Project JANUS

Complete Technical Documentation

A Brain-Inspired Architecture for Autonomous Financial Systems

The Complete JANUS Suite

Volume I: Architecture Overview
The philosophical foundation and system design

Volume II: Forward Service
Real-time trading, pattern recognition, and execution

Volume III: Backward Service
Memory consolidation, schema formation, and learning

Volume IV: Neuromorphic Architecture
Brain-inspired components and cognitive mapping

Volume V: Rust Implementation
Production-ready ML system with Rust and Python

Classification: Master Technical Document

Version: 2.0 (Complete Edition)

Author: Jordan Smith

github.com/nuniesmith

Date: December 25, 2025

"The god of beginnings and transitions, looking simultaneously to the future and the past."

Contents

Preface: The Complete JANUS Documentation

This document represents the complete technical specification for **Project JANUS**, a neuromorphic trading intelligence system that bridges neuroscience, machine learning, and quantitative finance.

How to Read This Document

This master document combines five interconnected volumes that can be read independently or as a unified whole:

1. **Volume I - Architecture Overview (main.tex)**: Start here for the big picture. This volume explains the philosophical motivation, the dual-service architecture, and how all components fit together.

2. **Volume II - Forward Service (forward.tex)**: Deep dive into the real-time trading engine. Read this for implementation details of visual pattern recognition (DiffGAF + ViViT), logical reasoning (Logic Tensor Networks), and the decision-making system.

3. **Volume III - Backward Service (backward.tex)**: Explore the learning and memory system. This volume covers the three-timescale memory hierarchy, Sharp-Wave Ripple simulation, schema consolidation, and UMAP visualization.

4. **Volume IV - Neuromorphic Architecture (neuro.tex)**: Understand the brain-inspired design philosophy. Maps each trading component to its neuroscience equivalent, from cortex to cerebellum.

5. **Volume V - Rust Implementation (rust.tex)**: Production deployment guide. Details the Rust-first ML stack, FastAPI gateway, Docker/Kubernetes deployment, and migration roadmap.

Document Structure

Each volume maintains its independence while contributing to the unified vision. Cross-references are provided where concepts span multiple volumes. Implementation checklists at the end of each technical volume provide actionable development guidance.

Intended Audience

- **Quantitative Researchers**: Volumes I, II, and III
- **ML Engineers**: Volumes II, III, and V

- **System Architects**: Volumes I, IV, and V
- **Neuroscience-Curious**: Volumes I and IV
- **DevOps/Infrastructure**: Volume V

Welcome to the future of algorithmic trading.

Part I

Architecture Overview

[12pt, a4paper]article

[utf8]inputenc [T1]fontenc pmboxdraw newunicodechar [english]babel helvet setspace [top=2.5cm, bottom=2.5cm, left=2.5cm, right=2.5cm]geometry amsmath, amssymb, amsfonts graphicx xcolor fancyhdr titlesec enumitem listings tcolorbox tabularx array

xurl hyperref

▼▼ $\rightarrow\rightarrow$ $\leftarrow\leftarrow$ $\leftrightarrow\leftrightarrow$ $\Rightarrow\Rightarrow$ ..... $\geq\geq$ $\leq\leq$ $\neq\neq$ $\approx$ $\in\in$ $\notin$ $\wedge\wedge$ $\vee\vee$ $\neg\neg$ $\times\times$ $\div\div$ ∞ $\sum\sum$ $\prod\prod$ $\int\int$ $\surd$ $\partial\partial$ $\nabla\nabla$ $\alpha\alpha$ $\beta\beta$ $\gamma\gamma$ $\delta\delta$ $\epsilon\epsilon$ $\theta\theta$ $\lambda\lambda$ $\mu\mu$ $\pi\pi$ $\sigma\sigma$ $\tau\tau$ $\phi\phi$ $\omega\omega$ $\Delta\Delta$ $\Sigma\Sigma$ Π $\Omega\Omega$

Project JANUS

Neuromorphic Trading Intelligence

A Brain-Inspired Architecture for Autonomous Financial Systems

The Two-Faced Architecture

Forward (Janus Bifrons): The Conscious Trader
Visual perception, logical reasoning, and real-time execution

Backward (Janus Consivius): The Sleeping Mind
Memory consolidation, schema formation, and learning

Classification: Main Architecture Document

Version: 2.0

Author: Jordan Smith
github.com/nuniesmith

Date: December 25, 2025

"The god of beginnings and transitions, looking simultaneously to the future and the past."

Abstract

Financial markets have evolved into complex adaptive systems operating at timescales far beyond human perception. Modern high-frequency trading requires decision-making in microseconds, processing millions of data points across multiple modalities—price movements, order flow toxicity, news sentiment, and macroeconomic signals—while adhering to strict regulatory and risk management constraints. The challenge is not merely computational speed, but the integration of *perception*, *reasoning*, and *memory* into a unified autonomous system.

This document presents **Project JANUS**, a neuromorphic trading intelligence system inspired by the bifurcated nature of its namesake—the Roman god who simultaneously looks forward and backward. JANUS represents a paradigm shift from monolithic deep learning models to a brain-inspired architecture that mirrors the functional specialization and information flow patterns observed in biological neural systems.

