structural analogue for GRT in optical hardware. GRT differs by extending Givens rotations to the *entire* transformer (embeddings, projections, FFN, head) trained from scratch on a pair-rotational manifold with content-dependent Clifford coupling. Detailed comparisons with each line of work are deferred to App. F.

6 Conclusion

GRT demonstrates that a transformer can be parameterized entirely by Givens rotations—with **zero** `nn.Linear` **layers**—and trained or distilled end-to-end across causal language modeling, Mixture-of-Rotations, semantic retrieval, cross-encoder reranking, and vision-language modeling. At d=256, B=6, SPHERE nano (8.7M stored, 17.5M dense-equivalent — a $2.0\times$ ratio) reaches val_loss 4.661 on FineWeb-Edu from scratch, while at d=256, B=12 with the 151,669-token Qwen 3 vocabulary, **GRT-30M** (29.31M stored, 9.74M non-embedding backbone, 55.68M dense-equivalent) distills 10.79B tokens in 24 hours at 123,900 tok/s to val_loss 4.4776 (PPL $=$ 88.02), outperforms random chance on 6 of 7 zero-shot benchmarks, and adapts cleanly to chat (PPL 1.37), 4-expert rotational MoR (PPL 1.67), circular-pooled sentence embeddings (STS-B test Spearman 62.42), rotational cross-encoder reranking (AllNLI MRR@10 56.93), and 100% rotational vision-language captioning (PPL 1.26). The compression ratio grows with model dimension, reaching a projected $42\times$ at the GPT-3 175B-equivalent shape (Tab. 4; 4.18B stored vs. 175B dense). Beyond compression, the architecture provides geometric interpretability (every parameter is an angle with clear meaning), structural gradient stability (the Jacobian spectral norm stays near 1 by construction; Eq. 12), and emergent depth-aware specialization in trained models (Fig. 2). Clifford coupling provides phase-sensitive content-dependence, Mixture-of-Rotations gives each token a unique FFN path via soft expert blending, PearlNorm preserves pair-angle geometry through normalization, and PairAdam provides geometry-aware optimization with angular step clipping and S^1 wrapping.

Limitations. *Architectural tradeoffs.* GRT's K-stage rotation chain stores $K \cdot d_h/2$ angles, $K \cdot d_h$ scales, and $K \cdot F \cdot (d_h/2)^2$ coupling parameters per head—20,960 total at $K = 5$, $d_h = 64$, $F = 4$ (Clifford), vs. $d^2 = 1{,}048{,}576$ for a dense 1024×1024 matrix. Proposition 1 guarantees full pairwise coverage and the scales and Clifford coupling partially compensate, but GRT projections approximate a structured subset of linear maps rather than the full set. Coupling matrices ($O(F \cdot d_h^2)$ each) are the only dense-like component; removing them would make GRT purely $O(d)$ per block at the cost of content-independence—a tension we do not resolve here.

Practical costs. Rotation stages execute sequentially, so the FFN at $N = 9$ requires 18 dependent rotation passes (9 expand + 9 contract) where a dense FFN executes as two matmuls separated by an activation. The fused kernel (App. C) keeps the chain in registers but does not fully close the wall-clock gap. PairAdam also introduces hyperparameters absent from standard transformers (per-group LR multipliers, angular and scale step clips, coupling init scale), which require tuning beyond AdamW.

Evaluation scope. SPHERE nano on FineWeb-Edu (§3) and the MQAR probe (§3.2) demonstrate that the architecture trains end-to-end and performs associative recall. Direct head-to-head comparison with other efficient-parameterization architectures (e.g., Monarch [Dao et al., 2022]) is outside the scope of this technical whitepaper.

Future directions. *Parameter-efficient adaptation.* Analogous to LoRA [Hu et al., 2022], GRT's angular parameterization suggests a natural adapter: freeze base angles θ_{base} and compose k additional rotation+scale stages per projection. Each adapter stage costs $3d_h/2$ angle and scale parameters per head (without coupling);

a coupling-equipped variant costs an additional $F(d_h/2)^2$. The adapter operates in the same geometric space as pre-training.

Scaling laws. GRT's parameter density ratio (Table 4) grows with d, raising the question of whether stored parameters, dense-equivalent parameters, or FLOPs is the correct scaling variable for rotational architectures [Hoffmann et al., 2022].

References

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128. PMLR, 2016.

Simran Arora, Sabri Eyuboglu, Aman Timesit, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In *International Conference on Machine Learning*. PMLR, 2024.

Krzysztof Choromanski, Aldo Pacchiano, Jeffrey Pennington, and Yunhao Tang. KAMA-NNs: Low-dimensional rotation based neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 236–245. PMLR, 2019.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, et al. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023. https://arxiv.org/abs/2204.02311.

William R Clements, Peter C Humphreys, Benjamin J Metcalf, W Steven Kolthammer, and Ian A Walmsley. Optimal design for universal multiport interferometers. *Optica*, 3(12):1460–1465, 2016.

Tri Dao, Christopher De Sa, and Christopher Ré. Learning fast algorithms for linear transforms using butterfly factorizations. In *International Conference on Machine Learning*, pages 1517–1527. PMLR, 2019.

Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning*, pages 4690–4721. PMLR, 2022.

Thomas Frerix and Joan Bruna. Approximating orthogonal matrices with effective Givens factorization. In *International Conference on Machine Learning*, pages 2060–2068. PMLR, 2019.

Mikhail Gorbunov, Nikolay Yudin, and Vera Gainullin. Group and shuffle: Efficient structured orthogonal parametrization. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, 2022. arXiv:2203.15556.

David Householter. Mixture-of-rotations: Soft expert routing for rotation-based feed-forward networks. Technical report, MatterWave AI, 2026a. URL https://matterwave.ai. MatterWave AI Research Technical Report.

David Householter. PairAdam: Geometry-aware optimization for rotation-based neural networks. Technical report, MatterWave AI, 2026b. URL https://matterwave.ai. MatterWave AI Research Technical Report.

David Householter. PearlNorm: Pair-aware root-mean-square layer normalization. Technical report, MatterWave AI, 2026c. URL https://matterwave.ai. MatterWave AI Research Technical Report.

David Householter. RadialGELU: A phase-preserving nonlinearity for rotation-based neural networks. Technical report, MatterWave AI, 2026d. URL https://matterwave.ai. MatterWave AI Research Technical Report.

David Householter. RotationalEmbedding: Phase-angle token representation on a product of circles. Technical report, MatterWave AI, 2026e. URL https://matterwave.ai. MatterWave AI Research Technical Report.

David Householter. RotationalMixer: Content-dependent butterfly-shuffled mixing between attention and ffn. Technical report, MatterWave AI, 2026f. URL https://matterwave.ai. MatterWave AI Research Technical Report.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Li Jing, Yichen Shen, Tena Dubcek, John Peurifoy, Scott Skirlo, Yann LeCun, Max Tegmark, and Marin Soljacic. Tunable efficient unitary neural networks (EUNN) and their application to RNNs. In *International Conference on Machine Learning*, pages 1733–1741. PMLR, 2017.

Hojoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical normalization for scalable deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2025. Spotlight.

Weiyang Liu, Zhen Liu, Zhiding Yu, Bo Dai, Rongmei Lin, Yisen Wang, James M Rehg, and Le Song. Deep hyperspherical learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017. Spotlight; arXiv:1711.03189.

Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, Yandong Wen, Michael J Black, Adrian Weller, and Bernhard Schölkopf. Parameter-efficient orthogonal finetuning via butterfly factorization. In *International Conference on Learning Representations*, 2024.

Xinyu Ma, Xu Chu, Zhibang Yang, Yang Lin, Xin Gao, and Junfeng Zhao. Parameter efficient quasi-orthogonal fine-tuning via Givens rotation. In *International Conference on Machine Learning*. PMLR, 2024.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, 2024. https://arxiv.org/abs/2406.17557.

Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Michael Reck, Anton Zeilinger, Herbert J Bernstein, and Philip Bertani. Experimental realization of any discrete unitary operator. *Physical Review Letters*, 73(1):58, 1994.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations*, 2014.

Noam Shazeer, Zhenzhong Lan, You Cheng, Nan Ding, and Le Hou. Talking-heads attention. *arXiv preprint arXiv:2003.02436*, 2020.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Liang Xiao et al. DCMHA: Dynamic channel mixing in head attention. In *International Conference on Machine Learning*. PMLR, 2024.

A Theoretical Properties

A.1 Expressiveness

Any matrix $M \in \mathbb{R}^{n \times n}$ admits a singular value decomposition $M = U\Sigma V^\top$ where U, V are orthogonal and Σ is diagonal. Each orthogonal factor decomposes into at most $\binom{n}{2}$ Givens rotations [Reck et al., 1994]—exactly the primitive GRT uses. A GRT projection chain (Eq. 3) interleaves Givens rotations with diagonal scales, directly mirroring this $U \cdot \Sigma \cdot V^\top$ structure.

The key question is how many stages K suffice. Without dimension shuffling, each stage rotates $d_h/2$ fixed adjacent pairs—K stages with the same pairing yield only block-diagonal transforms. Butterfly shuffling resolves this:

Proposition 1 (Full Coverage)**.** *For d_h dimensions with butterfly shuffle at stride $2^{k \bmod L}$ where $L = \log_2(d_h/2)$, any two dimensions $i \neq j$ are co-rotated in at least one of the $K = L$ stages.*

Proof sketch. Stage k pairs dimension i with dimension $i \oplus 2^{k \bmod L}$ (bitwise XOR on indices within each group). The strides $\{2^0, 2^1, \ldots, 2^{L-1}\}$ generate all single-bit differences, so any two indices differing in bit position b are paired at stage b. For indices differing in multiple bits, information propagates transitively through intermediate stages. □

Although $K = \log_2(d_h/2)$ stages provide $K \cdot d_h/2$ rotation angles—far fewer than the $\binom{d_h}{2}$ needed for a full Givens decomposition—the element-wise scales between stages break orthogonality constraints. The transform $R_2 \cdot \mathrm{diag}(s_1) \cdot R_1$ can represent maps that no single $R \cdot \mathrm{diag}(s)$ can, making the chain strictly more expressive than its angle count suggests. Ma et al. [Ma et al., 2024] independently study efficient Givens parameterizations of $\mathrm{SO}(d)$, supporting the viability of low-K representations. Moreover, the coupling mechanism (§2.6) makes each stage's angles input-dependent, giving each token a *different* effective transform—something static weight matrices cannot achieve.

A.2 Gradient Stability

The Jacobian of a Givens rotation $R(\theta)$ with respect to its input x is $R(\theta)$ itself—an orthogonal matrix with all singular values equal to 1. For the full projection chain, the Jacobian is:

$$\frac{\partial\, \mathrm{proj}_K(x)}{\partial x} = \prod_{k=1}^{K} S_k\, R(\phi_k) \tag{11}$$

where $S_k = \mathrm{diag}(s_k)$. The spectral norm satisfies:

$$\sigma_{\max}\left(\frac{\partial\, \mathrm{proj}_K}{\partial x}\right) \leq \prod_{k=1}^{K} \|s_k\|_\infty \tag{12}$$

Since scales are initialized at 1.0 and regularized by weight decay, the Jacobian spectral norm stays near 1 throughout training—no vanishing or exploding gradients through GRT projections, regardless of the number of stages K.

In dense layers $y = Wx$, the gradient $\partial y/\partial x = W$ has spectral norm $\sigma_{\max}(W)$, which drifts freely during training and requires careful initialization (Xavier, Kaiming) or explicit spectral normalization to control. GRT achieves gradient stability as a *consequence of its parameterization* rather than an initialization heuristic.

