Winnex Madhava Direct
Production-Grade Deterministic Vector Search

Winnex AI LAB - Technical Report

June 2026

License: Business Source License 1.1

IP Inquiries: pay@winnex.ai

Abstract

The Winnex Madhava Direct is a deterministic, auditable vector search method that replaces brute-force cosine similarity with progressive dimensionality reduction via QR-orthogonalized Johnson-Lindenstrauss projections on the Stiefel manifold V_k(R^d). Unlike FAISS HNSW or IVF, which rely on non-deterministic graph heuristics, Madhava Direct provides mathematically transparent search with known quality bounds and complete auditability. On the News Category Dataset (10,000 documents, all-MiniLM-L6-v2 384D via QJL to 128D, GPU P100): the Madhava Direct achieves 88% of FAISS FlatIP NDCG@10 at 1.0ms with pool configuration 2000/400/100, and scales to 100,000 documents with 174x speedup over FlatIP (6.9ms vs 1,209ms).

1. Introduction

Modern vector search faces a fundamental tension between speed and interpretability. FAISS HNSW achieves O(log N) search time with 95%+ recall but operates as an opaque graph traversal. The Winnex Madhava Direct addresses this gap by providing deterministic, mathematically transparent search with known, measurable quality degradation relative to exact search.

This document presents the reference implementation and empirical validation of the Madhava Direct, a component of the Winnex AI Enterprise Stack (Layer 3: Search and Retrieval). The Madhava Direct operates independently of the HMC and PiPrimeAnchors components documented in prior Zenodo records (20856138, 20852884), functioning as a standalone production service.

2. Method

2.1 Manifold: Stiefel V_k(R^d)

The Madhava Direct projects corpus vectors onto the Stiefel manifold V_k(R^d), the set of all d x k matrices with orthonormal columns (P^T P = I_k). For k=1, this reduces to the unit sphere S^(d-1). QR decomposition of random matrices ensures P^T P = I_k, making the Pythagorean decomposition valid for Cauchy-Schwarz bound computations.

2.2 Progressive Dimensionality

The search proceeds through four stages:

Stage 1 (4D): Project all N vectors onto 4 dimensions. Compute cosine similarity with query. Keep top 1,000 candidates. Cost: O(N * 4).
Stage 2 (16D): Project remaining vectors onto 16 dimensions. Compute cosine similarity. Keep top 200 candidates. Cost: O(1,000 * 16).
Stage 3 (64D): Project remaining vectors onto 64 dimensions. Compute cosine similarity. Keep top 50 candidates. Cost: O(200 * 64).
Stage 4 (128D exact): Compute exact cosine similarity on 50 candidates. Return top 10. Cost: O(50 * 128).

Total search cost: O(N * 4 + 16,000 + 12,800 + 6,400) = O(N * 4 + 35,200) operations. At N=10,000: 75,200 ops vs 1,280,000 for exhaustive search (17x reduction).

2.3 QR-Orthogonalized Projections

```
def _build_projection(d, dim):
    M = np.random.randn(d, dim)
    Q, _ = np.linalg.qr(M.T)
    P = Q[:, :d].T  # P^T P = I_d, guaranteed
    return P
```

QR decomposition ensures the projection matrix belongs to the Stiefel manifold. This is critical: without orthogonality, the Pythagorean decomposition fails, and the Cauchy-Schwarz bound becomes algebraically invalid. Empirical verification across 50,000 query-vector pairs shows zero bound violations for QR-orthogonalized projections.

3. Benchmark Results

3.1 Main Comparison

Dataset: News Category Dataset (10,000 documents, all-MiniLM-L6-v2, 384D via QJL to 128D). GPU: NVIDIA P100. 20 queries by category.

Method	NDCG@10	T(ms)	vs Exact	NDCG/ms
FAISS FlatIP (exact)	0.4610	0.439	100%	1.050
FAISS HNSW(ef=64)	0.4692	0.299	101.8%	1.569
Madhava Direct (Default)	0.3504	0.732	76.0%	0.479
Madhava Direct (Conservative)	0.4055	1.001	88.0%	0.405
Madhava Direct (Maximum)	0.4242	1.384	92.0%	0.307

3.2 Scaling Performance

As corpus size increases, Madhava Direct maintains near-constant search time while FlatIP and HNSW degrade linearly.

N	FlatIP(ms)	HNSW(ms)	Madhava(ms)	Speedup	Mem(MB)
1,000	0.863	2.256	0.395	2.2x	<1
5,000	13.252	34.816	0.540	24.5x	3
10,000	30.279	43.344	1.042	29.0x	7
50,000	367.932	435.429	3.259	112.9x	20
100,000	1209.688	1258.467	6.921	174.8x	40

3.3 Pool Configuration Trade-off

Four configurations allow balancing quality against speed:

Configuration	Pools	NDCG@10	Time(ms)	Ops
Default	1000->200->50	0.3504 (76%)	0.857	16,800
Conservative	2000->400->100	0.4055 (88%)	1.001	27,200
Maximum	5000->1000->200	0.4242 (92%)	1.384	55,200

4. Complexity Analysis

Search Complexity: O(N * d_1 + K_1 * d_2 + K_2 * d_3 + K_3 * D), where d_i are projection dimensions and K_i are pool sizes. The first stage (4D) operates on all N documents. Subsequent stages operate on constant-sized pools.

Build Complexity: O(N * D * sum(dims)) for initial projections, executed once. Measured build time: 19.1ms for 10,000 documents (vs 200ms+ for HNSW graph construction).

Memory: O(N * D + N * sum(dims)) = O(N * (D + sum(dims))). For N=100,000, D=128, sum(dims)=84: approximately 40MB total (vs 500MB+ for HNSW with M=32 neighbors at 100K nodes).

4.1 Comparison with HNSW

HNSW: O(log N) search, build requires full graph construction (O(N log N) with high constant). Non-deterministic: same query and data may produce different results across runs. Memory: O(N * D + N * M) where M is neighbor count (typically 32-64).

Madhava Direct: O(N * 4 + const) search, O(N * D * sum(dims)) build. Deterministic: same query and data always produce identical results. Memory: O(N * (D + sum(dims))) with no graph overhead.

5. Comparison Summary

Property	FAISS HNSW	FAISS FlatIP	Madhava Direct
Search Complexity	O(log N)	O(N)	O(N*s1 + const)
NDCG vs Exact	101.8%	100%	76-92% (configurable)
Deterministic	No	Yes	Yes
Auditability	None	Exact only	Mathematical bounds
Build Time (10K)	~200ms	<1ms	~19ms
Memory (100K)	~500MB	~50MB	~40MB
Speed at 100K	1,258ms	1,209ms	6.9ms

6. License

Business Source License 1.1

This license permits study, testing, and non-production evaluation. Commercial deployment requires a separate license agreement from the copyright holder.

IP Inquiries: pay@winnex.ai

7. References

1. Padilha, K. A. (2026). Winnex Madhava Direct Benchmark. Zenodo. DOI: 10.5281/zenodo.21011420

2. Padilha, K. A. (2026). Winnex Madhava Cascade. Zenodo. DOI: 10.5281/zenodo.20970487

3. Padilha, K. A. (2026). O(K) Navigation Proof. Zenodo. DOI: 10.5281/zenodo.20856138

4. Padilha, K. A. (2026). Corrected O(K) Benchmark. Zenodo. DOI: 10.5281/zenodo.20852884