The architecture is fundamentally dual:

- **JANUS Forward (Janus Bifrons)**: The "wake state" trading engine that operates in real-time market conditions. It implements visual pattern recognition through Gramian Angular Fields (GAF) and Video Vision Transformers (ViViT), symbolic reasoning through Logic Tensor Networks (LTN), multimodal fusion via gated cross-attention, and neuromorphic decision-making inspired by basal ganglia dual pathways.
- **JANUS Backward (Janus Consivius)**: The "sleep state" memory consolidation system that processes trading experiences during off-market hours. It implements a three-timescale memory hierarchy (hippocampal episodic buffer, sharp-wave ripple replay, and neocortical schema formation), UMAP-based cognitive visualization, and integration with vector databases for long-term knowledge storage.

By separating the hot-path real-time inference (Forward) from the cold-path batch learning (Backward), JANUS achieves both microsecond-latency execution and deep, contemplative learning—mirroring the wake-sleep cycle of biological brains. The system is implemented in Rust for performance-critical components and Python for training pipelines, with explicit neuromorphic mapping to brain regions: visual cortex, prefrontal cortex, hippocampus, basal ganglia, thalamus, amygdala, hypothalamus, cerebellum, and integration centers.

This main document provides the conceptual framework, architectural philosophy, and system integration overview. Detailed technical specifications are provided in companion documents:

- `janus_forward.tex` — Forward service algorithms and implementation

- `janus_backward.tex` — Backward service memory architecture
- `janus_neuromorphic_architecture.tex` — Brain region mapping and neuromorphic design
- `janus_rust_implementation.tex` — Rust implementation strategy and deployment

Key Contributions:

1. A neuromorphic architecture that maps trading functions to specialized brain regions
2. Differentiable Gramian Angular Fields (DiffGAF) for learnable time series imaging
3. Logic Tensor Networks for hard constraint enforcement in financial decision-making
4. Sharp-wave ripple simulation for experience replay with 10-20× time compression
5. Dual-service architecture separating real-time inference from batch consolidation
6. Rust-first implementation for safety-critical financial systems

Contents

1 Introduction: The Crisis of Complexity

1.1 The Evolution of Quantitative Trading

The history of quantitative trading can be characterized by escalating complexity and decreasing human interpretability:

- **Quant 1.0 (1970s-1990s):** Rule-based expert systems and technical indicators (moving averages, RSI, Bollinger Bands). Human-interpretable but brittle and unable to capture complex non-linear dynamics.
- **Quant 2.0 (1990s-2010s):** Statistical arbitrage and factor models (ARIMA, GARCH, Fama-French factors). Economically grounded but limited by linear assumptions and Gaussian priors.
- **Quant 3.0 (2010s-2020s):** Deep learning and end-to-end optimization (LSTMs, Transformers, Deep RL). Powerful pattern recognition but opaque reasoning, regulatory risks, and catastrophic failures during regime shifts.
- **Quant 4.0 (2020s-present):** Neuro-symbolic systems combining neural perception with symbolic reasoning. The integration of vision-based pattern recognition, logical constraint satisfaction, and hierarchical decision-making.

Project JANUS represents a realization of the Quant 4.0 vision, moving beyond pure deep learning toward hybrid systems that combine the intuitive power of neural networks with the logical rigor of symbolic AI and the biological plausibility of neuromorphic architectures.

1.2 The Black Box Crisis

Modern deep reinforcement learning agents can achieve superhuman performance on complex tasks, but this performance comes at a cost: *opacity*. A DRL agent trained solely to maximize Sharpe ratio may discover strategies that:

- Exploit simulator artifacts that don't exist in live markets (reality gap)
- Violate regulatory constraints (wash sales, market manipulation, front-running)
- Fail catastrophically during regime shifts unseen in training data
- Exhibit emergent behaviors that are impossible to audit or explain

In traditional software engineering, this would be unacceptable. Financial systems require:

1. **Explainability:** Every trading decision must be auditable and defensible
2. **Safety:** Hard constraints (risk limits, regulatory rules) must be enforced
3. **Robustness:** Systems must degrade gracefully under extreme market conditions
4. **Adaptability:** Systems must learn from experience without catastrophic forgetting

The challenge is to build systems that achieve these properties while maintaining the pattern recognition power of modern deep learning.

1.3 The Neuromorphic Solution

Biological brains solve many of the same challenges faced by autonomous trading systems:

- **Real-time processing:** Sensory-motor loops operate in milliseconds
- **Multi-timescale learning:** Immediate reactions (reflexes), medium-term adaptation (skill learning), long-term consolidation (memory)
- **Constraint satisfaction:** Motor commands must respect physical constraints (joint limits, balance)
- **Homeostasis:** Internal states (hunger, stress) must be regulated
- **Memory consolidation:** Experiences are replayed and abstracted during sleep

By mapping trading functions to brain regions with analogous roles, JANUS creates a system that is both neuroscientifically inspired and functionally specialized:

This mapping is not merely metaphorical—it guides the architectural design, data flow, and optimization strategies throughout the system.