The coupling contribution is architecturally bounded by $\tanh$, adding at most ± 1 radian to base angles. This architectural bound ensures content-dependent modulation does not destabilize the rotation chain.

At block boundaries, residual gates with mode `bounded_depth` constrain $g_{\text{attn}}, g_{\text{mix}}, g_{\text{ffn}} \in [0.1, 1.0]$ via $g = 0.55 + 0.45\tanh(\hat{g})$, with raw $\hat{g}$ initialized so that $g = 1/\sqrt{B}$ at step 0. This depth-balances per-block contributions so that the cumulative residual stream is $O(\sqrt{B})$ rather than $O(B)$, complementing the post-block PearlNorm retraction which then holds the residual RMS at the pair-aware normalized value (Table 2).

B Parameter Counting

Embedding. $d/2$ phase angles + 1 log-magnitude per token:

$$P_{\text{embed}} = V \cdot \left(\frac{d}{2} + 1\right) \tag{13}$$

Per-block stored parameters. Rotation angles, scale factors, cross-head mixer, FFN (with optional MoR), and norms. Let $E_{\text{eff}} = \max(E, 1)$, so $E_{\text{eff}} = 1$ when MoR is off ($E = 0$) and $E_{\text{eff}} = E$ otherwise:

$$P_{\text{block}}^{\text{stored}} = \underbrace{6Kd}_{\text{Attention}} + \underbrace{3K_M\, d/2}_{\text{Mixer}} + \underbrace{E_{\text{eff}} \cdot 12Nd}_{\text{FFN}} + \underbrace{[E \geq 2]\, Ed}_{\text{Router}} + \underbrace{2d}_{\text{PearlNorm}} + \underbrace{2}_{\text{Gates}} \tag{14}$$

where attention contributes 4 projections $\times K$ stages $\times h$ heads $\times (d_h/2 \text{ angles} + d_h \text{ scales}) = 6Kd$; the RotationalMixer contributes $K_M \times (d/2 \text{ angles} + d \text{ scales}) = 3K_M d/2$ ($K_M =$ `n_mixer_rotations`); the FFN contributes $E_{\text{eff}} \times 2 \times N \times (d_{\text{ff}}/2 + d_{\text{ff}}) = E_{\text{eff}} \cdot 12Nd$ (with $d_{\text{ff}} = 4d$, and a single shared parameter set when MoR is off); the MoR router adds $E \cdot d$ only when $E \geq 2$ (Iverson bracket $[E \geq 2]$); PearlNorm contributes $2d$ (pre-attention + pre-FFN); residual gates add 2 scalars per block.

Coupling parameters. Cross-pair feature mixing (Clifford, $F = 4$):

$$P_{\text{coupling}}^{\text{block}} = \underbrace{K \cdot d \cdot F \cdot d_h}_{\text{Attention}} + \underbrace{K_M \cdot (d/2) \cdot F \cdot G}_{\text{Mixer (block-diag)}} \tag{15}$$

The attention term covers all 4 projections (Q/K/V/O) $\times K \times h \times (d_h/2) \times (F \cdot d_h/2)$. The mixer term uses block-diagonal coupling at full d_{model} width with group size G (auto-selected, see App. C), reducing the cost from $O(d^2)$ to $O(d \cdot G)$.

Total model parameters.

$$P_{\text{total}} = P_{\text{embed}} + B \cdot \left(P_{\text{block}}^{\text{stored}} + P_{\text{coupling}}^{\text{block}}\right) + d + 1 \tag{16}$$

where the final $d + 1$ accounts for the output PearlNorm and temperature scalar.

Dense equivalent. The parameter count of a standard transformer at matched dimensions:

$$P_{\text{dense}} = V \cdot d + 12 \cdot B \cdot d^2 \tag{17}$$

where $12d^2$ per block $= 4d^2$ (Q/K/V/O) $+ 8d^2$ (FFN with $d_{\text{ff}} = 4d$).

Density. Density $= P_{\text{dense}}/P_{\text{total}}$. GRT models are named by $P_{\text{eff}} = P_{\text{dense}}$.

Verification (SPHERE 200M). $d = 1024$, $h = 16$, $B = 12$, $V = 50$K, $K = 5$, $N = 11$ ($= N_{\max}$ at $d_{\text{ff}} = 4096$). Per-component parameter counts and density ratio are reported in Table 4.

Verified exact against `model.parameters()`: SPHERE nano ($K = 5$, $N = 9$) = 8,686,051; SPHERE 200M ($K = 5$, $N = 11$) = 43,831,669.

C Implementation Details

HuggingFace inference compatibility. The model is exposed as `GRTForCausalLM`, a subclass of HuggingFace `PreTrainedModel` with `GenerationMixin`, so any saved checkpoint can be loaded via `AutoModelForCausalLM.from_pretrained(...)` and used with `model.generate()` (beam search / top-k / top-p / contrastive search) without conversion. Configuration and tokenizer follow the same pattern (`GRTConfig` subclasses `PretrainedConfig`; the BPE tokenizer subclasses `PreTrainedTokenizer`), and checkpoints are saved as safetensors. Inference and downstream fine-tuning thus drop in to existing HuggingFace pipelines unchanged.

Training pipeline. Training is *not* routed through HuggingFace `Trainer` or `accelerate`; the reported runs use a custom training loop (`training/pretrain.py`) built directly on PyTorch (≥ 2.4) with AMP `bf16` autocast on both ROCm and CUDA backends. The training stack is organized as a thin set of utilities in `training/lib/pretrain/`: model construction (`model_builder.py`), parameter-group assembly for PairAdam (`optimizer.py`), data loading (`data.py`), checkpointing (`checkpointing.py`), and the step loop with EMA spike-skip and gradient diagnostics (`training_loop.py`); the PairAdam optimizer itself lives in `training/lib/pair_adam.py`. Datasets are tokenized once and cached to disk (BPE tokenizer trained from scratch on the same corpus, vocab size 50K); FineWeb-Edu `sample-10BT` is loaded via the HuggingFace `datasets` library only as a data source, not as part of the training loop. Mixed-precision is handled by `torch.amp.autocast` with no `GradScaler` (bf16 is numerically stable enough that loss scaling is unnecessary). The full reproducibility checklist, including which scripts run which tests, lives in `REPRODUCIBILITY.md` at the supplementary-zip root.

Fused CUDA/HIP kernel. GRT's rotation+scale stages are inherently sequential: each stage's output feeds the next, so the K (attention) or N (FFN) stages cannot be parallelized across one another. Additionally, the per-pair Givens rotations (2×2 operations on dimension pairs) are too small to utilize GPU tensor cores, which are designed for large matrix multiplies. A pure PyTorch implementation thus incurs K or N separate kernel launches per projection, each underutilizing the hardware.

To address this throughput bottleneck, we provide a fused CUDA/HIP kernel that batches all K (or N) sequential stages into a single kernel launch: each GPU thread processes one dimension pair through all stages, reading angles and scales from shared memory. This eliminates inter-stage launch overhead and keeps the rotation chain entirely in registers. The kernel is JIT-compiled on first call ($\sim$30–60s), automatically detects ROCm vs. NVIDIA, and falls back to the equivalent PyTorch implementation if compilation fails. The fused kernel is a *throughput optimization only*—it computes the same rotation chain as the PyTorch fallback and does not change the architecture or its outputs.

Dimension shuffle patterns. Table 10 compares the available shuffle modes.

Table 10: Dimension shuffle patterns. Illustrative breadth values shown for $d_h = 64$, $d_{\text{ff}} = 2048$ ($K_{\max} = 5$, $N_{\max} = 10$); the same patterns apply at all SPHERE configs.

Mode	Pattern	Attn breadth	FFN breadth	Notes
`none`	fixed pairs	1/5	1/10	no cross-pair mixing
`alternating`	shift 0, 1, 0, 1, …	2/5	2/10	minimum for expressiveness
`butterfly`	bit-reversal	5/5	10/10	**optimal**—full breadth
`fibonacci`	0, 1, 1, 2, 3, 5, …	$\sim$5/5	$\sim$10/10	quasi-periodic, near-optimal

Cross-head mixer. The **RotationalMixer** applies butterfly-shuffled rotation+scale stages with content-dependent Clifford coupling at the full d_{model} width—the same primitive used in attention and FFN projections. Because butterfly strides at full d width pair dimensions across head boundaries, the mixer provides inter-head communication that per-head attention projections cannot. With coupling enabled, different tokens experience different cross-head mixing. GRT uses 1 RotationalMixer stage with Clifford coupling, contributing $\sim$8.6K parameters per block at $d = 256$ (with group size $G = 16$).

Coupling feature modes. GRT supports three coupling feature modes of increasing expressiveness:

- **Energy** ($F = 1$): $S_0[i] = x_{2i}^2 + x_{2i+1}^2$—the squared radius (total intensity) of each pair.
- **Stokes** ($F = 3$): $S_0[i]$ (intensity), $S_1[i] = x_{2i}^2 - x_{2i+1}^2$ (horizontal–vertical polarization), $S_2[i] = 2x_{2i}x_{2i+1}$ (diagonal polarization). These are the first three Stokes parameters from optics, providing a richer signal than intensity alone.
- **Clifford** ($F = 4$): A $\mathrm{Cl}(2,0)$ multivector per pair: scalar $s_i = x_{2i}^2 + x_{2i+1}^2$ (energy), vector components $v_{1,i} = x_{2i}$, $v_{2,i} = x_{2i+1}$, and bivector $b_i = x_{2i}\, x_{2i+1}^{(\text{next})} - x_{2i+1}\, x_{2i}^{(\text{next})}$ (wedge product with the cyclically adjacent pair, where "next" wraps cyclically: pair $P-1$'s neighbor is pair 0). The bivector captures inter-pair phase relationships that Stokes features miss.

Energy ($F=1$) is degenerate on the unit circle: after PearlNorm, all pairs have similar S_0, so coupling receives a nearly flat input. Stokes and Clifford modes break this degeneracy by providing phase-sensitive features. An optional **coupling temperature** provides a learnable per-head scalar τ_h that scales the coupling input before bounding: mod $= \tanh(\tau_h \cdot W_k \cdot \bar{e})$, allowing each head to independently control the saturation regime of the content-dependent modulation. SPHERE nano disables coupling temperature.

Coupling density. Attention projections use *full-dense* coupling: $W_k \in \mathbb{R}^{(d_h/2)\times(F\cdot d_h/2)}$ per stage, so every pair's features can modulate every pair's angle within a head. The RotationalMixer uses *block-diagonal* coupling at the full d_{model} width: pairs are grouped into blocks of size G (auto-selected as the largest of $\{16, 8, 4, 2\}$ dividing $d/2$), and each group's features only modulate that group's angles. This reduces mixer coupling from $O(d^2)$ to $O(d \cdot G)$. The FFN has no coupling: when MoR is active ($E \geq 2$) content-dependence comes from soft expert blending; when MoR is off ($E = 0$, the SPHERE nano default) the FFN is purely position-wise.

Coupling matrices are initialized as $W \sim \mathcal{N}(0, \sigma^2_{\text{coupling}})$ with all feature channels at equal scale, letting the model discover which Clifford components matter during training.

Shuffle is the *wiring diagram*—which dimension pairs can interact (zero parameters, just a permutation). Coupling is the *volume knob*—how much signal actually flows, conditioned on input content. Both are needed: shuffle provides connectivity paths, coupling controls signal through those paths.

