WINNEX: Weighted Inference Neural Network Enhance

A Deterministic, Training-Free Framework for Enhancing Pretrained Language Models

DOI: 10.5281/zenodo.21925934

Author: Klenio Araujo Padilha

WINNEX BRASIL SOLUCOES EMPRESARIAIS LTDA - ME | Brazil

CNPJ: 58.364.637/0001-47

Contact: pay@winnex.ai

"A genialidade em engenharia raramente esta na descoberta de uma nova fisica, mas na orquestracao de componentes classicos para resolver um problema sistemico."

Pre-Patent Technical Specification. This document states the mathematics, architecture, verified results, patent claims (with honest viability analysis), business model, prior-art distinction, and explicit limitations of the WINNEX framework.

Abstract

This document is the pre-patent technical specification for WINNEX - Weighted Inference Neural Network Enhancement - a deterministic, training-free framework that enhances a pretrained language model without modifying its weights or running any gradient step. The framework composes three published, pip-installable Winnex packages into a single inference path:

- winnex-nano - the native inference engine: deterministic spectral tokenizer (character -> quaternion spectrum, no BPE, no vocabulary, round-trip CER = 0 across 10 languages), multimodel weight balancer (W' = Sum_i alpha_i R(q_i) W_i), streaming generator, and an OpenAI-compatible SSE server.
- winnex-xfactor - the manifold projector: orthogonal projector P = XX^T computed structurally from the model's embed_tokens by power iteration with deflation (O(D^2r) vs O(D^3)), idempotent (P^2 = P).
- winnex-madhava - the bound engine: deterministic vector search in which every excluded document carries a mathematical proof - a Cauchy-Schwarz bound - that it could not be in the top-K. Zero bound violations by construction.

The three packages share one mathematical core (the Madhava bound-guided kernels), are written with native C++20 cores, and carry reproducible Kaggle benchmarks on real data (BIGANN-100M, Qwen embeddings, News-210K). This document states the mathematics, the architecture, the verified results, the patent claims and their honest viability analysis, the business model behind the BSL 1.1 license, the prior-art distinction, and - explicitly and honestly - the current limitations.

Honest positioning, stated up front. WINNEX is not a substitute for fine-tuning, LoRA, or prompt engineering for generation quality. Its differentiated value is provable, auditable retrieval: a retrieval system in which every excluded document carries a mathematical proof that it could not be in the top-K. A bank or hospital that must answer to a regulator buys this certificate - not benchmark-perfect prose. Sections 6, 8, and 9 develop this position: what the framework does not claim, which patent claims are strong and which are not, and why the business model sells certainty rather than intelligence.

1. Positioning within the Winnex Stack

The WINNEX framework composes layers that are already published as separate Zenodo records and PyPI packages. This document unifies them into a single enhancement method.

Layer	Component	Reference
Inference stack	Winnex AI Inference Stack	10.5281/zenodo.19630736
Vector engine	Winnex Madhava (Cascade)	10.5281/zenodo.20970487
Orchestration	Winnex RAI Architecture	10.5281/zenodo.21292595
Enterprise	Winnex Enterprise Stack	10.5281/zenodo.21107295
Spectral	PsiQRH framework	10.5281/zenodo.17171112
Navigation	O(K) navigation proof	10.5281/zenodo.20856138
X-Factor math	X-Factor mathematics	10.5281/zenodo.21861809
Method (this doc)	WINNEX	this record

WINNEX does not re-implement its pieces: it composes the spectral tokenizer (nano), the manifold projector (xfactor), and the bound engine (madhava), and states the resulting enhancement method and its guarantees. Everything below is measured in the public packages; nothing in this document requires unpublished code.

2. The Problem

2.1 Enhancing a model you cannot retrain

In regulated industries - legal, medical, financial, patents, compliance - an organization often holds a pretrained model (Qwen, Llama, BERT, DeepSeek, GPT) whose training pipeline it does not own and cannot afford to run. Standard enhancement (fine-tuning, LoRA, RLHF) requires training data, compute, and access to the training path - all of which are unavailable or prohibited by the deployment environment.

2.2 A retrieval problem that approximate indexes cannot answer

The same organizations deploy retrieval-augmented generation (RAG) over sensitive corpora. Every approximate index in wide use - HNSW, IVF, IVF-PQ, product quantization - answers 'here is a fast guess at the top-K'. None can answer the question a regulator actually asks:

"Prove that your search did not miss a relevant document."