Brain Region	Neuroscience Role	Trading Role
Visual Cortex	Pattern recognition in images	GAF/ViViT market pattern detection
Prefrontal Cortex	Logic, planning, rule adherence	LTN constraint enforcement
Hippocampus	Episodic memory, replay	Experience buffer, SWR simulation
Neocortex	Long-term schemas	Consolidated strategies, vector DB
Basal Ganglia	Action selection, Go/No-Go	Dual-pathway decision engine
Cerebellum	Motor control, prediction	Order execution, market impact model
Thalamus	Attention, multimodal fusion	Gated cross-attention fusion
Amygdala	Fear, threat detection	Circuit breakers, risk alerts
Hypothalamus	Homeostasis, drive states	Risk appetite, position sizing

Table 1: Neuromorphic Mapping: Brain Regions to Trading Functions

2 Architectural Philosophy: The Two Faces of JANUS

2.1 Why Dual Architecture?

The central insight of Project JANUS is that *real-time decision-making* and *reflective learning* are fundamentally different computational regimes that should be decoupled:

- **Forward (Wake State):** Operates during market hours under extreme latency constraints (microseconds to milliseconds). Must process streaming data, fuse multimodal inputs, evaluate logical constraints, and execute trades—all while maintaining deterministic worst-case performance.
- **Backward (Sleep State):** Operates during off-market hours without latency constraints. Can perform computationally expensive operations: prioritized experience replay, UMAP dimensionality reduction, schema clustering, vector database consolidation, and model retraining.

This separation mirrors the biological wake-sleep cycle, where:

- During wakefulness, the brain prioritizes fast sensory-motor integration
- During sleep, the brain replays experiences (sharp-wave ripples in hippocampus), consolidates memories to neocortex, and prunes synapses

2.2 Janus Bifrons: The Forward Face

Janus Bifrons ("two-faced") represents the conscious, awake trader. The Forward service implements:

1. **Visual Pattern Recognition:**
    - Differentiable Gramian Angular Fields (DiffGAF) transform 1D time series into 2D images with learnable normalization
    - GAF Video sequences capture spatiotemporal market evolution
    - Video Vision Transformer (ViViT) processes multi-frame sequences with factorized spatial-temporal attention
    - Limit Order Book (LOB) heatmap fusion for microstructure awareness
2. **Symbolic Reasoning:**
    - Logic Tensor Networks (LTN) embed regulatory and risk constraints as differentiable logical predicates
    - Lukasiewicz T-Norm operations (AND, OR, NOT, IMPLIES, FORALL, EXISTS) enable gradient-based satisfaction
    - Knowledge base includes wash sale rules, Almgren-Chriss risk constraints, and VPIN toxicity thresholds
3. **Multimodal Fusion:**
    - Gated Cross-Attention fuses visual embeddings (ViViT), time series forecasts (Chronos-Bolt), and text embeddings (FinBERT)
    - Learnable gating weights determine modality importance dynamically
4. **Neuromorphic Decision Engine:**
    - Basal ganglia-inspired dual pathways: Direct (Go) and Indirect (No-Go)
    - Cerebellar forward model predicts market impact and execution cost
    - Action is authorized only when Go > No-Go and LTN constraints are satisfied

2.3 Janus Consivius: The Backward Face

Janus Consivius (“sower”, “planter”) represents the reflective, consolidating mind. The Backward service implements:

1. **Three-Timescale Memory Hierarchy:**
    - *Short-term (Hippocampus):* Episodic buffer stores raw experiences with pattern separation and sparse encoding
    - *Medium-term (SWR):* Sharp-wave ripple simulation replays high-priority experiences with 10-20× time compression

- *Long-term (Neocortex):* Schema formation via clustering and consolidation to vector database (Qdrant)

2. **Prioritized Replay:**
   - Experiences prioritized by TD-error, LTN violation severity, and recency
   - SumTree data structure for $O(\log N)$ sampling
   - Importance sampling correction to prevent bias
3. **Cognitive Visualization:**
   - AlignedUMAP projects experiences across training epochs to detect schema formation
   - Parametric UMAP enables real-time anomaly detection and regime shift visualization
4. **Vector Database Integration:**
   - Qdrant stores consolidated schemas as high-dimensional embeddings
   - Similarity search retrieves relevant past experiences for few-shot adaptation
   - Periodic pruning removes outdated or redundant schemas

2.4 Information Flow Between Services

The Forward and Backward services communicate asynchronously:

1. **During Market Hours (Forward Active):**
   - Forward service processes live market data and executes trades
   - Experiences (state, action, reward, next state, LTN violations) are written to shared episodic buffer
   - No blocking—buffer writes are lock-free and wait-free
2. **During Off-Hours (Backward Active):**
   - Backward service consumes episodic buffer
   - Prioritized replay generates training batches
   - SWR simulation compresses experiences
   - Schemas are consolidated to Qdrant
   - Updated model weights are exported (ONNX) and versioned
3. **Model Deployment:**

- Forward service loads new ONNX models during market close
- Shadow deployment allows parallel evaluation before promotion
- Graceful fallback to previous model if validation fails

