Project JANUS

Neuromorphic Trading Intelligence

Complete Technical Specification

A Brain-Inspired Architecture for Autonomous Financial Systems

Unified Documentation

This document consolidates all technical specifications of Project JANUS:

1. **Main Architecture** — System design and philosophical foundation
2. **Forward Service** — Real-time decision-making and execution
3. **Backward Service** — Memory consolidation and learning
4. **Neuromorphic Architecture** — Brain-region mapping
5. **Rust Implementation** — Production deployment guide
6. **Advanced Neuroscience Integration** — Extended brain-inspired mechanisms

Author

Jordan Smith

Date

February 15, 2026

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

Contents

Part I

Main Architecture

Overview

The Epistemological Transition to Quant 4.0

The trajectory of algorithmic trading has historically been defined by a tension between interpretability and capability. We are currently witnessing a phase transition from the "black box" empiricism of deep learning (**lecun2015deep**) toward a new paradigm of Neuro-Symbolic integration (**garcez2024mapping**). Project JANUS stands at the vanguard of this transition, termed **Quant 4.0**. This architecture does not merely iterate on existing statistical methods but fundamentally reimagines the financial agent as a biological entity—one that perceives, reasons, remembers, and fears.

Historical Evolution of Quantitative Finance

- **Quant 1.0 (1980s-1990s):** Era of heuristics and expert systems with high interpretability but extreme rigidity
- **Quant 2.0 (1990s-2000s):** Statistical rigour through mean reversion, cointegration, and factor models (**fama1993common**)
- **Quant 3.0 (2010s-Present):** Deep learning hegemony with LSTMs, Transformers (**vaswani2017attention**), and Deep Reinforcement Learning
- **Quant 4.0 (JANUS):** Neuro-Symbolic AI achieving adaptability of deep learning with reliability of rule-based systems

The Dual-Process Architecture

The architectural philosophy of JANUS is strictly biomimetic, mirroring the Dual-Process Theory of cognition (**kahneman2011thinking**; **evans2008dual**). The system is bifurcated into two distinct but interacting services:

Service	Persona	Cognitive Role	Biological Analogue
Forward Service	Janus Bifrons	Perception & Action	Basal Ganglia & Thalamus
Backward Service	Janus Consivius	Memory & Learning	Hippocampus & Neocortex

This separation allows JANUS to optimize for latency on the hot path (Forward Service) while reserving heavy computational resources for consolidation and schema formation on the cold path (Backward Service), implementing the Complementary Learning Systems theory (**mcclelland1995complementary**).

In addition to the core Forward and Backward services, the production system deploys three supporting services: an **Execution Service** for multi-exchange order routing, a **Data Service** for centralized market data management, and a **CNS (Central Nervous System) Service** for system-wide health monitoring and preflight validation (Section 6).

Note: The detailed mathematical specifications for each component are presented in Parts 2-6 below.

Part II

Forward Service (Janus Bifrons)

Abstract

JANUS Forward represents the "wake state" of the JANUS trading system, responsible for all real-time decision-making during market hours. This service combines:

- **Visual Pattern Recognition** using Gramian Angular Fields (GAF) (**wang2015imaging**) with LSTM and Video Vision Transformer (ViViT) (**arnab2021vivit**) temporal modeling
- **Symbolic Reasoning** via Logic Tensor Networks (LTN) (**badreddine2022logic**) for constraint satisfaction
- **Multimodal Fusion** integrating time series, visual, order book, and sentiment data through specialized fusion engines
- **Dual-Pathway Decision Making** inspired by basal ganglia architecture (**collins2014opponent**
- **Ensemble Regime Detection** combining Hidden Markov Models, statistical, and technical methods for market state identification

The Forward service operates on a hot path with strict latency requirements, implementing a six-stage neural pipeline (`regime` → `hypothalamus` → `amygdala` → `gating` → `correlation` → `execution`) that routes signals through brain regions in real time. FPGA acceleration (**marino2023mevit**; **vemeko2023fpga**) is planned for nanosecond-level latency in future high-frequency trading applications.

1 Visual Pattern Recognition: DiffGAF and Vision Models

The visual subsystem transforms time series data into spatiotemporal images, enabling the system to "see" market patterns that traditional numerical methods miss. This approach is grounded in the work of **wang2015imaging**, who demonstrated that imaging time series significantly improves classification and imputation tasks by exposing temporal correlations to the inductive biases of convolutional neural networks.