For a bank proving to a regulator (Bacen, SEC, EU AI Act) that no critical contract was omitted, or a hospital proving to a health authority (ANVISA, HIPAA, LGPD) that no medical record relevant to a query was excluded, 'we think it is fine' is not a defensible answer. HNSW/IVF recover 92-98% of the exact top-K in the best case and as low as 48% with aggressive quantization - with no way to know which documents were silently dropped.

2.3 The claim of this work

WINNEX states a different route: a model can be enhanced at inference time using only the geometry it already learned, and its retrieval can carry a per-document mathematical proof of completeness. The enhancement is:

- Deterministic - the same input always produces the same output.
- Training-free - no gradient step, no training data.
- Structural - read directly from the model's embed_tokens matrix.
- Provable - the retrieval guarantee is a mathematical bound, not a claim.

3. The Mathematics

3.1 The manifold hypothesis

A pretrained language model has an embedding matrix E in R^(VxD) mapping V tokens to D-dimensional hidden vectors. The V rows of E do not fill R^D; they concentrate near a lower-dimensional structure. Formally, the token embeddings lie on a Riemannian manifold M embedded in R^D, and WINNEX captures the linearized tangent structure of M - the principal subspace where the embeddings concentrate.

3.2 Centering and covariance

```
mu = (1/V) Sum_i E_i                (mean embedding)
Ebar = E - mu 1^T                   (centered matrix, V x D)
C = (1/V) Ebar^T Ebar               (D x D empirical covariance)
```

C is symmetric positive semi-definite.

3.3 Spectral decomposition

```
C = U Lambda U^T
```

with U = [u_1 ... u_D], U^T U = I_D, and Lambda = diag(lambda_1 ... lambda_D), lambda_1 >= ... >= lambda_D >= 0. The eigenvectors are the principal directions of the token manifold; lambda_k is the variance captured by direction k.

3.4 Computational method - bound-guided power iteration

The full eigendecomposition is O(D^3). Following the Madhava principle - bound-guided pruning without exhaustive computation - the top-r eigenpairs are computed by power iteration with deflation:

```
for k = 1 ... r:
    initialize v_k (deterministic, unit norm)
    repeat:
        w = C v_k
        lambda_k = v_k^T C v_k      (Rayleigh quotient)
        v_k = w / ||w||
    until |Delta lambda_k| < eps
    deflate: C <- C - lambda_k v_k v_k^T
```

Cost: O(D^2 r iters) versus O(D^3) - O(D/r) times faster at fixed iterations, and measured ~2000x faster than a full Jacobi/eigh at D = 2048. The Rayleigh quotient plays the same role as the Cauchy-Schwarz bound in the Madhava engine: a provable selection criterion without scanning everything.

3.5 Effective rank

```
r(tau) = min{ r : Sum_{k=1..r} lambda_k >= tau * tr(C) }
```

WINNEX uses tau = 0.95 by default (95% of the embedding variance). On real Qwen embeddings (D=128) this yields effective rank 118 with variance captured 0.954 - the manifold is genuinely low-dimensional.

3.6 The X-Factor operator

```
X = [u_1 ... u_r] in R^(D x r)      (the X-Factor basis)
P = X X^T in R^(D x D)             (orthogonal projector)
```

Properties (verified to numerical precision):

```
P^2 = P     (idempotent)
P^T = P     (symmetric)
```

For any h in R^D, ||h - Ph|| is the distance to the manifold subspace, and Ph is the best r-rank approximation of h in the least-squares sense (Eckart-Young).

3.7 Text embedding via the composite operator

A text T = c_1...c_n is mapped to a deterministic feature vector. The primary representation is the character histogram:

```
h_j(T) = (1/||.||) Sum_i 1[ c_i == j (mod B) ]    j = 0 ... B-1
```

normalized to unit L2 norm. The histogram is chosen over a raw spectral average because the latter collapses (pairwise cosine similarity between different texts tends to ~0.99); the histogram separates them with cosine spread ~0.13-0.60. The histogram is expanded into R^D by zero-padding and L2 normalization, then projected:

```
psi~ = [ h_0 ... h_{B-1}, 0 ... 0 ] / ||[h_0 ... h_{B-1}, 0 ... 0]||
h = P psi~ = X X^T psi~ in R^D
```

The composite operator:

```
Phi_X(T) = X X^T pad_L2( histogram(T) )
```

Training-free: X depends only on E. Deterministic: the same text always yields the same h. Manifold-respecting: Phi_X(T) is in span(X).