Breadth/depth analysis. Table 11 shows the butterfly shuffle sweet spots—configurations where both attention and FFN achieve full breadth with zero depth (no redundant stages). *Breadth* counts unique dimension pairings; *depth* counts stages that revisit existing pairings. Butterfly shuffle scales logarithmically: $K_{\max} = \log_2(d_h/2)$ for attention, $N_{\max} = \lfloor \log_2(d_{\text{ff}}/2) \rfloor$ for FFN. Coupling cost per block follows $K \times d \times d_h$ and depends only on K—adding FFN stages (N) is free in coupling parameters.

Table 11: Butterfly shuffle sweet spots: full breadth, zero depth for both attention and FFN ($d_{\text{ff}} = 4d$, $d_h = 64$). Active configs marked with $\star$.

Model	d	d_h	d_{ff}	K	Attn breadth	N	FFN breadth	Coupling/blk
SPHERE nano*	256	64	1,024	5	5/5	9	9/9	81.9K
SPHERE 200M*	1,024	64	4,096	5	5/5	11	11/11	328K

Beyond $K_{\max}$: depth and expressiveness. At $K = K_{\max}$, every pair of dimensions has interacted exactly once—*full breadth, zero depth*. A full Givens decomposition of an arbitrary matrix in $\mathrm{SO}(d_h)$ requires $\binom{d_h}{2}$ rotations; at $d_h = 64$, $K_{\max} = 5$ provides 160 rotations vs. 2,016 needed (7.9% coverage). Stages beyond $K_{\max}$ revisit existing pairings, adding *depth*—a second pass applies different angles with intervening scales that break orthogonality, so $R(\theta') \cdot \mathrm{diag}(s) \cdot R(\theta)$ is strictly more expressive than either rotation alone. The current design prioritizes clean geometry ($K = K_{\max}$); exploring loss vs. K beyond this point is future work.

Initialization. Table 12 gathers the initialization scales for every learnable parameter group. Embedding phase angles are initialized small to start tokens near a common reference point on the pair manifold, and embedding log-magnitudes start at zero (unit pair radius). Block rotation angles use the same small-init convention, which keeps the network close to the identity transform at step 0 so that PearlNorm normalization dominates pair geometry early on. Coupling matrices initialize even smaller ($\mathcal{N}(0, 0.01^2)$) so that the content-dependent angle perturbation $\Delta\theta = W \cdot f(x)$ is negligible at start; coupling becomes meaningful only as W trains. Scales (softplus-parameterized) initialize to raw $\hat{s} = \mathrm{softplus}^{-1}(1.0) \approx 0.541$, so the multiplicative scale starts at exactly 1.0 (identity), consistent with the near-identity rotation init. The cosine-softmax temperature initializes at $\hat{s} = 30$ (linear regime of softplus, so $s \approx 30$), placing init logits in the broadband softmax regime rather than the saturation regime.

D PairAdam Optimizer

GRT's rotational parameterization requires geometry-aware optimization [Householter, 2026b]. Standard AdamW treats every parameter element independently, but GRT has three structurally distinct parameter groups with different gradient dynamics and manifold constraints. **PairAdam** extends AdamW's update rule with three modifications:

Parameter	Init	Config key
Embedding phase angles	$\mathcal{N}(0, 0.1^2)$	`phase_init_scale`
Embedding log-magnitude	0.0	`log_magnitude_init`
Block rotation angles	$\mathcal{N}(0, 0.1^2)$	`angle_init_scale`
Block scales (raw)	$\hat{s} = \text{softplus}^{-1}(1.0) \approx 0.541$ (softplus $\to 1.0$)	—
Coupling matrices	$\mathcal{N}(0, 0.01^2)$, diagonal pattern	`coupling_init_scale`, `coupling_init_mode`
Cosine-softmax temperature	$\hat{s} = 30$ (linear softplus regime)	`output_scale_init`
RoPE log-frequencies	standard RoPE schedule $\log(10000^{-2i/d})$	`pos_freq_base`
Residual gates	$\hat{g}$ such that $g = 1/\sqrt{B}$	`residual_gate_mode = bounded_depth`
PearlNorm scales (all positions)	$\mathbf{1}$	—

Table 12: Initialization values for every learnable parameter group. SPHERE nano and SPHERE 200M use these defaults.

1. Factored second moments for coupling matrices. Coupling matrices $W_k \in \mathbb{R}^{(d_h/2)\times(F\cdot d_h/2)}$ are the only dense-like component. Standard Adam stores per-element second moments v_{ij} at $O(mn)$ cost. PairAdam instead maintains *row factors* $r_i = \mathbb{E}[g^2_{i,:}]$ and *column factors* $c_j = \mathbb{E}[g^2_{:,j}]$, reconstructing the second moment as $\hat{v}_{ij} = r_i \cdot c_j / \bar{r}$ (outer product with mean normalization). This reduces memory from $O(mn)$ to $O(m+n)$ while capturing the bipartite structure of coupling interactions: row factors track how much each output pair learns, column factors track input pair learning rates. This factorization is natural for GRT's coupling matrices, which mediate structured interactions between feature pairs.

2. Angular step clipping. Rotation angles live on S^1; large updates can overshoot meaningful angular distances. PairAdam clips angle updates to a maximum step size $|\Delta\theta| \leq \theta_{\max}$ per step (SPHERE nano and SPHERE 200M use $\theta_{\max} = 0.5\,\text{rad} \approx 28.6°$, set via the `max_angular_step` config key). This prevents the optimizer from "jumping" across the angle manifold during loss spikes.

3. S^1 modular wrapping. After each update, rotation angles are wrapped to $[-\pi, \pi]$ via $\theta \leftarrow (\theta + \pi) \bmod 2\pi - \pi$, keeping parameters in the canonical range. Without wrapping, angles drift unboundedly despite representing equivalent rotations modulo 2π, which degrades Adam's moment estimates.

Parameter groups. PairAdam assigns each parameter to a group with tailored settings:

Table 13: PairAdam per-group settings. Each group has distinct gradient dynamics requiring separate treatment.

Setting	Angles	Scales	Coupling
Peak LR	10^{-3}	3×10^{-4}	2×10^{-4}
Adam ϵ	10^{-8}	10^{-8}	10^{-8}
Weight decay	0.0	0.01	0.01
2nd moments	diagonal	diagonal	factored
Angular clip	0.5 rad	—	—
S^1 wrap	✓	—	—
Init	$\mathcal{N}(0, 0.1)$	1.0 (softplus)	$\mathcal{N}(0, 0.01)$

Rotation angles (θ): Gradients flow through $\cos\theta$ and $\sin\theta$, so gradient magnitude is bounded by 1 regardless of activation scale. No weight decay (angles have no natural "zero" to regularize toward). S^1 wrapping and angular clipping are enabled.

Scale factors (s): Parameterized via softplus ($s = \log(1 + \exp(\hat{s}))$), initialized at 1.0 (identity). The softplus gradient $\sigma(\hat{s}) \in (0, 1)$ naturally damps updates. Standard diagonal moments with weight decay toward 1.0.

Coupling matrices (W_k): Factored second moments capture the bipartite structure. With Clifford features ($F = 4$) and small initialization ($\mathcal{N}(0, 0.01)$, set via `coupling_init_scale`), coupling gradients stay comparable in magnitude to angle gradients during early training.

Per-group LR multipliers. GRT's parameter groups span very different effective dimensionalities (a d_{model}-wide embedding angle table vs. a single learnable `output_scale` scalar) and play structurally different roles (dense rotations vs. output-side temperature). PairAdam supplies a per-group LR multiplier μ_g such that each group's effective LR is $\mu_g \cdot \eta_{\text{base}}$. SPHERE nano uses $\mu_g = 1.0$ for every group except RoPE ($\mu_{\text{rope}} = 0.1$, the

standard convention for positional-encoding parameters). At larger scale the multipliers `scale_lr_mult`, `output_scale_lr_mult`, and `log_magnitude_lr_mult` are dialed down: SPHERE 200M uses 0.3, 0.1, 0.1 respectively (Tab. 14); `coupling_lr_mult` and `embedding_lr_mult` remain at 1.0 in both configurations.

D.1 Training details

Table 14 lists the recipe parameters for both SPHERE configurations. Items not listed default to their PyTorch / HuggingFace conventions; full configs are in `training/configs.py`. Both runs use FineWeb-Edu `sample-10BT` [Penedo et al., 2024], a BPE tokenizer trained from scratch on the same data, and AMP mixed-precision; sequence length is 512 for SPHERE nano and 1024 for SPHERE 200M.

Setting	SPHERE nano	SPHERE 200M
Schedule		
Peak LR (`lr`)	10^{-3}	5×10^{-4}
LR schedule	cosine	cosine
Min LR ratio (`min_lr_ratio`)	0.1	0.1
Warmup steps (`warmup_steps`)	5000	5000
Total steps (`max_steps`)	100000	100000
Batching		
Microbatch (`batch_size`)	64	32
Grad accum (`gradient_accumulation_steps`)	2	2
Effective batch	128	64
Tokens / optim step	65,536	65,536
Regularization & clipping		
Global gradient clip (`max_grad_norm`)	1.0	1.0
Per-group clip mults (`{angle,scale,coupling}_grad_clip_mult`)	1/1/1	1/1/1
Per-group LR mults (`{scale,output_scale,log_magnitude}_lr_mult`)	1/1/1	0.3/0.1/0.1
Angle weight decay	0	0
Scale / coupling weight decay (`weight_decay`, `coupling_weight_decay`)	0.01	0.01
Per-step angle clip (`max_angular_step`, rad)	0.5	0.5
Per-step scale clip (`max_scale_step`)	—	0.05
z-loss weight (`z_loss_weight`)	10^{-4}	10^{-4}
Loss-spike factor (`spike_factor`, default)	3.0	3.0
Grad spike-skip threshold ($\times$ EMA)	100	5.0
Max consecutive skips (`spike_max_consec_skips`, default)	10	10
Architecture knobs		
Cosine head s init (`output_scale_init`)	30.0	30.0
RoPE base (`pos_freq_base`)	10000	10000
Learnable RoPE (`learnable_rope`)	✓	✓
Post-block PearlNorm (`residual_norm_mode`)	✓	✓
Gate mode (`residual_gate_mode`)	bounded_depth	bounded_depth
Mixer rotations (`n_mixer_rotations`)	1	1
Coupling features (`coupling_features`)	Clifford ($F=4$)	Clifford ($F=4$)
Peri-LN (`peri_ln`)	off	off
MoR experts (`ffn_n_experts`)	0 (dense)	0 (dense)
Data		
Corpus	FineWeb-Edu `sample-10BT`	FineWeb-Edu `sample-10BT`
Documents (`max_documents`)	6,000,000	12,000,000
Tokens consumed (100K steps)	6.55B	6.55B
Hardware		
GPU	1$\times$ AMD Radeon 8060S (Strix Halo)	1$\times$ NVIDIA GH200 (96 GB)
Sequence length (`seq_len`)	512	1024
Mixed precision	AMP (bf16)	AMP (bf16)

Table 14: Training recipe for the two reported *dense* configurations. The dense FFN ($E=0$) is the simpler of two configurations covered by the SPHERE family; **Mixture-of-Rotations** runs ($E\geq 2$) are reported in the MoR component paper [Householter, 2026a] and use the same recipe with the addition of MoR-specific hyperparameters. Items not listed default to PyTorch conventions. Optional features for future work (`pc_aux`, `deep_supervision`, Ouroboros refinement) are described in Appendix **??**. Full configs in `training/configs.py` (`nano_v1` and `200m` keys).