1.1 Mathematical Foundation: Gramian Angular Fields

Time series are encoded into polar coordinates and projected onto Gramian matrices, creating 2D representations that preserve temporal correlations (**wang2015imaging**). Recent research (**fusion2025gaf**) has validated that GAF encodings substantially outperform raw time-series inputs in classification tasks by capturing multi-scale temporal structures.

1.1.1 Input Preprocessing

Given raw market data $X = \{x_1, x_2, \ldots, x_T\}$ where $x_t \in \mathbb{R}^D$ (multi-feature time series), we first apply feature selection to extract F relevant features.

1.1.2 Step 1: Normalization

For inference, we apply min-max normalization to the domain $[-1, 1]$, followed by Piecewise Aggregate Approximation (PAA) for temporal resizing:

$$\tilde{x}_t = 2 \cdot \frac{x_t - x_{\min}}{x_{\max} - x_{\min}} - 1 \tag{1}$$

ensuring the subsequent $\arccos$ operation is well-defined with $\tilde{x}_t \in [-1, 1]$.

For the training pipeline, we employ learnable affine transformations with domain constraints:

$$\tilde{x}_t = \tanh\left(\gamma \odot \frac{x_t - \mu}{\sigma} + \beta\right) \tag{2}$$

where $\gamma, \beta \in \mathbb{R}^F$ are learned parameters, and μ, σ are running statistics. The $\tanh$ function guarantees $\tilde{x}_t \in (-1, 1)$.

1.1.3 Step 2: Polar Coordinate Transformation

Map normalized values to angular space:

$$\phi_t = \arccos(\tilde{x}_t) \in [0, \pi] \tag{3}$$

$$r_t = \frac{t}{T} \quad \text{(normalized timestamp)} \tag{4}$$

1.1.4 Step 3: Gramian Field Generation

Construct the Gramian Angular Summation Field (GASF):

$$\mathbf{G}_{ij} = \cos(\phi_i + \phi_j) = \tilde{x}_i \tilde{x}_j - \sqrt{1 - \tilde{x}_i^2}\sqrt{1 - \tilde{x}_j^2} \tag{5}$$

Or the Gramian Angular Difference Field (GADF), which JANUS employs to encode velocity of price changes as visual textures:

$$\mathbf{G}_{ij} = \sin(\phi_i - \phi_j) = \sqrt{1 - \tilde{x}_i^2}\tilde{x}_j - \tilde{x}_i\sqrt{1 - \tilde{x}_j^2} \tag{6}$$

This transformation allows the system to visually perceive volatility regimes and microstructure dynamics (**wang2015imaging**).

1.1.5 Differentiable GAF (DiffGAF)

The Rust implementation provides a fully differentiable GAF engine with analytically computed Jacobians for end-to-end gradient flow. The key derivative through the polar mapping is:

$$\frac{d\phi_k}{d\tilde{x}_k} = \frac{-1}{\sqrt{1 - \tilde{x}_k^2}} \tag{7}$$

which enables backpropagation through the entire GAF encoding pipeline. Numerical gradient verification tests confirm correctness within 10^{-4} tolerance.

1.2 3D Spatiotemporal Manifolds: GAF Video

To capture temporal dynamics, we generate a sequence of GAF frames using sliding windows.

1.2.1 Sliding Window GAF Video Generation

Given a time series of length T, window size W, and stride S:

1. Extract windows: $X_k = \{x_{(k-1)S+1}, \ldots, x_{(k-1)S+W}\}$ for $k = 1, \ldots, N$
2. Generate GAF for each window: $\mathbf{G}_k = \text{GAF}(X_k) \in \mathbb{R}^{W \times W}$
3. Stack into video: $\mathbf{V} = [\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_N] \in \mathbb{R}^{N \times W \times W}$

The data ingestion pipeline handles windowed buffering and streaming through dedicated preprocessing modules.

1.3 Vision Model Architecture

1.3.1 DiffGAF-LSTM

The DiffGAF-LSTM vision model pairs the DiffGAF encoding with an LSTM temporal model. GAF frames are encoded and fed sequentially into an LSTM network that captures inter-frame temporal dependencies. This architecture provides robust performance while maintaining the differentiability required for end-to-end training.

1.3.2 Video Vision Transformer (ViViT)

The ViViT model uses a factorized spatiotemporal transformer (**arnab2021vivit**). Unlike standard Vision Transformers (**dosovitskiy2020image**) which process static images, ViViT factorizes attention across both space (price/volume levels of the limit order book) and time (sequences of LOB snapshots), enabling the system to track dynamic microstructure events.