Structural limitation - the histogram is order-blind. A histogram is a bag-of-characters: it destroys syntax,

word order, and positional semantics. The sentences 'the doctor cured the patient' and 'the patient cured the doctor' produce exactly the same h(T). The composite operator Phi_X is therefore appropriate for fixed representations - retrieval, deduplication, classification - where permutation-invariance is tolerable or even desirable. It is not a mechanism for sequence generation: injecting an order-free vector into a forward pass whose attention is calibrated on ordered token streams cannot recover the lost order. The open frontier of Section 6 is therefore not merely 'unmeasured quality'; it is a structural limitation of the representation for the generation task. This is why the retrieval and audit path is the load-bearing claim of the framework, and why the generation path is stated as a mechanism, never as a product.

3.8 Weighted model fusion

Multiple checkpoints blend into one weight matrix with explicit mixing coefficients:

```
W' = Sum_i alpha_i R(q_i) W_i     with Sum alpha_i = 1
```

Each W_i is rotated by a unit quaternion R(q_i) before the weighted sum - a controlled phase alignment that respects each model's learned manifold. The nano WeightBalancer implements this operator and validates it: blend error 0.000000 on synthetic weights, fused model preserves behavior (cos > 0.99).

3.9 The retrieval bound - why exclusions are provable

For any query q and candidate vector v, with P a QR-orthogonalized projection:

```
<v,q>  =  <Pv,Pq> + <v-P^T Pv, q-P^T Pq>
```

and by Cauchy-Schwarz:

```
|<v,q> - <Pv,Pq>|  <=  ||v - P^T Pv|| * ||q - P^T Pq||  =: e(v) e(q)
```

This yields a two-sided interval for the true inner product:

```
<Pv,Pq> - e(v)e(q)  <=  <v,q>  <=  <Pv,Pq> + e(v)e(q)
```

Retention direction (inclusion): any candidate whose upper bound is at or above the K-th best may be in the top-K and must be retained. Because the engine never discards a candidate without proof that it cannot be in the top-K, bound_violations == 0 by construction. The residual ||v - P^T Pv|| is computed on the real float32 projection, not the int8-quantized one - this is what the inequality requires, and it is what makes the bound exact rather than approximate.

4. The Architecture

The framework is realized by three published packages connected into a single inference path. Each package is independently pip-installable and has its own reproducible benchmark.

4.1 winnex-nano - the native inference engine

Module	Purpose	Verified
SpectralTokenizer	Character -> quaternion spectrum	CER = 0, 10 languages
WeightBalancer	Fusion W' = Sum alpha_i R(q_i) W_i	Blend error 0.000000
StreamEngine	Chunk gen via Madhava build	3-chunk stream 1.7 ms
server_sse	OpenAI-compatible /v1/chat/completions	Live SSE server

The tokenizer is a quaternion spectrum: each character is mapped deterministically to a quaternion state, and the text is reconstructed exactly from the states (round-trip). There is no vocabulary and no BPE merge step. Its honest trade-off is stated in Section 6: it is 50-70x slower and ~1000x less compact than BPE, and is an autonomous encoding for the native engine, not a drop-in replacement for a BPE-trained model.

4.2 winnex-xfactor - the manifold projector

The X-Factor is a training-free operator that embeds text into a pretrained model's latent space, computed structurally from embed_tokens - never by running the model:

```
P, rank, variance = winnex_xfactor.compute_xfactor(E, vocab, dim, tau)
h = winnex_xfactor.embed_spectral(psi, P)   # length D, on the manifold
```

The API exposes compute_xfactor, the XFactor object (.project, .project_batch, .projector, .basis, .rank, .variance_captured, .dim), expand_spectral (zero-pad + L2 normalize) and embed_spectral (full text -> manifold projection). The eigendecomposition is power iteration with deflation - the same bound-guided philosophy as the Madhava engine.