Token budget and reference points. SPHERE nano consumes $\sim$6.55B training tokens over 100K steps on FineWeb-Edu `sample-10BT` (effective batch 128, seq_len 512), reaching val loss 4.661 at 17.5M dense-

equivalent parameters (8.7M stored under the compressed parameterization; Tab. 5). As a public reference point, the modded-nanoGPT speedrun[2] reaches FineWeb val loss 3.28 at 124M parameters in $\sim$1.7B tokens.

Model	Params	Tokens	Data	Val loss
modded-nanoGPT	124M	1.7B	FineWeb	3.28
SPHERE (this work)	17.5M (8.7M stored)	6.55B	FineWeb-Edu	4.661

The two numbers differ in tokenizer, data filter, and $\sim 7\times$ capacity — they are not a controlled comparison. Reading them as one: SPHERE's loss sits in the expected range for its size, and its compressed parameterization trains end-to-end without loss-curve pathology. The like-for-like 17.5M dense baseline at matched FLOPs and tokens is the comparison that would settle the parameter-efficiency question, and is deferred to a follow-up scaling study.

z-loss. GRT adds a small logsumexp-squared penalty $\lambda_z \cdot (\log Z(x))^2$ to the cross-entropy objective, where $Z(x) = \sum_v \exp(\ell_v(x))$ is the partition function [Chowdhery et al., 2023]. The penalty stabilizes softmax under cosine-similarity logits by discouraging large $\log Z$, which translates to bounded peak logits. We use $\lambda_z = 10^{-4}$ in all configurations.

Gradient clipping. We clip the global pre-step gradient norm to `max_grad_norm`$= 1.0$ in addition to PairAdam's per-element angular and scale clips. Per-parameter-group clip multipliers (`angle_grad_clip_mult`, `scale_grad_clip_mult`, `coupling_grad_clip_mult`) are configurable but kept at 1.0 in all reported runs. The training loop also maintains an EMA of the global gradient norm and skips the optimizer step when the current norm exceeds `spike_skip_threshold` times that EMA (default $100\times$, used in all reported runs), with a deadlock-breaker after `spike_max_consec_skips`$= 10$ consecutive skips that re-anchors the EMA to the current norm. A separate `spike_factor`$= 3.0$ guards against pure loss spikes (independent of the gradient EMA): when the running loss EMA is exceeded by this factor and we are past warmup, the step is skipped.

E Notation

Symbol	Meaning
Architecture dimensions	
d	Model dimension (d_{model})
h	Number of attention heads
d_h	Head dimension (d/h)
d_{ff}	FFN intermediate dimension ($4d$ in SPHERE configs)
V	Vocabulary size
B	Number of transformer blocks
Stage and feature counts	
K	Number of rotation+scale stages in attention (Q/K/V/O)
$K_{\max}$	$\log_2(d_h/2)$ — stage count for full butterfly coverage
K_M	Number of stages in the RotationalMixer (`n_mixer_rotations`)
N	Number of rotation stages in each FFN phase (expand, contract)
E	Number of MoR experts (0 = dense FFN, $\geq$2 = soft-routed MoR)
F	Coupling feature multiplier (1 energy, 3 Stokes, 4 Clifford)
G	Block-diagonal coupling group size in the mixer
Learnable parameters	
θ	Rotation angle, wrapped to $[-\pi, \pi]$
s	Scale factor, $s = \log(1 + \exp(\hat{s}))$ via softplus, init 1.0
W_k	Coupling matrix at stage k, shape $(d_h/2) \times (F \cdot d_h/2)$
τ_h	Optional per-head coupling temperature (off in SPHERE nano)

[2] https://github.com/KellerJordan/modded-nanogpt

F Related work: detailed comparisons

We expand the related-work summary in §5 with line-by-line comparisons against the most directly related lines of work.

Givens rotations in neural networks. Choromanski et al. [Choromanski et al., 2019] (KAMA-NNs) replace dense weight matrices with products of k Givens rotations, achieving $O(d \log d)$ parameters per layer with random or learned index pairs. Jing et al. [Jing et al., 2017] parameterize unitary RNN transitions as Givens products with a butterfly-patterned pairing schedule—the same FFT-style stride pattern GRT adopts for dimension shuffling. Frerix and Bruna [Frerix and Bruna, 2019] study efficient Givens factorizations for approximating orthogonal matrices, finding that fixed index patterns limit expressiveness; GRT avoids this by using deterministic butterfly pairings that guarantee full coverage and learning angles directly rather than approximating pre-existing dense matrices. These prior works apply Givens rotations to individual components (MLPs, RNN transitions). GRT extends the primitive to the *entire* transformer—embeddings, attention projections, FFN, and output head—with deterministic butterfly coverage and content-dependent coupling.

Orthogonal fine-tuning. OFT [Qiu et al., 2023] applies block-diagonal orthogonal transforms during fine-tuning, preserving hyperspherical energy (pairwise angular distances between neurons). BOFT [Liu et al., 2024] reduces the parameter cost from $O(d^2)$ to $O(d \log d)$ via butterfly-factorized 2×2 orthogonal blocks—structurally similar to GRT's butterfly-shuffled Givens stages. Ma et al. [Ma et al., 2024] (qGOFT) study efficient Givens-based parameterizations of $\mathrm{SO}(d)$, further validating low-rank rotational representations. These methods adapt pre-trained dense weights; GRT eliminates dense weights entirely and trains from scratch.

Structured matrices. Dao et al. [Dao et al., 2019] show that products of butterfly matrices can represent any linear transform with $O(n \log n)$ parameters instead of $O(n^2)$. Monarch [Dao et al., 2022] extends this to training. GRT adopts butterfly permutations for dimension shuffling but replaces block-diagonal matrices with Givens rotations and element-wise scales—a more constrained parameterization that preserves geometric interpretability. Gorbunov et al. [Gorbunov et al., 2024] study structured orthogonal parametrization with group-and-shuffle patterns related to our shuffle strategies.

Orthogonal parameterization and gradient stability. Unitary evolution RNNs [Arjovsky et al., 2016] constrain recurrent matrices to the unitary group to prevent vanishing and exploding gradients, and Saxe et al. [Saxe et al., 2014] show orthogonal initialization avoids exponentially slow saddle-point plateaus. GRT's Givens parameterization maintains orthogonality throughout training at $O(d \log d)$ cost with individually interpretable angle parameters.

Hyperspherical and manifold-constrained networks. Liu et al. [Liu et al., 2017] introduce SphereConv, replacing $y = W \cdot x$ with $y = f(\theta_{x,W})$ where θ is the angle between input and weight (linear, cosine, and sigmoid variants), and prove tighter Lipschitz bounds than standard CNNs at depth. SimbaV2 [Lee et al., 2025] replaces all LayerNorm with L_2-normalization, projects weights to the unit hypersphere after each gradient step, and uses LERP convex combinations in place of additive residuals to control feature/weight norm growth in deep RL—a motivation directly aligned with ours. The formal theory of optimization on smooth manifolds via retractions is developed by Absil, Mahony, and Sepulchre [Absil et al., 2008]; the residual-update + PearlNorm pattern in GRT is a first-order retraction in this framework. Liu's SphereConv applies angle-based ops layer-wise (CNNs), and SimbaV2 imposes a global L_2 constraint (RL, no pair structure); GRT generalizes the idea to a *pair-rotational* manifold where every operation is a rotation+scale on $(\cos, \sin)$ pairs, with PearlNorm as the retraction that closes the geometry across depth.

Cross-head mixing. Talking-Heads attention [Shazeer et al., 2020] inserts learned $h \times h$ projections before and after softmax, mixing information across attention heads at negligible cost. DCMHA [Xiao et al., 2024] extends this with input-dependent $h \times h$ mixing matrices. Both operate on the small head-count axis (typically $h = 8$–32), not the d-dimensional representation. GRT's RotationalMixer (§2.8) provides cross-head mixing in d-space via butterfly-shuffled Givens rotations with content-dependent coupling—butterfly strides at the full d_{model} width pair dimensions across head boundaries by construction. The mechanism is complementary: Talking-Heads mixes head-level attention patterns, while the RotationalMixer mixes the representation dimensions that belong to different heads.

Photonic interferometers. GRT's architecture has a direct structural correspondence with photonic interferometer meshes [Reck et al., 1994, Clements et al., 2016]: each Givens rotation is a phase shifter, butterfly shuffle defines the mesh topology, and coupling acts as an electro-optic modulator making phase shifts input-dependent. Both systems achieve universality by composing embedded 2D rotations with a fixed interconnect.

G Implementation correctness

GRT is implemented in PyTorch with fused CUDA/HIP kernels for the hot-path operations (Givens butterfly rotation, Clifford coupling, PearlNorm, RadialGELU, mixer, fused embedding, fused z-loss). To ensure that the kernels compute the same mathematical operation as the eager-mode reference, we maintain a parity-test suite (`notebooks/test_*_parity.py`) which, for each kernel, constructs a small synthetic input on GPU, runs forward and backward through both the fused kernel and the eager fallback, and asserts element-wise gradient agreement to floating-point precision. A companion script (`notebooks/compare_kernel_eager.py`) performs the same comparison at the full-model level via two subprocesses with the kernel toggle (`GRT_NO_KERNEL=1`) flipped.

Beyond kernel parity, the supplemental material includes targeted unit tests for optimizer plumbing (`test_param_group_assignment.py`, `test_pair_adam_scale_clip.py`) and a parameter-state mutation script (`diagnose_drift_unit_test.py`) that applies single perturbations to a freshly-initialized model and measures the resulting gradient response.

These tests are run before any reported training run; passing them is a precondition for results in this paper. The supplemental `REPRODUCIBILITY.md` enumerates each test, what it verifies, and how to run it.

H Empirical checkpoint analysis

We probe the trained SPHERE nano checkpoint to verify that the architectural design choices manifest in the learned model state. The analysis script (`notebooks/analyze_checkpoint.py`) loads a checkpoint, computes per-block, per-stage statistics on the state dict, runs a forward pass with hooks for residual-stream RMS measurement, and emits a JSON summary alongside the figures shown below. Numbers reported here are from `sphere-nano/checkpoint-100000` (100K training steps into a 100K run).

Training curve. Figure 3 shows validation loss and accuracy through the full 100K steps. Loss descends monotonically from 7.68 at step 2K to 4.66 at step 100K; validation accuracy reaches 0.259.

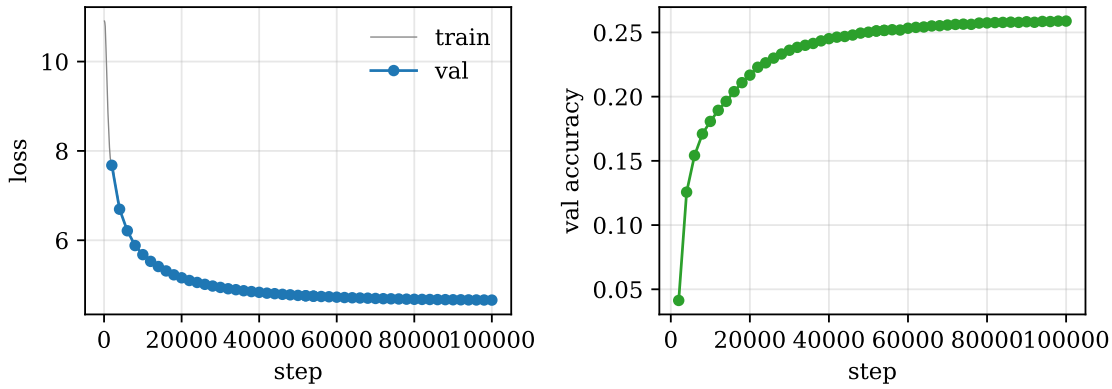

Figure 3: SPHERE nano training: validation loss (left) and validation accuracy (right) through 100K steps. Eval cadence is every 2K steps. Final values: val_loss 4.661, val_acc 0.259, val_ppl 105.7.