Patch Embedding: Divide each frame $\mathbf{G}_k$ into non-overlapping patches:

$$\mathbf{P}_k = \text{Reshape}(\mathbf{G}_k) \in \mathbb{R}^{P\times(p^2)} \tag{8}$$

where $P = (W/p)^2$ is the number of patches per frame.

Spatial Attention: Apply self-attention within each frame:

$$\mathbf{Z}_k^{(l)} = \text{MSA}(\text{LN}(\mathbf{Z}_k^{(l-1)})) + \mathbf{Z}_k^{(l-1)} \tag{9}$$

Temporal Attention: Apply attention across frames:

$$\mathbf{H}^{(l)} = \text{MSA}(\text{LN}([\mathbf{Z}_1^{(l)}, \dots, \mathbf{Z}_N^{(l)}])) \tag{10}$$

Both the DiffGAF-LSTM and ViViT architectures are implemented natively in Rust, with the system selecting the appropriate model based on configuration and available resources. The ViViT model gracefully degrades to the DiffGAF-LSTM model when computational constraints require it.

2 Ensemble Regime Detection

Market regime identification is a critical upstream component that conditions all downstream decision-making. JANUS employs an ensemble approach combining multiple detection methods to robustly classify market states.

2.1 Detection Methods

2.1.1 Hidden Markov Model (HMM)

A Gaussian HMM models latent regime states with observable market features:

$$P(\mathbf{x}_t \mid z_t = k) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{11}$$

where $z_t \in \{1, \dots, K\}$ is the latent regime state and the transition matrix $\mathbf{A}_{ij} = P(z_{t+1} = j \mid z_t = i)$ captures regime persistence and switching dynamics.

2.1.2 Statistical Methods

Complementing the HMM, JANUS applies rolling statistical tests including variance ratio tests for mean-reversion detection, Hurst exponent estimation for trend persistence, and distributional tests for fat-tail identification.

2.1.3 Technical Methods

Classical technical analysis signals (trend strength indicators, volatility measures, momentum oscillators) are fused with the statistical methods to provide corroborating evidence for regime classification.

2.2 Regime Categories

The ensemble classifier identifies seven market regimes, each mapped to distinct trading behavior profiles:

Regime	**Characteristics**	**System Behavior**
Bull	Sustained upward trend, expanding volume	Amplify Direct (Go) pathway
Bear	Sustained downward trend, risk-off sentiment	Amplify Indirect (No-Go) pathway
Ranging	Low directional conviction, bounded price action	Favor mean-reversion strategies
Crisis	Extreme volatility, correlation breakdown	Engage Amygdala circuit breakers
Recovery	Post-crisis normalization, decreasing volatility	Gradually release risk constraints
Bubble	Parabolic price acceleration, euphoric sentiment	Heighten caution via Hypothalamus
Deflation	Sustained contraction, liquidity withdrawal	Reduce position sizing aggressively

The detected regime feeds into the Hypothalamus module for homeostatic regulation and the Basal Ganglia for dopamine-modulated action selection, forming the first stage of the brain wiring pipeline.

3 Logic Tensor Networks: Symbolic Reasoning Engine

LTNs bridge neural networks and first-order logic, enabling differentiable constraint satisfaction (**badreddine2022logic**). This neuro-symbolic approach allows JANUS to enforce regulatory and risk constraints directly within the gradient descent optimization, a capability absent in pure deep learning systems (**garcez2024mapping**).

3.1 Mathematical Foundation

The central innovation of LTNs (**badreddine2022logic**) is the ability to make Boolean logic differentiable using Real Logic, specifically Łukasiewicz t-norms (**lukasiewicz2024deep**).

3.1.1 Grounding Function

Map logical constants to real vectors:

$$\mathcal{G} : \mathcal{C} \to \mathbb{R}^d \tag{12}$$

3.1.2 Predicate Grounding

A predicate $P(x)$ is grounded as a neural network $f_\theta : \mathbb{R}^d \to [0, 1]$, where truth values range continuously from 0 to 1 rather than being discrete binary values.

3.2 Łukasiewicz T-Norm Operations

Following **badreddine2022logic** and **lukasiewicz2024deep**, we employ fuzzy logic operators that are differentiable and thus compatible with backpropagation.