4.3 winnex-madhava - the bound engine

The bound engine answers the question no approximate index can answer. It is one engine with four execution modes:

Mode	Input contract	Query cost	Guarantee
default (C++)	uint8 (BIGANN-style)	O(N.d) full bound scan	0 vio, per-doc proof
hybrid (MadHybrid)	float32 or uint8	sublinear (nprobe x cell)	0 vio within cells
speed (GPU OpenCL)	float32/uint8	fused QK^T+topk	exact, 2.41 ms @ 1M
streaming	mmap, never in RAM	O(N) over int8	100M in 31.4 GB

The internal pipeline is a two-stage cascade with post-filter: Stage 1 applies a wide bound B1 (cheap QR projection) and prunes every vector to k1 - any vector pruned here is mathematically proven not to be in the exact top-K. Stage 2 applies a tighter bound B2 and prunes the k1 survivors to k2 (k2_max = 2000 caps the survivors at scale, making the exact stage tractable at 100M). The post-filter recomputes the exact metric on the surviving top-k2, so the result is the true top-K of the surviving set. Modulation ranks survivors by B1 + alpha(B2 - B1) with alpha = sigmoid((e1-e2)/mean(e1)) - an error-backpropagation refinement that never sacrifices the 0-violation guarantee. Pruning always uses the tightest available bound; modulation is used only for ranking (the stack's FIX(1) invariant). Memory is controlled by quant='int8' (~4x less than float32; 18.6 GB for 100M x 128D).

4.4 The composition - one inference path

```
Text
 |
 +- 1. winnex-nano - SpectralTokenizer
 |        text -> quaternion spectrum  (no BPE, no vocab)
 |        round-trip CER = 0 - 10 languages
 |
 +- 2. winnex-xfactor - manifold projector
 |        spectrum -> expand -> P psi~  (onto the manifold)
 |        P^2 = P - P^T = P - training-free - O(D^2 r)
 |
 +- 3. ForwardEngine - dense forward
 |        h -> RMSNorm/QKV/RoPE/GQA/SiLU -> logits -> tokens
 |        native C++20 - no CUDA - no GPTQ
 |
 +- 4. winnex-madhava - bound engine  (parallel path)
```

```
spectral embedding -> cosine bound -> top-K
0 violations by construction - per-document proof
```

4.5 The shared mathematical core

The Madhava engine's kernels (QK^T matmul, selective top-K with Cauchy-Schwarz bounds) are reused by the X-Factor and the streaming engine - one mathematical core, no duplicated code. The same bound-guided philosophy drives search, projection, and selection across the network: the Rayleigh quotient bounds the eigenvalue, Cauchy-Schwarz bounds the inner product, and both are selection criteria that never require scanning everything.

5. Verified Results

The following results are measured on real data and reproducible in the public notebooks (each installs the package from PyPI and runs on real corpora).

5.1 End-to-end connections

Connection	Validated result	Guarantee
Spectral round-trip	CER = 0 in 10 languages	Invertible, no vocab
X-Factor projection	P^2 = P (diff 0.000000)	Stable projector
X-Factor effective rank	rank 118, var 0.954	Low-dimensional
Forward dense	Qwen2.5-1.5B -> logits	Native path works
Autoregressive	3-5 tokens, KV cache	Generation works
Spectral -> Madhava RAG	7/8 top-1, 0 violations	Native retrieval
Weight fusion	cos > 0.99, top-1 5/5	Preserves behavior

5.2 The bound engine against the exact-scan ceiling

From the real benchmark (winnex-madhava-1-7-real-benchmark-vs-hnsw-ivf-pq, BIGANN-100M subset 1M, 100 queries, recall vs the exact-scan ceiling on the same subset - the valid reference):

Method	R@10	Lat (ms)	Build (s)	Vio
Exact-scan ceiling (local)	1.0000	-	3.0	-
Madhava bound (int8 5%/1%)	0.9960	45.8	2.0	0
Madhava speed GPU (OpenCL)	1.0000	6.14	1.5	-
Madhava speed GPU batch	1.0000	3.09	-	-
HNSW(ef=128)	0.9760	0.56	159	-
HNSW(ef=64)	0.9330	0.34	159	-
IVF(np=50)	0.9250	0.75	61	-
IVF-PQ(np=10)	0.4780	0.23	10	-

Read honestly: the bound engine recovers 99.6% of the exact top-10 with 0 bound violations and a ~77x faster build than HNSW (2.0 s vs 159 s). The GPU speed mode is exact (R@10 = 1.0) at 6.14 ms single-query and 3.09 ms/query batch - the fused QK^T+topk kernel drops single-query latency from 47.8 ms to 2.41 ms (20x) at 1M by splitting the scan into M work-groups so all GPU compute units stay active. Approximate baselines lose recall: HNSW 97.6%, IVF 92.5%, IVF-PQ 47.8% - faster, but not provably complete.