Residual stream RMS at depth. Figure 4 plots the per-block residual-stream RMS at this trained state, measured by a forward pass on random tokens. With the post-block PearlNorm in place, RMS stays in a

tight $0.46-0.52$ band across all six blocks; the trained PearlNorm scale parameters compress the per-pair output below the init target of $1/\sqrt{2} \approx 0.707$. Either way, RMS is bounded across depth, in contrast to the unbounded $1.5 \to 5.3$ growth that would occur without the post-block normalization (Table 2).

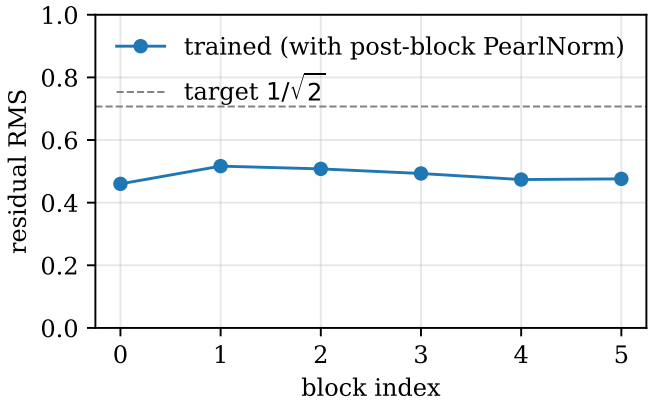

Figure 4: Residual-stream RMS measured at each block's output, trained SPHERE nano at step 100K. Forward pass on random tokens. The dashed line marks the target value $1/\sqrt{2}$ that the post-block PearlNorm would emit if scales were exactly identity. The trained scales depart slightly from 1, so the measured values cluster around but do not match the target exactly.

Butterfly stage activation. Figure 5 reports the per-stage rotation-angle standard deviation, averaged across blocks, for the four attention projections. All five butterfly stages depart from the initialization scale of 0.10, indicating each stride contributes to the learned transform. The pattern is consistent across Q, K, V, O.

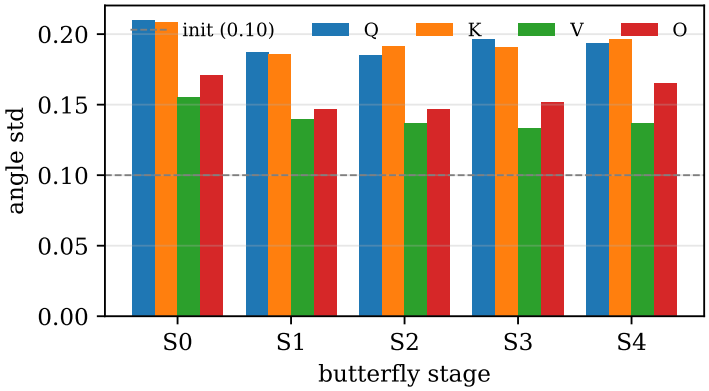

Figure 5: Per-stage attention angle std at trained state, averaged across 6 blocks. The dashed line shows the initialization value ($\sigma = 0.1$, `angle_init_scale`). All five butterfly stages develop non-trivial structure under training.

Residual gate trajectory. Figure 6 shows the per-block residual gate values ($g_{\text{attn}}, g_{\text{mix}}, g_{\text{ffn}}$) at the trained state. Gates are initialized at $1/\sqrt{B} \approx 0.408$ for $B = 6$ (dashed line); after 100K steps the mixer gates have learned a striking U-shaped depth profile: blocks 0–1 and block 5 reach $g_{\text{mix}} \in [0.94, 0.99]$ while middle blocks 2–4 sit at $g_{\text{mix}} \approx 0.7$. Attention and FFN gates also move from init, with the final block's gates rising more than middle blocks'. The `bounded_depth` parameterization keeps all gates in $[0.1, 1.0]$.

Coupling matrix growth. Figure 7 plots the maximum-absolute coupling matrix entry per block. From an init scale of $\sigma_c = 0.01$ with diagonal bias $10\sigma_c = 0.1$, the trained coupling magnitudes have grown by roughly

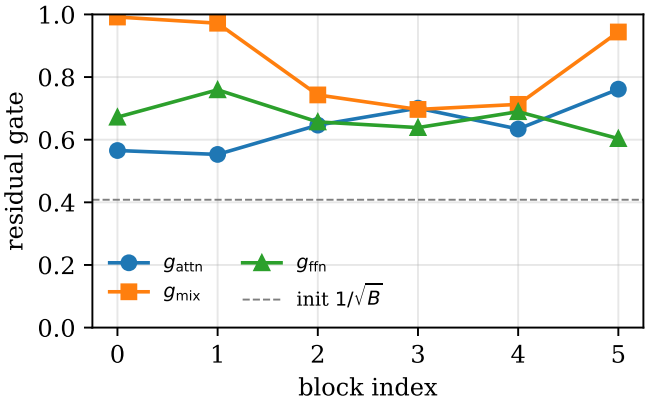

Figure 6: Residual gate values per block at trained state. Dashed line: initialization at $1/\sqrt{B}$. The `bounded_depth` parameterization holds all gates in $[0.1, 1.0]$.

two orders of magnitude in attention. This indicates the content-dependent angle modulation mechanism, which is initialized to be small, has become active during training.

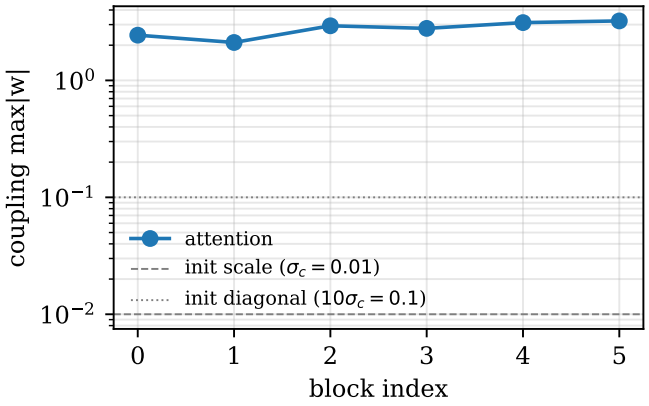

Figure 7: Coupling-matrix max absolute entry per block, log scale. Dashed line: random-init scale ($\sigma_c = 0.01$). Dotted line: diagonal-bias init ($10\sigma_c = 0.1$). After 100K steps coupling magnitudes reach $2.1-3.2$, two orders of magnitude above init.

Embedding distribution. The embedding mean resultant length R_p (a standard isotropy measure: 0 for uniform on the sphere, 1 for collapsed) decreases monotonically through training, from 0.995 near initialization to 0.866 at step 100K. This is consistent with the small initial phase scale (`phase_init_scale` $= 0.1$); larger init values (e.g. π) produce more isotropic embeddings at the cost of higher initial loss. The embedding-isotropy/loss tradeoff is left for future ablation.

Vocabulary on the unit circle. Figure 8 visualizes the embedding distribution directly: each token's $(\cos\theta_t, \sin\theta_t)$ coordinates are plotted at two pair indices, a randomly-chosen pair (left) and the pair with the largest learned variance (right). The right panel shows that on the most-active pair the model has learned to spread tokens broadly along the circle; the left panel shows that random pairs still exhibit a clustering near a preferred angle, consistent with the high R_p value above. The geometry of token storage is fundamentally angular, not Euclidean.

PearlNorm scale distributions. Figure 9 shows the histogram of per-element PearlNorm scale parameters at four positions in the network: pre-attention (`norm1`), pre-mixer, pre-FFN (`norm_ffn`), and post-block

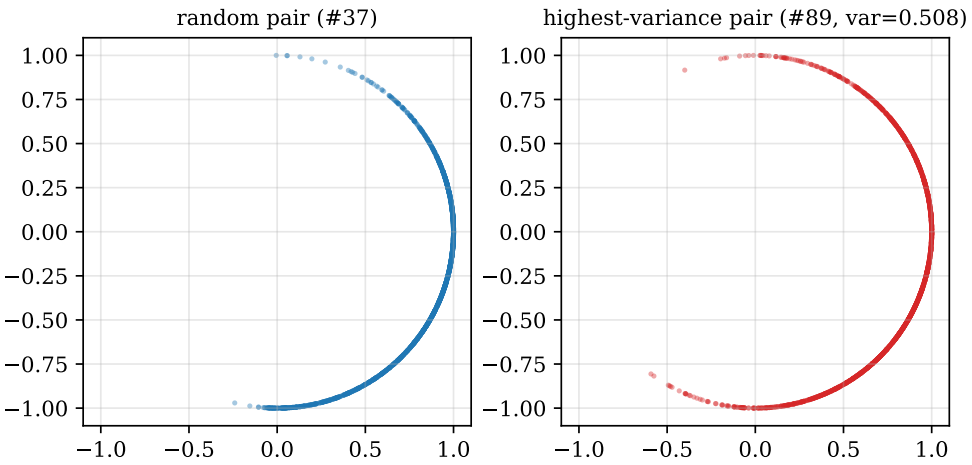

Figure 8: Token embeddings projected to two phase pairs. Each point is a vocabulary token at coordinates $(\cos\theta_t, \sin\theta_t)$ within the indicated pair index. Left: a randomly chosen pair. Right: the pair with the largest learned variance.

(`res_norm_block`). All four start at 1.0 at initialization; training pushes them toward block-and-position specific distributions. The post-block (**R**etraction) PearlNorms develop the broadest distribution, consistent with their role of correcting per-pair drift across all $d/2$ pairs.

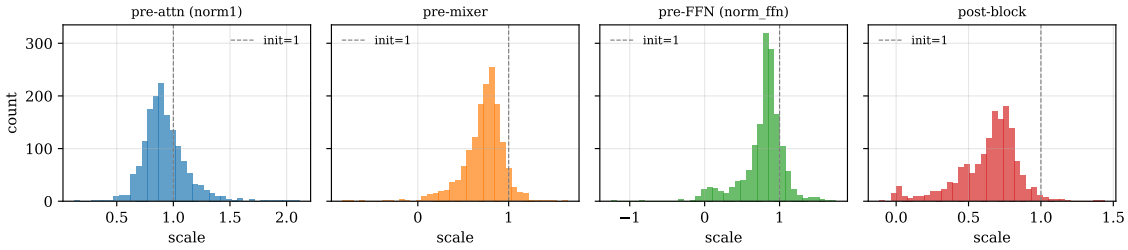

Figure 9: Histograms of trained PearlNorm scale parameters at four positions in the network, aggregated across all blocks. Dashed line: initialization value **1**.

Logit lens: where do predictions form? Figure 10 applies the tied-embedding cosine head to each block's hidden state and reports two diagnostics: top-1 prediction agreement with the final block (left), and the entropy of the head's softmax distribution (right). Top-1 agreement is at noise floor (~ 0.001) for blocks 0 through 4 and rises to 0.006 at block 5; per-block KL-to-final ranges over 10.5–10.9 nats across all six blocks (uniform baseline $\log V \approx 10.8$ nats). Decoding is therefore concentrated at the final block — early-block residuals carry information that is not aligned with the unembedding basis until block 5 rotates it into the readout subspace. Linear probes (Tab. 16 and accompanying text) recover non-trivial next-token accuracy ($> 10\%$) at every block, indicating the information is present throughout the stack but in a basis the fixed cosine head cannot read until the final block.