3 Core Components: Hybrid Intelligence

3.1 Vision: Seeing the Market's Geometry

3.1.1 Why Visual Encoding?

Financial time series are traditionally represented as 1D sequences of scalars. This representation is efficient for storage but discards the rich topological structure of market dynamics. By transforming time series into images, we unlock the architectural power of Computer Vision:

- **Convolutional layers** detect hierarchical spatial patterns (edges, textures, shapes)
- **Translation invariance** recognizes patterns regardless of position
- **Attention mechanisms** focus on salient regions

In the context of transformed time series, these visual patterns correspond to:

- *Edges* $\rightarrow$ Regime transitions (trend reversals, volatility shifts)
- *Textures* $\rightarrow$ Microstructure patterns (volatility clustering, mean reversion)
- *Shapes* $\rightarrow$ Macrostructure formations (head-and-shoulders, flags, triangles)

3.1.2 Differentiable Gramian Angular Fields (DiffGAF)

JANUS introduces **DiffGAF**, a learnable variant of Gramian Angular Fields that enables end-to-end gradient flow from the visual classifier back through the imaging transformation.

Given a time series $X = \{x_1, x_2, \ldots, x_n\}$:

Step 1: Learnable Normalization

$$\tilde{x}_i = \tanh\left(\frac{x_i - \min(X)}{\max(X) - \min(X)} \cdot \alpha + \beta\right) \tag{1}$$

where α, β are learnable parameters optimized during training.

Step 2: Polar Encoding

$$\begin{cases} \phi_i = \arccos(\tilde{x}_i), & -1 \leq \tilde{x}_i \leq 1 \\ r_i = \frac{t_i}{N}, & t_i \in \mathbb{N} \end{cases} \tag{2}$$

Step 3: Gramian Field Generation

Gramian Angular Summation Field (GASF):

$$\text{GASF}_{i,j} = \cos(\phi_i + \phi_j) \tag{3}$$

Gramian Angular Difference Field (GADF):

$$\text{GADF}_{i,j} = \sin(\phi_i - \phi_j) \tag{4}$$

The resulting $n \times n$ image preserves temporal correlation structure while enabling spatial convolution.

3.1.3 GAF Video and ViViT

Static images capture instantaneous market state. To capture *evolution*, JANUS generates GAF video sequences using sliding windows:

$$\mathcal{V} = \{GAF(X_{t:t+w}), GAF(X_{t+s:t+w+s}), \ldots\} \tag{5}$$

where w is window size and s is stride. The resulting 3D tensor $\mathcal{V} \in \mathbb{R}^{F \times H \times W}$ is processed by a Video Vision Transformer (ViViT) with:

- **Spatial attention** within each frame (volatility clusters, trend structures)
- **Temporal attention** across frames (regime transitions, momentum shifts)

3.2 Logic: Enforcing the Rules of the Game

3.2.1 The Necessity of Symbolic Constraints

Deep learning excels at pattern recognition but lacks logical precision. In finance, "close enough" is unacceptable:

- A wash sale (selling at a loss and repurchasing within 30 days) triggers tax penalties
- Exceeding risk limits violates Almgren-Chriss constraints and regulatory mandates
- Order flow toxicity (high VPIN) indicates informed trading and adverse selection risk

These are not soft preferences—they are *hard constraints* that must be satisfied with mathematical certainty.

3.2.2 Logic Tensor Networks (LTN)

LTN embeds First-Order Logic (FOL) into neural networks by grounding logical symbols in continuous tensor space:

- **Constants** $\rightarrow$ Tensor embeddings
- **Predicates** $\rightarrow$ Neural networks outputting $[0, 1]$ truth values
- **Logical connectives** $\rightarrow$ Fuzzy logic t-norms

JANUS uses Lukasiewicz t-norms for improved gradient flow:

$$\text{AND}(p, q) = \max(0, p + q - 1) \tag{6}$$
$$\text{OR}(p, q) = \min(1, p + q) \tag{7}$$
$$\text{NOT}(p) = 1 - p \tag{8}$$
$$\text{IMPLIES}(p, q) = \min(1, 1 - p + q) \tag{9}$$

3.2.3 Knowledge Base Examples

Wash Sale Constraint:

$$\forall t : \text{Sold}(t) \wedge \text{Loss}(t) \implies \neg\text{Buy}(t, t + 30) \tag{10}$$

"If a position was sold at a loss, do not repurchase within 30 days."
Almgren-Chriss Risk Constraint:

$$\forall \tau : \text{Variance}(\tau) \leq \lambda \cdot \text{MarketImpact}(\tau) \tag{11}$$

"Execution variance must not exceed risk tolerance."
VPIN Toxicity Constraint:

$$\text{VPIN}(t) > \theta \implies \text{Widen}(\text{spread}) \vee \text{Halt}(\text{trading}) \tag{12}$$

"If order flow toxicity exceeds threshold, widen spreads or halt."

3.3 Fusion: Integrating Multiple Realities

3.3.1 The Multimodal Challenge

Markets are inherently multimodal. A complete understanding requires:

- **Visual patterns** (GAF video, LOB heatmaps)

- **Time series forecasts** (Chronos-Bolt probabilistic predictions)
- **Text semantics** (news, earnings calls, social media)
- **Logical constraints** (LTN predicate evaluations)

The challenge is not merely concatenating these modalities but learning their relative importance dynamically.