5.3 Build speed and streaming scale

Task	winnex-madhava	HNSW

Index 1M (build)	2.0 s	159 s (78x slower)
Index 10M (build)	~2.4 s	~30 min+
Index 100M (streaming)	~342 s	~6+ hours

Scale	Build (s)	Lat (ms)	R@10	NDCG	RSS (GB)	Vio
100K	1.3	7.5	0.750	0.658	0.4	0
1M	5.9	66	0.843	0.667	0.8	0
10M	34.0	698	0.501	0.544	3.8	0
100M	342.6	7592	0.780	0.813	31.4	0

100M vectors (12.8 GB) indexed in 342.6 s (~5.7 min, 4 CPUs) via mmap - the raw corpus is never loaded into RAM, and 0 bound violations at every scale. This is what makes continuous ingestion / dynamic RAG practical: rebuild the whole index on every ingestion, because the build is seconds, not hours.

5.4 The X-Factor and the spectral tokenizer

- X-Factor on real Qwen embeddings (Kaggle winnex-xfactor-benchmark-manifold): effective rank 118, variance captured 0.954, compute 0.042 s at D=128 (power iteration with deflation, O(D^2 r)).

- Spectral tokenizer (Kaggle winnex-nano-benchmark-spectral): round-trip perfect 8/8, char error rate 0.00%, encode throughput 234K chars/s, 3-chunk stream 1.7 ms, no vocabulary (vs 151K BPE tokens).

6. Honesty Statement (limitations)

Consistent with the practice of the other Winnex Zenodo records, this section states what the framework does not claim.

- 1. No speed advantage at high dimension for the bound scan. At d = 1536 the residual of the bound grows with sqrt(d), the bound no longer discriminates, and the bounded scan becomes slower than a direct exact scan (measured 13x at d=1536). The value in this regime is the certificate (0 bound violations, deterministic, auditable), not latency.

- 2. The spectral tokenizer is not a BPE replacement. It is 50-70x slower and ~1000x less compact than BPE. It is an autonomous encoding for the native engine, not a drop-in replacement for a BPE-trained model.

- 3. The X-Factor projects onto the principal subspace of the input embeddings, not the deep internal geometry of the transformer. It is a linear approximation of the input-embedding manifold.

- 4. Generation quality is the open frontier - and structurally constrained. The mechanism works (forward produces logits, generate produces tokens), and quality with the spectral path is not yet demonstrated with standard metrics (perplexity, MMLU, human eval). Beyond 'not yet measured', the histogram representation is order-blind by construction (Section 3.7), so the generation path cannot be assumed to recover word order from an order-free vector. The retrieval path is the load-bearing claim; the generation path is a mechanism under investigation.

- 5. The stage-2 cascade can corrupt ranking at high dimension. Configurations that guarantee recall must retain all candidates.

- 6. Quantization with a +1 shift corrupts the metric at high dimension. Queries and corpora must be quantized identically, or float32 used.

- 7. Hybrid and O(K) modes are approximate by design. Hybrid trades recall for speed like any IVF index; the O(K) anchor navigation reaches 100% of the exact-scan ceiling at nprobe=8 but drops to ~67% at nprobe=4. Only the full bound scan and the GPU exact scan carry the 0-violation guarantee at full recall.

- 8. The default engine's input contract is uint8. Passing raw float32 embeddings to default mode truncates them and recall collapses; float32 embeddings must use hybrid or speed mode.

- 9. Weighted quaternion fusion (Claim 3) is unvalidated against SOTA merging. The operator is mathematically well-defined, but no empirical comparison against SLERP, TIES, or DARE on real checkpoints is published in this document or the public benchmarks. Its viability rests on that missing evidence.

7. Engineering Philosophy: Orchestration over Discovery

"A genialidade em engenharia raramente esta na descoberta de uma nova fisica, mas na orquestracao de componentes classicos para resolver um problema sistemico."

WINNEX contains no new physics. Its components are deliberately classical: PCA on a covariance matrix (1930s), power iteration with deflation (1950s), the Cauchy-Schwarz inequality (1888), character histograms (a counting primitive), and quaternions (1843).

What is new is the systemic problem these components are orchestrated to solve: proving, to a regulator, that a retrieval system did not omit a critical document. The orchestration is the invention - and this is exactly the lens through which the patent claims in Section 8 must be read. It is also why the claims that re-assert a single classical component are weak (obvious), while the claim that composes them into an auditable method of system is the strongest.