3.2.1 Conjunction (AND)

For training, we use Product Logic to ensure smooth gradients:

$$u \land v = u \cdot v \tag{13}$$

For inference/evaluation, standard Łukasiewicz logic is used:

$$u \land v = \max(0, u + v - 1) \tag{14}$$

3.2.2 Disjunction (OR)

$$u \lor v = \min(1, u + v) \tag{15}$$

3.2.3 Negation (NOT)

$$\neg u = 1 - u \tag{16}$$

3.2.4 Implication (IF-THEN)

For training (Product Logic):

$$u \Rightarrow v = 1 - u + u \cdot v \tag{17}$$

For inference (Łukasiewicz Logic):

$$u \Rightarrow v = \min(1, 1 - u + v) \tag{18}$$

3.2.5 Bi-Implication (Equivalence)

$$u \Leftrightarrow v = 1 - |u - v| \tag{19}$$

3.2.6 Linguistic Hedges

The implementation extends classical fuzzy logic with linguistic hedge operators for nuanced truth-value modification:

$$\text{very}(x) = x^2 \tag{20}$$

$$\text{somewhat}(x) = \sqrt{x} \tag{21}$$

$$\text{slightly}(x) = x - x^2 \tag{22}$$

$$\text{extremely}(x) = x^3 \tag{23}$$

These hedges allow more expressive constraint formulation (e.g., “very risky” vs. “somewhat risky” conditions).

3.3 Knowledge Base Formulation

The knowledge base $\mathcal{KB}$ encodes regulatory constraints and risk management rules as logical predicates. The implementation provides a comprehensive compliance engine covering multiple constraint categories.

3.3.1 Wash Sale Constraint

The Wash Sale Rule (**irs2024wash**)—a critical regulatory constraint for active traders—prevents claiming tax losses on securities sold and repurchased within 30 days. The implementation enforces the full 30-day window both before and after the sale, tracks loss sales per symbol, computes disallowed loss amounts, and blocks violating trades:

$$\forall t : \text{Sell}(t) \wedge \text{Buy}(t') \wedge |t - t'| < 30 \Rightarrow \neg\text{TaxLoss}(t) \tag{24}$$

3.3.2 Almgren-Chriss Risk Constraint

Following the optimal execution framework of **almgren2001optimal**, we constrain market impact relative to volatility:

$$\forall \text{order} : \text{Execute}(\text{order}) \Rightarrow \text{Slippage}(\text{order}) < \lambda \cdot \text{Volatility} \tag{25}$$

This ensures trades remain within the efficient frontier between expected cost and risk.

3.3.3 Additional Constraint Categories

Beyond the foundational constraints above, the production knowledge base includes:

- **Position Limits:** Maximum exposure per asset and aggregate portfolio
- **Capital Allocation:** Constraints on capital deployment across strategies
- **Risk Limits:** Value-at-Risk, maximum drawdown, and volatility thresholds
- **Proprietary Firm Rules:** Compliance with specific prop trading firm requirements (daily loss limits, trailing drawdown, consistency rules)

3.4 Logical Loss Function

3.4.1 Satisfiability Aggregation

$$\text{SAT}(\mathcal{KB}) = \text{p-mean}_{i=1}^{|\mathcal{KB}|}(\phi_i) \tag{26}$$

The evaluation context maintains EMA-smoothed satisfaction scores with per-constraint violation tracking for monitoring and diagnostics.

3.4.2 Logical Loss

$$\mathcal{L}_{\text{logic}} = 1 - \text{SAT}(\mathcal{KB}) \tag{27}$$

4 Multimodal Fusion: Specialized Gated Attention

JANUS integrates multiple data modalities through gated cross-attention (**gatedattention2023**), allowing the system to dynamically weight inputs based on their predictive uncertainty. This is the machine-learning analogue of thalamic attentional gating in biological systems (**thalamus2018attention**).

4.1 Input Modalities

The production system processes four specialized data streams through dedicated fusion engines:

- **Order Book:** Full limit order book depth, bid-ask dynamics, and microstructure features
- **Price:** Multi-timeframe price action including OHLCV and derived indicators

- **Volume:** Volume profile analysis, volume-weighted metrics, and participation rates
- **Sentiment:** News feeds (NewsAPI, CryptoPanic) and social sentiment signals

Additionally, the system integrates supplementary data sources including weather data (via OpenWeatherMap) and space weather/celestial data for correlation analysis with commodity and energy markets.

4.2 Gated Cross-Attention Mechanism

4.2.1 Attention Computation

Following the attention mechanism of **vaswani2017attention**, with support for causal masking and multi-head configuration:

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{28}$$

4.2.2 Gating Mechanism

The gating function suppresses noise and amplifies signal, similar to thalamic regulation (**halassa2014thalamic**):

$$g = \text{sigmoid}(\mathbf{W}_g[\mathbf{v}; \mathbf{t}; \mathbf{s}] + \mathbf{b}_g) \tag{29}$$

Each modality-specific fusion engine applies this gating independently, allowing the system to dynamically suppress noisy order book data during low-liquidity periods while amplifying sentiment signals during news-driven regimes.