3.3.2 Gated Cross-Attention (GCA)

JANUS uses GCA to fuse modalities with learnable gating:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{13}$$

$$g = \sigma(W_g \cdot [x_{\text{visual}}; x_{\text{time}}; x_{\text{text}}] + b_g) \tag{14}$$

$$x_{\text{fused}} = g \odot \text{Attention}(x_{\text{visual}}, x_{\text{time}}, x_{\text{time}}) \tag{15}$$

The gating weight $g \in [0, 1]$ determines how much to trust each modality. During high volatility, visual patterns may dominate; during earnings season, text embeddings may be prioritized.

3.4 Decision: The Neuromorphic Motor System

3.4.1 Basal Ganglia Dual Pathways

The basal ganglia implements action selection via competing pathways:

- **Direct Pathway (Go):** Facilitates action execution
- **Indirect Pathway (No-Go):** Inhibits action execution

JANUS mirrors this with dual neural networks:

$$\text{Go}(s) = f_{\text{direct}}(s; \theta_{\text{Go}}) \tag{16}$$

$$\text{NoGo}(s) = f_{\text{indirect}}(s; \theta_{\text{NoGo}}) \tag{17}$$

$$a = \begin{cases} a_{\text{proposed}} & \text{if Go}(s) > \text{NoGo}(s) \wedge \text{LTN}(s, a) > \tau \\ \text{HOLD} & \text{otherwise} \end{cases} \tag{18}$$

This architecture naturally implements risk-averse behavior: actions are blocked unless both pathways agree AND logical constraints are satisfied.

3.4.2 Cerebellar Forward Model

The cerebellum predicts sensory consequences of motor commands. JANUS implements a market impact predictor:

$$\hat{p}_{\text{fill}} = f_{\text{cerebellum}}(a, s_{\text{LOB}}) \tag{19}$$

This prediction enables:

- **Trajectory adjustment:** Modify order size/timing to minimize slippage
- **Error correction:** Compare predicted vs. actual fill price and update model

4 Memory and Learning: The Sleeping Mind

4.1 The Three-Timescale Hierarchy

Biological memory systems operate at multiple timescales:

- **Hippocampus:** Rapid encoding of episodic details (seconds to hours)
- **Sharp-Wave Ripples:** Replay and consolidation during rest (minutes to hours)
- **Neocortex:** Long-term schemas and abstract knowledge (days to years)

JANUS replicates this hierarchy:

4.1.1 Short-Term: Hippocampal Episodic Buffer

Raw experiences are stored with minimal processing:

$$e_t = (s_t, a_t, r_t, s_{t+1}, \text{done}_t, \text{LTN}_t, \text{meta}_t) \tag{20}$$

where LTN_t records constraint violations and meta_t includes timestamps, market regime labels, etc.

Pattern separation ensures diverse storage:

$$\text{similarity}(e_i, e_j) < \tau_{\text{sep}} \implies \text{store both} \tag{21}$$

4.1.2 Medium-Term: Sharp-Wave Ripple (SWR) Simulation

During sleep, the hippocampus replays experiences at 10-20× real-time speed. JANUS simulates this with:

1. **Prioritized sampling:** Select high TD-error, high LTN-violation experiences
2. **Time compression:** Process entire trading day in 10-30 minutes
3. **Batch generation:** Create training batches for model updates

Priority weighting:

$$p_i = (|\delta_i| + \epsilon)^\alpha + \beta \cdot \text{LTN\_violation}_i + \gamma \cdot \text{recency}_i \tag{22}$$

4.1.3 Long-Term: Neocortical Schemas

Repeated experiences are abstracted into schemas—general patterns that transcend specific episodes:

$$\theta_{\text{schema}} \leftarrow \theta_{\text{schema}} + \eta \cdot \text{recall\_gate} \cdot \nabla_\theta \mathcal{L} \tag{23}$$

Recall gating prevents interference:

$$\text{recall\_gate} = \mathbb{1}[\text{cosine\_similarity}(e_{\text{current}}, \text{schema}) > \tau_{\text{recall}}] \tag{24}$$

Only experiences similar to existing schemas contribute to consolidation, reducing catastrophic forgetting.

4.2 Cognitive Visualization: UMAP

4.2.1 AlignedUMAP for Schema Detection

During training, experiences are projected to 2D space across epochs:

$$\mathcal{L}_{\text{UMAP}} = \sum_{i,j} w_{ij} \log\left(\frac{1}{1 + \|y_i - y_j\|^2}\right) + (1 - w_{ij}) \log\left(1 - \frac{1}{1 + \|y_i - y_j\|^2}\right) \tag{25}$$

Cluster formation in UMAP space indicates emerging schemas. Cluster stability across epochs validates consolidation.

4.2.2 Parametric UMAP for Anomaly Detection

A neural network learns the UMAP projection:

$$y = f_{\text{UMAP}}(x; \theta) \tag{26}$$

At inference time, new experiences are projected in real-time. Outliers indicate:

- Regime shifts (market structure change)
- Novel strategies (unexplored state space)
- Data quality issues (sensor failures)

5 Implementation Strategy: Rust-First Architecture

5.1 Why Rust?

Financial systems demand:

- **Performance:** Microsecond latencies, zero-copy operations
- **Safety:** No null pointers, no data races, no undefined behavior
- **Reliability:** Predictable performance, deterministic memory usage
- **Auditability:** Strong type system, explicit error handling

Rust provides all of these without garbage collection pauses or runtime overhead.