8. Patent Claims (draft) and Viability Analysis

The following claims represent the architecture described in this document as prior art. No patent application has been filed as of the date of this publication.

- Claim 1 - Weighted Inference Neural Network Enhancement (WINNEX). A method for enhancing a pretrained neural network at inference time without modifying its weights, comprising: computing an orthogonal projector P = XX^T from the principal subspace of the model's token-embedding matrix; projecting an arbitrary text representation onto the manifold via Phi_X(T) = XX^T pad_L2(histogram(T)); and feeding the projected hidden state to the model's forward pass.
- Claim 2 - Bound-guided power iteration with deflation. A method for computing the top-r eigenpairs of an embedding covariance by power iteration with deflation, using the Rayleigh quotient as a provable selection bound, achieving O(D^2 r) instead of O(D^3).
- Claim 3 - Weighted quaternion model fusion. A method for fusing multiple model weight matrices by weighted quaternion rotation, W' = Sum_i alpha_i R(q_i) W_i with Sum alpha_i = 1, respecting each model's learned manifold.
- Claim 4 - Deterministic spectral RAG with proof. A method for retrieval-augmented generation in which the query and corpus are represented by a deterministic spectral histogram, retrieved by a Cauchy-Schwarz bound engine with 0 bound violations by construction, and the per-document exclusion certificate is recorded in an audit trail.
- Claim 5 - Composition of spectral tokenizer, manifold projector, and bound engine as a single inference enhancement path. A method in which text moves through spectral tokenization, manifold projection, native forward, and proof-based retrieval as one deterministic pipeline whose per-document exclusion certificates are emitted as an auditable output.
- Claim 6 - Variance-weighted effective rank selection. A method for selecting the effective rank r(tau) that captures fraction tau of the embedding variance, used to size the manifold projector.

8.1 Honest viability assessment

A patent office (USPTO, INPI, EPO) can reject a claim on two independent grounds: Sec.101 subject-matter ('abstract idea' - pure mathematics is not patentable) and Sec.103 obviousness (activity

inventiva - the combination is obvious to a person skilled in the art). Assessed honestly, claim by claim:

Claim	Sec.103 obviousness	Sec.101 abstract idea	Viability
1 (PCA+histogram+projector)	High	Medium	Low-Medium
2 (power iteration)	High	High	Low
3 (quaternion fusion)	Medium	Medium	Medium
4 (spectral RAG + certificate)	Medium	Medium	Medium
5 (composition as system)	Low	Low-Medium	Highest
6 (variance threshold rank)	High	High	Low

The strategic reading. Claims 2 and 6 are most likely rejected for obviousness: power iteration and variance thresholds are classical mathematics that a practitioner would apply without invention. Claim 1 now faces the strongest obviousness reference from the retrieval literature - the LSA / SVD-of-histograms line from the 1990s - and should be treated as defensible-but-weak, not medium. Claims 3 and 4 are defensible but narrow.

The patentable core is the system composition - Claims 4 + 5. The distinctive, defensible asset is not any single component but the pipeline: deterministic spectral representation, model-derived projector, bound engine with per-document Cauchy-Schwarz certificate, and auditable trail. Because no approximate index (HNSW, IVF, PQ) can produce such a certificate - they trade recall for speed by design - and because the problem it solves (regulatory proof of retrieval completeness) is systemic and unaddressed, the composition has the best chance of being granted as a 'method and system for auditable, deterministic information retrieval'. This is the claim a bank or hospital buys: a provable recall certificate, not benchmark-prose. It aligns exactly with the engineering philosophy of Section 7: the invention is the orchestration, and the business value (Section 9) is certainty, not intelligence.

9. Business Model under BSL 1.1

9.1 Positioning - not a competitor in general intelligence

WINNEX is not attempting to compete with OpenAI, Anthropic, or Meta on 'general intelligence' or generation quality. It does not sell a frontier model, and it does not sell prompt engineering or a LoRA replacement. It sells what the license and the architecture both state: a mathematical clearing house for RAG - a provable, auditable retrieval layer whose output is a mathematical receipt (per-document exclusion certificate) that a regulated institution can present to a regulator. The buyer is not the team chasing benchmarks; the buyer is the CTO who must sign a civil-liability statement over the model's answers and needs proof that the system retrieved every relevant document without approximate heuristics.