Attention entropy across heads and depth. Figure 11 shows the per-head causal attention entropy averaged over query positions, computed at the trained state. Some heads exhibit lower entropy (sharper, more selective routing) and others higher (broader, copy-style). The depth-wise pattern is two-phase rather than monotone: per-block mean entropy rises from 3.16 bits at block 0 to 3.73 at block 1 before contracting through blocks 3–5 to 1.82 bits at the final block (Tab. 16). Block 5 contains the sharpest single head observed, with mean entropy 0.31 bits — effective key support $2^{0.31} \approx 1.24$, i.e. on average each query places more than 80% of its mass on a single key. The seq-len upper bound is $\log_2 128 = 7$ bits.

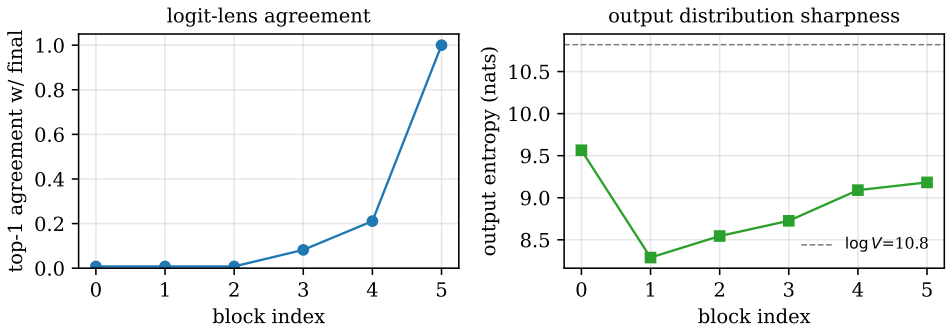

Figure 10: Logit-lens analysis: applying the tied output head to each block's residual stream. Left: top-1 prediction agreement with the final block's prediction. Right: per-block softmax entropy; the dashed line is $\log V$ (the uniform-distribution upper bound).

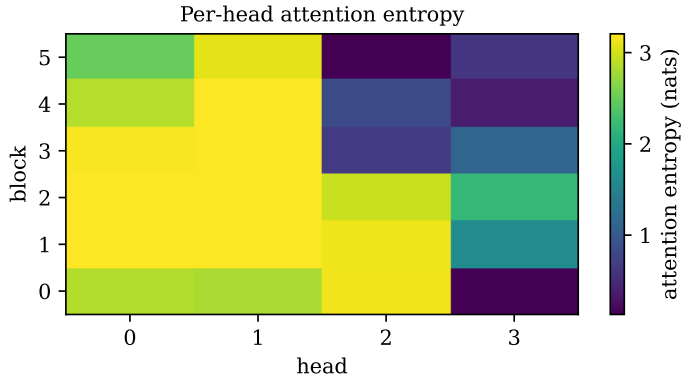

Figure 11: Per-head attention entropy (in nats) at trained state. Each cell is one (block, head) pair. Higher entropy = broader attention; lower = more selective. No head is fully collapsed to a delta.

Learned RoPE frequencies. Figure 12 compares the trained RoPE frequencies (`learnable_rope=True`) to the canonical RoPE schedule $\omega_i = 10000^{-2i/d}$. The frequencies have moved noticeably from init: high-frequency pairs (low pair index) have shifted, while low-frequency pairs follow the schedule more closely. The model is adapting positional resolution per dimension pair.

Per-stage scales. Figure 13 reports the trained scale magnitudes (softplus of raw parameters) per butterfly stage, averaged across blocks. Like angle stds (Fig. 5), all five stages have departed from initialization ($s = 1.0$), indicating each stage's per-dimension scaling contributes to the learned transform.

Block representation similarity. Figure 14 shows the cosine similarity between mean-pooled block outputs. Adjacent blocks have high similarity (≥ 0.9), and similarity decays smoothly with block separation. The block representations evolve continuously through the stack rather than partitioning into distinct "stages," consistent with the residual-stream view of transformer computation.

Pair radii through depth. GRT's geometric invariant is that activations live on the pair manifold $\mathcal{M} = (S^1)^{d/2}$ (modulo per-pair magnitude). Figure 15 shows the empirical distribution of pair radii $r_i = \sqrt{x_{2i}^2 + x_{2i+1}^2}$ at each block's output. The post-block PearlNorm holds the radius distribution close to a single peak around $r \approx 1$ throughout the stack, in contrast to the unbounded spread that would result without it. This is the geometric counterpart to the residual-RMS measurement in Figure 4, but resolved at the per-pair level.

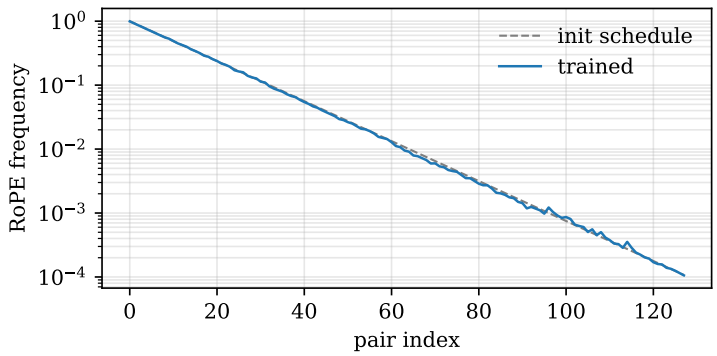

Figure 12: RoPE frequency per pair index, log scale. Dashed: canonical init schedule $\omega_i = 10000^{-2i/d}$. Solid: trained values.

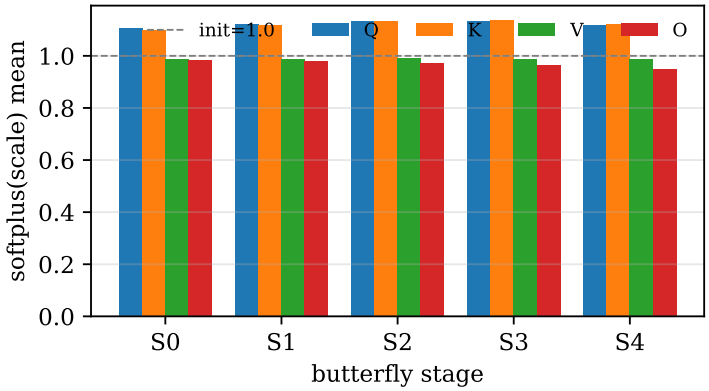

Figure 13: Per-stage scale magnitudes ($\mathrm{softplus}(\hat{s})$), mean across blocks, for the four attention projections. Dashed: initialization at 1.0.

Output phase distribution. Figure 16 shows a polar histogram of phase angles $\varphi_i = \mathrm{atan2}(x_{2i+1}, x_{2i})$ at the final block, pooled across all pairs and positions. A uniform distribution would indicate that the model uses the full circle as a representational basis; pronounced peaks would indicate angular collapse. The trained distribution is broadly populated with non-trivial angular structure, indicating the model uses the full circle without collapsing onto a small set of modes.

Per-block residual write magnitude. Figure 17 reports the relative magnitude of each block's residual update, $\|\Delta x\|/\|x\|$, averaged over batch and positions. All blocks contribute non-trivially, with the final block making the largest update; no block appears redundant.

Per-block prediction examples. Figure 18 renders the logit-lens decoded top-1 predictions for a fixed input through every block. Cells highlighted in green indicate cases where the prediction matches the input token exactly. Predictions become more diverse and contextually plausible across the stack, with the final block's predictions reflecting the FineWeb-Edu corpus prior (factual / encyclopedic content) rather than collapsing onto a single mode.

Diagnostic scalars at trained state. Tables 15 and 16 report internal-state diagnostics computed on the SPHERE nano 100K checkpoint over a 32-sentence validation batch (seq_len 128). Numbers are reported without further interpretation; thresholds in the column headers are indicative ranges from prior work on small dense transformers and are included only as anchors.

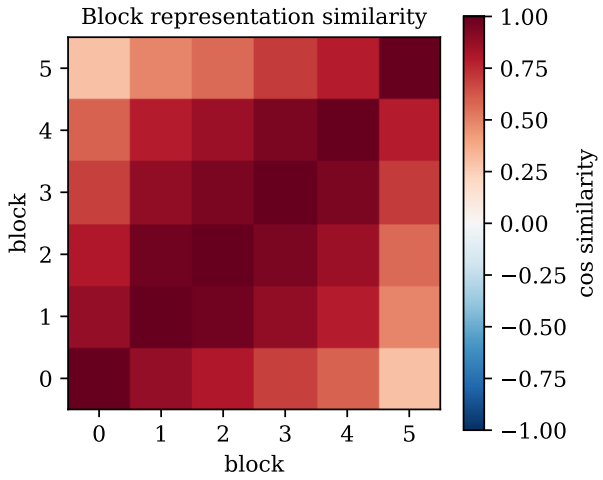

Figure 14: Pairwise cosine similarity between mean-pooled block-output representations. Bright diagonal = adjacent blocks are similar; off-diagonal smoothness = continuous evolution of representations through depth.

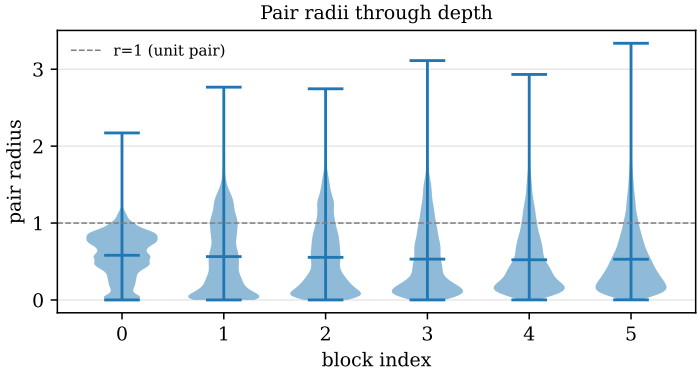

Figure 15: Distribution of pair radii r_i at each block's output, trained SPHERE nano at step 100K. Each violin pools all $d/2$ pairs $\times$ all positions $\times$ batch. Dashed line: $r = 1$ (unit pair).

Cosine floor matches angular concentration. Two embeddings stored as 128 phase pairs at near-unit radius have per-pair dot $\cos(\alpha_i - \beta_i)$. For independent tokens drawn from a per-pair circular distribution with resultant length R_i, $\mathbb{E}[\cos(\alpha_i - \beta_i)] = R_i^2$, and the aggregate prediction is $\mathbb{E}[\cos(a, b)] \approx \overline{R^2}$. With $R_p = 0.865$ this predicts $\overline{R^2} = 0.751$; the measured pairwise cosine on a vocabulary sample is 0.750 ± 0.001 (Fig. 19). The mean cosine is fully accounted for by the angle distribution's concentration; it is not a structural floor of the storage manifold.

Per-block attention entropy. Mean per-row attention entropy $H_{b,h} = \mathbb{E}_{i \geq 8}[-\sum_j A_{b,h,i,j} \log_2 A_{b,h,i,j}]$ in bits, computed over 32 sentences at sequence length 128. Every trained head sits below the uniform baseline $\log_2(128) = 7$ bits (max 4.17 at block 2, head 1). Block 5, head 3 has $H = 0.307$ bits, an effective key support of $2^{0.307} \approx 1.24$ keys — a near-deterministic head (Fig. 20).