5.2 Service Architecture

JANUS is implemented as two Rust services with a Python training gateway:

1. **Forward Service (Rust):**
   - Async runtime (Tokio) for concurrent market data streams
   - ONNX Runtime for model inference (ViViT, LTN predicates)
   - Lock-free queues for experience buffer writes
   - gRPC API for order submission
2. **Backward Service (Rust):**
   - Rayon for parallel batch processing
   - SumTree (custom implementation) for prioritized replay
   - Qdrant client for vector database operations
   - UMAP projection (via ONNX or direct Rust port)
3. **Training Gateway (Python):**
   - PyTorch for model training
   - FastAPI for REST endpoints
   - Celery for async batch jobs (SWR simulation, model export)
   - Model export to ONNX for Rust consumption

5.3 ML Framework Migration Path

1. **Phase 1 (Months 1-3):** Hybrid PyTorch + ONNX
    - Train in PyTorch, export to ONNX, infer in Rust
    - Establish benchmarks and validation pipelines
2. **Phase 2 (Months 4-6):** Rust-native inference
    - Port inference to tch-rs (libtorch) or Candle
    - Eliminate Python dependency for deployment
3. **Phase 3 (Months 7-12):** Full Rust ML stack
    - Port training to Burn or Candle
    - Single-language codebase for maximum auditability

5.4 Deployment Architecture

Development:

- Docker Compose for local multi-service orchestration
- Shared volumes for model artifacts and experience buffers
- PostgreSQL for metadata, Qdrant for vectors, Redis for pub/sub

Production:

- Kubernetes for orchestration and auto-scaling
- StatefulSets for Qdrant and PostgreSQL
- Helm charts for versioned deployments
- Prometheus + Grafana for monitoring
- Shadow deployment for A/B testing

6 Safety and Compliance: The Glass Box

6.1 Architectural Invariants

JANUS enforces architectural invariants through Rust's type system:

1. **No Panics in Hot Path:**
   - All errors are typed (using `thiserror`)
   - `unwrap()` and `expect()` forbidden in production code
   - Fallback strategies for all error conditions
2. **LTN Constraints Always Evaluated:**
   - Actions cannot be executed without LTN evaluation
   - Constraint violations logged and traced
   - Circuit breakers halt trading on repeated violations
3. **Memory Bounds Enforced:**
   - Episodic buffer has maximum size with FIFO eviction
   - No unbounded allocations in hot path
   - Pre-allocated buffers for latency-critical operations
4. **Temporal Guarantees:**
   - SWR compression factor $\in [10, 20]$ verified at runtime
   - Replay cannot exceed configured wall-clock time
   - Timeout guards on all external API calls

6.2 Explainability and Auditability

Every trading decision generates an audit trail:

1. **Visual Attention:** Grad-CAM heatmaps show which GAF frames influenced the decision
2. **Logical Trace:** LTN evaluations are logged with predicate truth values
3. **Modality Weights:** GCA gating values indicate which inputs were trusted
4. **Pathway Activation:** Go/No-Go scores explain action selection
5. **Forward Model Error:** Predicted vs. actual slippage quantifies model confidence

This trace is stored in structured logs (JSON) and indexed for regulatory queries.

6.3 Circuit Breakers and Kill Switches

The **Amygdala** subsystem implements safety-critical overrides:

1. **Volatility Spike:** Halt trading if realized volatility $> 3\times$ historical
2. **Drawdown Limit:** Halt if portfolio drawdown $> 10\%$ in single session
3. **LTN Violation Rate:** Halt if $> 5\%$ of actions violate constraints
4. **Execution Anomaly:** Halt if average slippage $> 2\times$ predicted
5. **Manual Override:** Human operators can trigger emergency halt via API

All circuit breakers are *fail-safe*: they default to halting trading on error or uncertainty.

7 Validation and Testing: Proving Robustness

7.1 Simulation and Backtesting

JANUS is validated through progressive realism:

1. **Unit Tests:** Each component tested in isolation
   - GAF transformation invertibility
   - LTN predicate gradient flow
   - SumTree priority sampling correctness
2. **Synthetic Markets:** Controlled environments with known properties
   - Mean-reverting Ornstein-Uhlenbeck process
   - Momentum-driven geometric Brownian motion
   - Regime-switching models with abrupt transitions
3. **Historical Replay:** Real market data with simulated execution
   - Out-of-sample testing on unseen time periods
   - Walk-forward optimization to prevent overfitting
   - Comparison with baseline strategies (buy-and-hold, momentum, mean-reversion)
4. **Black Swan Stress Tests:** Extreme scenarios
   - Flash crash (May 6, 2010)
   - COVID crash (March 2020)
   - GameStop short squeeze (January 2021)
5. **Paper Trading:** Live market data with simulated execution
   - Reality gap monitoring (predicted vs. actual slippage)
   - Latency profiling under production load
6. **Shadow Deployment:** Parallel execution with production system
   - JANUS generates signals but does not execute
   - Performance compared with existing live system
   - Gradual rollout (1% $\rightarrow$ 10% $\rightarrow$ 50% $\rightarrow$ 100%)