9.2 The customer and the product

Banks and hospitals deploy open-source LLMs (Qwen, Llama) on their own infrastructure. They are not allowed to send sensitive data to a US frontier API. Their problem is not 'generate better prose' - it is:

> *"Prove to a regulator (Bacen, ANPD, SEC, EU AI Act, HIPAA, LGPD) that the RAG system did not omit a document critical to the decision."*

The product is 'Mathematical Certification & Audit': a per-document exclusion certificate (every document left out of the top-K carries a proof it could not be in the top-K, defensible in court and in regulatory inspection); determinism (same input -> same output -> same audit trail, reproducible across runs); 0 bound violations by construction (a property of the algorithm, not a one-off benchmark); and instant rebuild for continuous ingestion (seconds, not HNSW's hours, so the certificate always reflects the current

corpus).

9.3 How the license operationalizes this

Phase	BSL 1.1 status	Sales motion
Evaluation	Free (non-production)	Bank installs, benchmarks, verifies the certificate
Pilot	Free (non-production)	Audit team signs off on the per-document proofs
Production	Commercial license	License cost vs cost of unrecoverable retrieval failure
ISV / platform	Commercial license	DB vendors and RAG platforms embed the engine

9.4 Why this is coherent with the architecture

Every engineering choice supports the business model: the deterministic tokenizer (auditable representation), the projector (manifold-respecting query), the bound engine (the certificate), the streaming build (current corpus, always), and the honest limitations (Section 6) - a buyer who must satisfy a regulator values the explicit statement of where the guarantee holds and where it does not. The document itself - with its draft claims and viability analysis - is part of the product: it is the technical specification a compliance department needs to make a defensible procurement decision.

10. Prior Art Distinction

Method	How it differs from WINNEX
Fine-tuning / LoRA / PEFT	Requires training; WINNEX is training-free
Prompt engineering / ICL	Operates on token string; WINNEX in manifold
RAG	Approximate indexes; WINNEX has per-doc proof
HNSW / IVF / IVF-PQ	Approximate; no completeness guarantee; no certificates
Model merging (TIES/DARE/SLERP)	WINNEX fusion is manifold-respecting
Quaternion NNs (QRNN/QCNN)	Use quaternions inside the model; WINNEX as fusion operator
Fixed random projections (JL)	Not orthogonal; WINNEX uses PCA geometry
Standard PCA / eigh	Exhaustive O(D^3); WINNEX is O(D^2 r)
LSA / SVD of TSVD-document (1990s)	Projected histograms via SVD-document (1990s). WINNEX applies PCA to a model-derived subspace and composes it with a proof-based bound engine. Strongest obvio
Vector DBs with audit logs	Log what was returned; WINNEX proves what could not be

The distinguishing claim is the composition: no prior work combines a deterministic spectral tokenizer, a model-derived orthogonal projector, a native dense forward, and a proof-based retrieval engine into one training-free enhancement path for regulated use.

11. License

Business Source License 1.1 (BSL 1.1). Free for evaluation and non-production use. Commercial license required for production. Change date converts to GPL v2.0 or later. Contact: pay@winnex.ai.

12. References

[1] Padilha, K. A. (2026). Winnex AI Inference Stack. Zenodo 10.5281/zenodo.19630736.

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

[3] Padilha, K. A. (2026). Winnex RAI Architecture. Zenodo 10.5281/zenodo.21292595.

[4] Padilha, K. A. (2026). Winnex Enterprise Stack. Zenodo 10.5281/zenodo.21107295.

[5] Padilha, K. A. (2026). X-Factor Mathematics. Zenodo 10.5281/zenodo.21861809.

[6] Padilha, K. A. (2026). PsiQRH framework. Zenodo 10.5281/zenodo.17171112.

[7] Padilha, K. A. (2026). O(K) navigation proof. Zenodo 10.5281/zenodo.20856138.

[8] winnex-nano, winnex-xfactor, winnex-madhava. PyPI. pypi.org/project/winnex-nano; winnex-xfactor; winnex-madhava.

[9] Reproducible benchmarks (Kaggle): winnex-madhava-1-7-real-benchmark-vs-hnsw-ivf-pq; winnex-madhava-hybrid-vs-hnsw-ivf-pq; winnex-madhava-stream-100m; winnex-nano-benchmark-spectral; winnex-xfactor-benchmark-manifold.