Pair-radius RMS is isotropic at every block. Let $r_b = \sqrt{\mathbb{E}[h^2]}$ be per-coordinate residual RMS at block b, and $\rho_b = \sqrt{\mathbb{E}[h_{2k}^2 + h_{2k+1}^2]}$ the pair-radius RMS treating consecutive coordinates as a 2D pair. The ratio ρ_b / r_b ranges over $\{1.4138, 1.4144, 1.4144, 1.4142, 1.4142, 1.4143\}$ across the six blocks, all within 4×10^{-4}

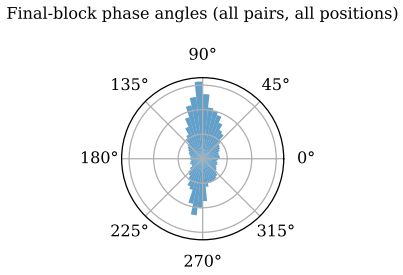

Figure 16: Polar histogram of phase angles $\varphi_i = \mathrm{atan2}(x_{2i+1}, x_{2i})$ at the final block, all pairs and positions pooled.

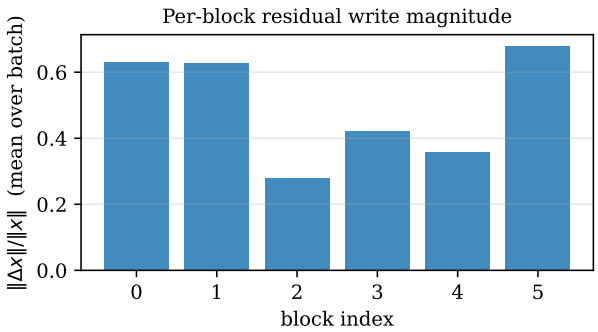

Figure 17: Per-block residual write magnitude $\|\Delta x\|/\|x\|$ at trained state.

of $\sqrt{2} \approx 1.4142$ (Fig. 21). The pair-aligned concentration alternative ($\rho_b/r_b \to 1$) is decisively ruled out: pair-coordinate energy is isotropic, not concentrated.

Decoding is concentrated at the final block. Logit-lens top-1 agreement with the final-block prediction is at noise floor ($\sim 10^{-3}$) for blocks 0–4 and rises to 0.006 at block 5; per-block KL-to-final is flat at 10.5–10.9 nats across all six blocks (uniform baseline $\log V \approx 10.82$ nats). No intermediate residual is decodable through the unembedding (Fig. 22). Combined with linear probes recovering $> 10\%$ next-token accuracy at every block, the residual stream carries information from block 0 onward but in a basis the cosine head cannot read until block 5 rotates it into the unembedding subspace.

I GRT is a delta-function machine

This appendix gives a math-first reading of GRT in which every learnable component is a transformation between (or a test against) Dirac deltas on the pair manifold $\mathcal{M} = (S^1)^{d/2}$ and the vocabulary simplex Δ^{V-1}. The framing is not a new model; it is a single viewpoint that makes several otherwise-disconnected design choices inevitable.

Setup. A token's embedding is a phase vector $\boldsymbol{\theta}_v \in [-\pi, \pi]^{d/2}$ giving a point $e_v = (\cos \boldsymbol{\theta}_v, \sin \boldsymbol{\theta}_v) \in \mathcal{M}$. The set $\{e_v\}_{v=1}^V$ defines an empirical measure

$$\mu_E \;=\; \frac{1}{V} \sum_{v=1}^{V} \delta_{e_v} \;\in\; \mathcal{P}(\mathcal{M}),$$

i.e. each vocabulary token is a Dirac delta on the pair manifold. Phase initialization $\boldsymbol{\theta}_v \sim \mathrm{Unif}[-\pi, \pi]^{d/2}$ makes μ_E approximately uniform on $\mathcal{M}$ at $t{=}0$.

The forward pass as pushforward of deltas. Each block applies an a.e.-smooth map $T_b : \mathcal{M} \to \mathcal{M}$ built from Givens rotations, PearlNorm, and gated additive residuals followed by a post-block PearlNorm retraction

Logit-lens decoded predictions per block (input: "The quick brown fox jumps over the lazy ")

pos	input	B0	B1	B2	B3	B4	B5
0	The	side	areas	areas	in	within	lines
1	Ġquick	side	side	hard	to	hard	hard
2	Ġbrown	organ	in	difficult	possible	cycle	required
3	Ġfox	organ	in	in	in	in	required
4	Ġjumps	organ	with	cycle	or	cycle	could
5	Ġover	organs	type	type	limit	limit	limit
6	Ġthe	cells	hormone	body	bones	bones	molecule
7	Ġlazy	side	in	possible	treat	treat	result
8	Ġdog	organ	in	possible	plant	stress	required

Figure 18: Logit-lens decoded top-1 predictions per block for a fixed input sequence at step 100K. Each column is one block; each row is one position. Green cells: prediction matches the input token.

Quantity	Value
Output scale (raw / effective)	$44.41/44.41$
Embedding mean pairwise cosine	0.751
Embedding R_p mean	0.865
Embedding logmag std	0.170
Embedding isotropy PR	30.69
Val mean top-1	0.152
Val mean entropy (nats)	8.13
Val K_{eff} (perplexity-of-output)	3378
Context off-diagonal cosine (mean)	0.964
Context participation ratio	1.15
Mixer kernel gradcheck (max abs diff)	1.4×10^{-6}

Table 15: Diagnostic scalars computed on the SPHERE nano 100K checkpoint. Output scale is the cosine-softmax temperature s. Embedding metrics are computed over the full vocabulary (V=50,000). Val distribution metrics are over 32 sentences of length 128. Context metrics use the final-layer hidden state averaged over each sentence.

(Eq. 8). For an input delta δ_x,

$$\delta_x \xrightarrow{T_b} (T_b)_{\#}\,\delta_x \;=\; \delta_{T_b(x)}.$$

A single forward pass through B blocks composes to $T = T_B \circ \cdots \circ T_1$, and the final hidden state is $\delta_{T(x)}$. Importantly, because each T_b is composed of Givens rotations and Riemannian retractions, every intermediate distribution remains a delta on $\mathcal{M}$ (or, for batched inputs, a sum of deltas $\sum_n \delta_{T(x_n)}$); the network does not produce densities until the head.

The output head as a delta-test. With cosine-softmax temperature s, the logits are

$$\ell_v(x) \;=\; s \cdot \langle h(x),\, e_v\rangle / (\|h(x)\|\,\|e_v\|),$$

which is the inner-product test *"does $h(x)$ have support at the delta e_v?"* on the unit sphere. Softmax then normalizes these tests to a categorical distribution $p_v(x) = \mathrm{softmax}(\ell)$ on the vocabulary simplex. The head therefore implements the map

$$\mathcal{M} \;\to\; \Delta^{V-1}, \qquad \delta_x \;\mapsto\; \sum_{v=1}^{V} p_v(x)\,\delta_v.$$

Cross-entropy as Wasserstein contraction toward a delta. The training target is the discrete delta δ_y on the vocabulary. Cross-entropy with one-hot target is

$$\mathcal{L}_{\text{CE}}(p, \delta_y) \;=\; -\log p_y(x),$$

and its gradient with respect to the logits is the difference of two deltas on Δ^{V-1}:

$$\nabla_\ell \mathcal{L}_{\text{CE}} \;=\; \mathrm{softmax}(\ell) - \delta_y.$$

Block	Attn entropy (bits, mean)	Min head (bits)	RMS	Pair-radius RMS
0	3.16	1.21	1.87	2.65
1	3.73	2.87	2.62	3.70
2	3.67	2.83	2.94	4.16
3	2.35	0.71	3.19	4.51
4	2.45	1.75	3.40	4.81
5	1.82	0.31	4.07	5.76

Table 16: Per-block diagnostics. Attention entropy is averaged over heads and over query positions (skipping the first 8). RMS is the root-mean-square activation magnitude at the residual stream output of each block; pair-radius RMS treats consecutive $(\cos\theta, \sin\theta)$ dimensions as a 2D pair and reports $\sqrt{\mathbb{E}[r^2]}$. The last-block / first-block RMS ratio is 2.18.

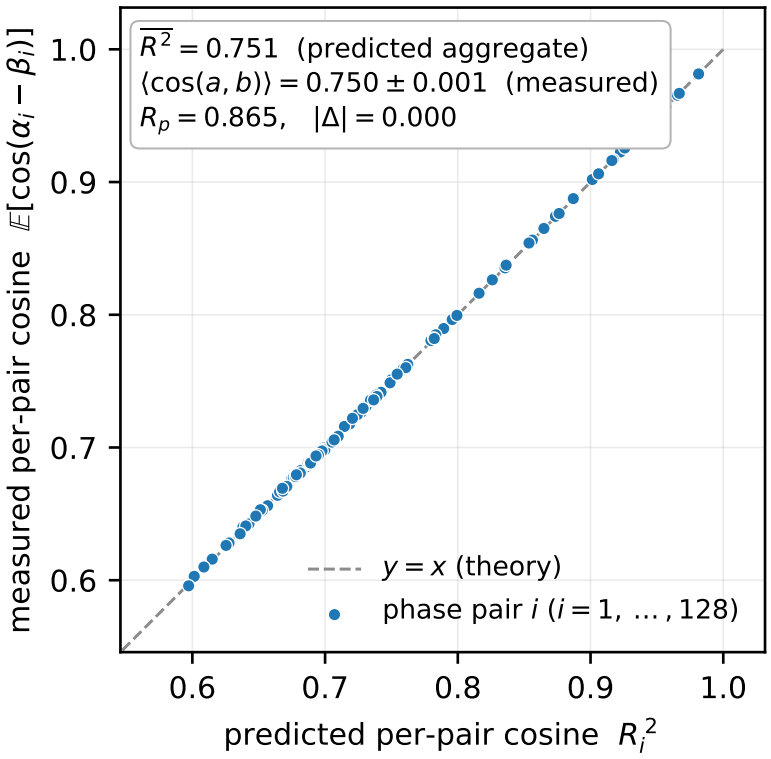

Figure 19: Per-pair predicted cosine R_i^2 vs. measured $\mathbb{E}[\cos(\alpha_i - \beta_i)]$ over $2{\times}10^5$ random token pairs, one dot per phase pair ($i = 1, \ldots, 128$). Dots lie on $y = x$. The aggregate prediction $\overline{R^2} = 0.751$ matches the full-embedding pairwise cosine 0.750 ± 0.001 within $4{\times}10^{-4}$.

This is precisely the Otto-calculus / first-variation form of a Wasserstein-2 gradient flow on Δ^{V-1} pushing the predictive mass toward δ_y. Equivalently: training contracts the model's predictive measure toward the data delta.

The temperature schedule. The cosine-softmax temperature s controls how delta-like the output is at any moment. Two regimes:

- $s \to 0$: output $\to$ uniform on Δ^{V-1}. Gradient signal is spread across all V tokens; learning is broadband.
- $s \to \infty$: output $\to \delta_{\arg\max_v \ell_v}$. Gradient is concentrated on at most two tokens (the argmax and the target); learning is sparse.