7.2 Comparative Benchmarks

JANUS is compared against:

- **Baseline:** Buy-and-hold, equal-weight portfolio
- **Traditional Quant:** ARIMA, GARCH, Fama-French factors
- **Deep Learning:** LSTM, Transformer, pure DRL (PPO, SAC)
- **Neuro-Symbolic (No Vision):** LTN + time series only
- **Vision-Only (No Logic):** ViViT without LTN constraints

Metrics evaluated:

- Sharpe Ratio, Sortino Ratio, Calmar Ratio
- Maximum Drawdown, Value-at-Risk (VaR), Conditional VaR
- Trade execution quality (average slippage, fill rate)
- LTN constraint violation rate (should be ≈ 0)
- Latency percentiles (p50, p95, p99, p99.9)

8 Future Directions: Towards Quant 5.0

8.1 Quantum Computing Integration

Quantum algorithms offer potential advantages for:

- **Portfolio Optimization:** Quadratic Unconstrained Binary Optimization (QUBO) via quantum annealing
- **Option Pricing:** Quantum Monte Carlo with exponential speedup
- **Risk Analysis:** Quantum amplitude estimation for tail risk computation

JANUS is designed to integrate quantum co-processors via:

- Modular risk engine interface (classical or quantum backend)
- Hybrid classical-quantum optimization loops
- Benchmarking quantum advantage on specific subproblems

8.2 Continual Learning and Meta-Learning

Current limitations:

- Models are retrained periodically but not continuously
- Catastrophic forgetting when market regimes shift
- Slow adaptation to novel market conditions

Future enhancements:

- **Elastic Weight Consolidation (EWC):** Protect important weights from updates
- **Meta-Learning (MAML):** Learn initialization that adapts quickly to new regimes
- **Lifelong Learning:** Incremental schema formation without full retraining

8.3 Multi-Agent Cooperation and Competition

Markets are inherently multi-agent. Future JANUS versions may include:

- **Population-Based Training:** Multiple JANUS instances with diverse strategies
- **Cooperative Agents:** Agents share schemas via federated learning
- **Adversarial Agents:** Simulate market makers, informed traders, noise traders
- **Game-Theoretic Equilibria:** Nash equilibrium strategies in multi-agent auctions

8.4 Regulatory AI and Automated Compliance

Symbolic reasoning can be extended to:

- **Automated Regulation Parsing:** Convert SEC rules to LTN predicates
- **Real-Time Compliance Monitoring:** Flag potential violations before execution
- **Regulatory Stress Testing:** Simulate proposed rule changes on strategy performance

9 Conclusion: The Path Forward

Project JANUS represents a synthesis of multiple frontiers in AI and computational finance:

- **Computer Vision:** GAF transforms time series into images, unlocking CNNs and ViTs
- **Symbolic AI:** LTN embeds logical constraints as differentiable predicates
- **Neuroscience:** Neuromorphic architecture mirrors biological brain organization
- **Systems Engineering:** Rust-first implementation ensures safety and performance
- **Memory Systems:** Wake-sleep cycle separates real-time inference from reflective learning

The dual architecture—Forward for real-time trading, Backward for consolidation—mirrors the biological imperative of balancing immediate response with long-term adaptation. By decoupling these concerns, JANUS achieves both microsecond latencies and deep contemplative learning.

The neuromorphic mapping is not metaphorical. Each brain region corresponds to a specific computational challenge in autonomous trading:

- Visual cortex detects patterns in transformed time series
- Prefrontal cortex enforces regulatory and risk constraints
- Hippocampus stores episodic experiences for replay
- Basal ganglia adjudicates action selection via dual pathways
- Cerebellum predicts execution outcomes
- Amygdala implements circuit breakers and risk alerts

This architecture is *explainable by design*. Every decision generates an audit trail tracing:

1. Which visual patterns were detected (Grad-CAM)
2. Which logical constraints were evaluated (LTN trace)
3. Which modalities were trusted (GCA gating)
4. Why the action was authorized (Go/No-Go scores)

This transparency is not incidental—it is the system's core value proposition. In an era of black box AI failures, JANUS offers a "glass box" alternative: a system that is both powerful and accountable.

9.1 Companion Documents

This main document provides the conceptual framework. Technical details are in:

1. **janus_forward.tex** — Algorithms for visual encoding, LTN reasoning, multimodal fusion, and decision-making in the Forward service
2. **janus_backward.tex** — Memory hierarchy, prioritized replay, SWR simulation, UMAP visualization, and vector database integration in the Backward service
3. **janus_neuromorphic_architecture.tex** — Complete neuromorphic mapping, brain region implementations, information flow diagrams, and architectural invariants
4. **janus_rust_implementation.tex** — Rust module structure, ML framework strategy, async service architecture, deployment pipelines, and implementation roadmap

9.2 Call to Action

The transition from Quant 3.0 to Quant 4.0 is not merely an incremental improvement—it is a paradigm shift. The systems we build today will shape the financial markets of the next decade. JANUS offers a blueprint for autonomous trading systems that are:

- **Powerful:** Leveraging state-of-the-art deep learning for pattern recognition
- **Safe:** Enforcing constraints through symbolic reasoning and fail-safe design
- **Transparent:** Generating auditable explanations for every decision
- **Adaptive:** Learning from experience through biologically-inspired memory systems
- **Scalable:** Implemented in Rust for production-grade performance and reliability

The future of quantitative finance will be shaped not by monolithic black boxes, but by hybrid systems that integrate the best of connectionist and symbolic AI, guided by the architectural principles discovered through millions of years of biological evolution.

JANUS is the first step on this path. The journey continues.

"The Roman god Janus is the god of transitions, passages, and new beginnings.
Looking simultaneously to the past and the future,
he embodies the duality required for true intelligence:
learning from history while adapting to the unknown."

References

1. Wang, Z., & Oates, T. (2015). *Imaging time-series to improve classification and imputation*. Proceedings of IJCAI.

2. Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., & Schmid, C. (2021). *ViViT: A Video Vision Transformer*. ICCV 2021.

3. Badreddine, S., d'Avila Garcez, A., Serafini, L., & Spranger, M. (2022). *Logic Tensor Networks*. Artificial Intelligence, 303, 103649.

4. Almgren, R., & Chriss, N. (2001). *Optimal execution of portfolio transactions*. Journal of Risk, 3, 5-40.

5. Daw, N. D., Niv, Y., & Dayan, P. (2005). *Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control*. Nature Neuroscience, 8(12), 1704-1711.

6. Buzsáki, G. (2015). *Hippocampal sharp wave-ripple: A cognitive biomarker for episodic memory and planning*. Hippocampus, 25(10), 1073-1188.

7. McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). *Why there are complementary learning systems in the hippocampus and neocortex*. Psychological Review, 102(3), 419.

8. McInnes, L., Healy, J., & Melville, J. (2018). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. arXiv:1802.03426.

9. Easley, D., López de Prado, M. M., & O'Hara, M. (2012). *Flow toxicity and liquidity in a high-frequency world*. The Review of Financial Studies, 25(5), 1457-1493.

10. Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). *Prioritized experience replay*. ICLR 2016.

11. Veličković, P., Ying, R., Padovano, M., Hadsell, R., & Blundell, C. (2019). *Neural execution of graph algorithms*. ICLR 2020.

12. Ansari, A. F., et al. (2024). *Chronos: Learning the language of time series*. Amazon Science.

13. Argyris, C., & Schön, D. A. (1974). *Theory in practice: Increasing professional effectiveness*. Jossey-Bass.

Appendix: Implementation Checklist

Forward Service (Rust)

- ☐ GAF transformation module with learnable parameters
- ☐ GAF video generation with sliding windows
- ☐ ONNX Runtime integration for ViViT inference
- ☐ LTN predicate evaluation engine
- ☐ Lukasiewicz t-norm implementations
- ☐ Gated cross-attention fusion
- ☐ Basal ganglia dual pathways (Go/No-Go)
- ☐ Cerebellar forward model for slippage prediction
- ☐ Lock-free experience buffer writes
- ☐ gRPC API for order submission
- ☐ Async runtime (Tokio) for market data streams
- ☐ Circuit breaker implementations (amygdala)
- ☐ Audit logging (JSON structured logs)
- ☐ Prometheus metrics export

Backward Service (Rust)

- ☐ Prioritized replay buffer with SumTree
- ☐ SWR simulation with time compression
- ☐ Importance sampling correction
- ☐ Schema formation via clustering
- ☐ Recall-gated consolidation
- ☐ Qdrant client for vector database
- ☐ UMAP projection (ONNX or native)
- ☐ AlignedUMAP for multi-epoch analysis

- ☐ Parametric UMAP for real-time anomaly detection
- ☐ Rayon parallel batch processing
- ☐ Model export and versioning
- ☐ Schema pruning logic

Training Gateway (Python)

- ☐ PyTorch training loop
- ☐ FastAPI REST endpoints
- ☐ Celery task queue for async jobs
- ☐ ONNX export pipeline
- ☐ Model validation scripts
- ☐ Hyperparameter tuning (Optuna)
- ☐ Experiment tracking (MLflow or Weights & Biases)

Infrastructure

- ☐ Docker Compose for local development
- ☐ Kubernetes manifests (Deployments, Services, ConfigMaps)
- ☐ Helm charts for versioned releases
- ☐ PostgreSQL for metadata
- ☐ Qdrant for vector storage
- ☐ Redis for pub/sub and caching
- ☐ Prometheus + Grafana monitoring
- ☐ CI/CD pipeline (GitHub Actions or GitLab CI)
- ☐ Shadow deployment workflow
- ☐ Automated backtesting on commit

Validation

- ☐ Unit tests (>80% coverage)
- ☐ Integration tests (end-to-end flows)
- ☐ Synthetic market simulations
- ☐ Historical backtest suite
- ☐ Black swan stress tests
- ☐ Paper trading validation
- ☐ Latency profiling
- ☐ Reality gap analysis