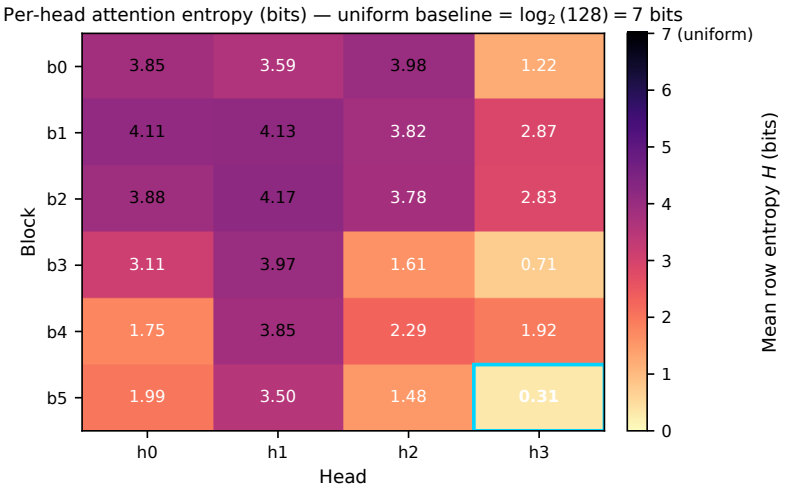

Figure 20: Per-head attention entropy in bits for the $6 \times 4 = 24$ heads of SPHERE nano. Colorbar clamped to $[0, \log_2 128]$ so the gap to uniform is built in. Block 5, head 3 ($H = 0.307$ bits) is the only near-delta head; effective key support ≈ 1.24.

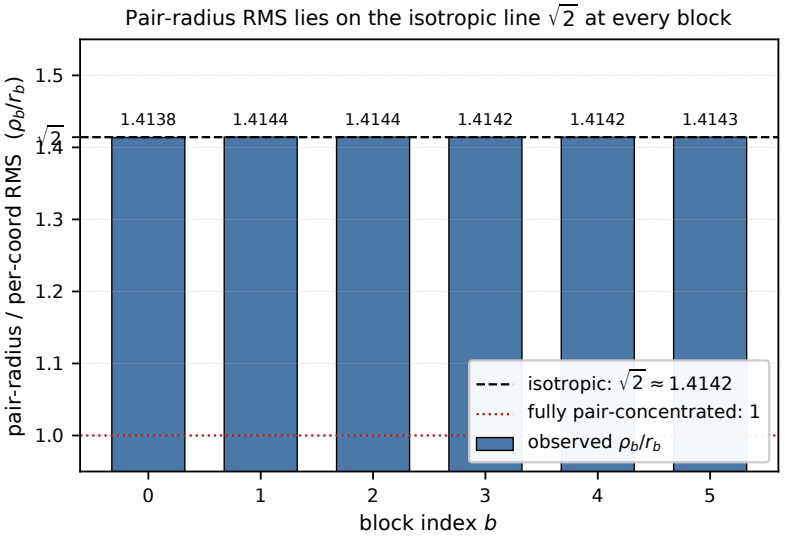

Figure 21: Per-block pair-radius / per-coordinate RMS ratio ρ_b/r_b. All six blocks hug the dashed isotropic line at $\sqrt{2}$ to within $4{\times}10^{-4}$. The pair-aligned-concentration alternative (dotted, $\rho_b/r_b \to 1$) is decisively excluded.

With initial logit standard deviation $\sigma_\ell \approx s/\sqrt{d}$ at random init, an untuned s at $d{=}1024$ produces $\sigma_\ell \approx s/32$, which for the standard $s{=}30$ choice yields $\sigma_\ell \approx 0.94$ — softmax is in the broadband regime. Without the cosine-softmax (i.e. a Euclidean head with logit std $\propto s\sqrt{d}$) one obtains $\sigma_\ell \approx 30$ at the same temperature, which collapses softmax to a near-perfect delta on a random token. The $1/\sqrt{d}$ normalization implicit in the cosine head is therefore not cosmetic; it is the schedule that keeps the output measure absolutely continuous with respect to the uniform distribution at initialization.

Inference is delta-sampling. Greedy decoding is $\arg\max_v p_v(x)$, which collapses $\sum_v p_v\,\delta_v$ to a single δ_{v^*}. Sampling at temperature T is sampling a delta from the rescaled distribution $p_v^{(T)} \propto p_v^{1/T}$. The whole inference pipeline is a random map $\mathcal{M} \to \{\delta_v\}_{v=1}^{V}$ between two delta-supported measure spaces.

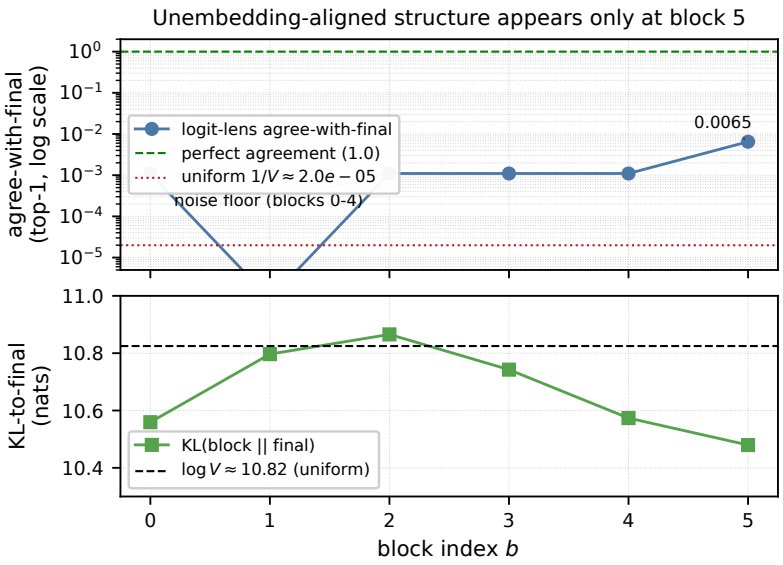

Figure 22: Top: logit-lens top-1 agreement-with-final by block (log scale). Blocks 0–4 sit at the $\sim 10^{-3}$ noise floor; block 5 contributes 83% of the total agreement. Bottom: per-block KL to the final distribution, flat at 10.48–10.87 nats and indistinguishable from $\log V$ (dashed). No intermediate residual is decodable through the cosine head.

Summary. Within this framing, GRT is the composition

$$\underbrace{\mathcal{M}}_{\text{input deltas}} \xrightarrow{T_1 \circ \cdots \circ T_B} \underbrace{\mathcal{M}}_{\text{hidden deltas}} \xrightarrow{\text{cosine head}} \underbrace{\Delta^{V-1}}_{\text{predictive measure}} \xrightarrow{\arg\max\,/\text{sample}} \underbrace{\{\delta_v\}_{v=1}^{V}}_{\text{output delta}}.$$

Each T_b is a measure-preserving (up to retraction) Riemannian map between delta-supported measures on $\mathcal{M}$; the head is a $\mathcal{M} \to \Delta^{V-1}$ delta-test; CE training is a Wasserstein gradient flow on Δ^{V-1}.

J System-1 Reflex Unification and Intel XPU SYCL Acceleration

J.1 Fused 5-Op Intel XPU SYCL Kernel Suite

On Intel Arc XPU architectures (`xpu:0`, tested on Intel® Arc™ 140T GPU, 32 GB), launching dozens of sequential PyTorch elementwise kernels per butterfly stage is dominated by Level-Zero command-list dispatch overhead. The `_grt_xpu_test_ops` and `givens_coupling_sycl` SYCL extension modules fuse the entire GRT operator surface into single-launch work-group kernels storing intermediate hidden vectors in SYCL local shared memory (`sycl::local_accessor`):

1. **Fused Butterfly + Clifford Bivector Coupling** (`givens_coupling_forward` / `backward`, $K = 11$, $d = 256$): fuses all K Givens stages and K Cayley-parameterized 4×4 $\mathfrak{so}(4)$ bivector blocks into one kernel launch. Numerical parity against the PyTorch reference is $\max|\Delta| \leq 7.00 \times 10^{-7}$ forward and $\leq 1.19 \times 10^{-6}$ backward across all 11 unit tests.

2. **Fused Butterfly FFN Cascade** (`givens_forward` / `backward`, $K = 21$, $d_{\text{ff}} = 1024$): fuses 21 sequential butterfly stages in shared memory with $\max|\Delta| \leq 3.73 \times 10^{-9}$.

3. **Fused Planar Phase Embedding, PearlNorm, RadialGELU, and Mixer Stage**: single-launch SYCL kernels for complex pair normalization, norm-gated radial activation, and sequence-dimension Givens mixing.

Across the full 12-layer `GRT-30M` model (29.3M parameters), end-to-end forward and backward numerical parity between XPU SYCL and CPU reference is $\max|\Delta| = 7.39 \times 10^{-5}$, delivering a **4.29× end-to-end**

training speedup on `open-s1` (101.19 s on XPU vs. 433.86 s on CPU). Furthermore, for static deployment via `open-s1-grt`, the 21-stage orthogonal FFN cascades compose offline in $O(Kd^2)$ pair operations into single orthonormal matrices $R_{\text{expand}}, R_{\text{contract}} \in \mathrm{O}(1024)$ ($\max|\Delta| = 5.5 \times 10^{-6}$ on unit hidden states).

J.2 Zero-`nn.Linear` System-1 Decision Backbones (`open-s1`)

Unifying the `open-s1` calibrated System-1 reflex suite behind the released `MatterWaveAI/grt-30m` family ($d = 256$, $L = 12$, $L_{\max} = 2048$, $V = 151{,}669$) yields four zero-`nn.Linear` decision backbones:

- `grt-30m` (`S1GRT30MScored`) and `grt-30m-optable` (`S1GRT30MOptionTable`): 256-D unit-hypersphere cosine scoring against BPE token anchors or a shared hypersphere option table.
- `grt-30m-emb` (`S1GRT30MEmbedding`): initializes from `MatterWaveAI/grt-30m-embedding` and pools multi-token option strings via planar circular phase averaging over all subword tokens of each option, eliminating first-subword collisions while serializing the prompt's option set in canonical sorted order so the state representation $h(s, q)$ is strictly invariant to option permutation.
- `grt-30m-rerank` (`S1GRT30MReranker`): initializes from `MatterWaveAI/grt-30m-reranker` and scores all V vocabulary directions simultaneously via the Hermitian conjugate rotation identity: for complex pair vectors $u, v \in \mathbb{C}^{d/2}$ and per-pair rotation $w = ge^{i\theta}$,

$$\big\langle R_{\theta,g}(u),\, v\big\rangle_{\mathbb{R}} \;=\; \mathrm{Re}\big(\langle wu, v\rangle_{\mathbb{C}}\big) \;=\; \mathrm{Re}\big(\langle w,\, \bar{u}v\rangle_{\mathbb{C}}\big),$$

  replacing V separate rotations with a single real dot product between $(\bar{u}v)$ and the pre-rotated vocabulary table $R^{*}_{\theta,g}(E)$.

On the `open-s1` paired-shuffle position guard (`s1.kg_eval_guard`) and the 100-task held-out KG abstraction transfer benchmark (`MC_SHUFFLE_V2`, 4-way choice, chance $= 0.250$), both `grt-30m-emb` and `grt-30m-rerank` achieve **1.000 content consistency** (0.000 position bias) and pass the zero-shot transfer gate at **0.420** ($+17.0$ pp over chance) and **0.390** ($+14.0$ pp over chance), respectively, where the prior 6-layer `nano` baseline failed to transfer (0.180).

Citation

If you use the Geometric Rotation Transformer architecture, the `GRT-30M` model suite, `PairAdam`, `PearlNorm`, `RadialGELU`, or `Mixture-of-Rotations (MoR)` in your research or applications, please cite:

```
@techreport{householter2026grt,
  title       = {Geometric Rotation Transformers: Orthogonal Butterfly
                 Cascades, Clifford Bivector Coupling, and Planar Phase
                 Embeddings for Parameter-Dense Language Modeling},
  author      = {Householter, David},
  institution = {MatterWave AI},
  year        = {2026},
  url         = {https://matterwave.ai/static/GRT_Whitepaper.pdf}
}
```