5 Decision Engine: Basal Ganglia Pathways

Action selection in JANUS is mediated by a model of the Basal Ganglia, comprising Direct (“Go”) and Indirect (“No-Go”) pathways (**collins2014opponent**; **foster2013hierarchical**). This architecture, known as Opponent Actor Learning (OpAL), creates dynamic risk tolerance that adapts to market conditions.

5.1 Praxeological Motor: Dual Pathways

5.1.1 Direct Pathway (Go Signal)

Encodes the benefits of an action, amplified by dopamine during high-confidence regimes (**dopamine2020reward**):

$$Q_\theta^+(s,a) = \mathbb{E}[\text{Reward} \mid s,a] \tag{30}$$

The implementation includes dopamine sensitivity parameters, learning rates for value updates, decay rates, and threshold-based action release.

5.1.2 Indirect Pathway (No-Go Signal)

Encodes the costs and risks, amplified during uncertainty (**collins2014opponent**):

$$Q_\theta^-(s,a) = \mathbb{E}[\text{Risk} \mid s,a] \tag{31}$$

The implementation provides risk assessment, inhibition generation, and caution scoring.

5.2 Action Selection

The final action is determined by the competition between pathways:

$$\mathbf{a}_t = \operatorname{softmax}(\mathbf{d}_{\text{direct}} - \lambda \cdot \mathbf{d}_{\text{indirect}}) \tag{32}$$

where $\lambda > 0$ is the inhibition weight parameter, and each pathway is computed as:

$$\mathbf{d}_{\text{direct}} = \text{ReLU}(\mathbf{W}_{\text{direct}}\mathbf{h} + \mathbf{b}_{\text{direct}}) \tag{33}$$

$$\mathbf{d}_{\text{indirect}} = \text{ReLU}(\mathbf{W}_{\text{indirect}}\mathbf{h} + \mathbf{b}_{\text{indirect}}) \tag{34}$$

where $\mathbf{h}$ is the fused state representation from the Thalamus.

An actor-critic framework ties both pathways together, with decision confidence scoring determining whether the action is released for execution.

5.3 Brain Wiring Pipeline

In production, decisions traverse a six-stage pipeline that chains brain regions together:

1. **Regime Detection:** Ensemble classifier identifies the current market state
2. **Hypothalamus:** Adjusts position sizing and risk appetite based on regime and portfolio homeostasis

3. **Amygdala:** Evaluates threat signals; may trigger circuit breakers before further processing
4. **Gating:** Thalamic attention gates filter and weight modality signals
5. **Correlation:** Cross-asset correlation tracking across monitored pairs informs diversification
6. **Execution:** Basal ganglia action selection routes to the Execution Service

This pipeline is operationally richer than the abstract dual-pathway model, reflecting the biological reality that action selection involves coordination across multiple brain regions.

5.4 Cerebellar Forward Model

The Cerebellar module simulates market dynamics to predict execution outcomes before committing to a trade.

5.4.1 Market Impact Prediction

Following the Almgren-Chriss framework (**almgren2001optimal**; **markwick2023almgren**; **almgren2024deep**), the model predicts slippage and volatility. To detect predatory environments, JANUS employs the VPIN (Volume-Synchronized Probability of Informed Trading) metric (**easley2011vpin**; **easley2012flow**), which serves as a proxy for “Flow Toxicity”—the probability that the counterparty has superior information. High VPIN levels often precede flash crashes and feed into the Amygdala circuit for threat detection.

$$\hat{p}_{t+1} = f_{\text{cerebellum}}(\mathbf{s}_t, \mathbf{a}_t) \tag{35}$$

5.4.2 Execution Error Correction

The Cerebellum also provides closed-loop error correction for execution quality, implemented through a PID controller:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)\, d\tau + K_d \cdot \frac{de(t)}{dt} \tag{36}$$

where $e(t)$ is the deviation between predicted and realized execution cost. Adaptive correction and feedback loops continuously refine the forward model’s predictions based on observed outcomes.

Part III

Backward Service (Janus Consivius)

Abstract

JANUS Backward represents the "sleep state" of the system, responsible for memory consolidation, schema formation, and learning from accumulated experience. This service implements the Complementary Learning Systems (CLS) theory (**mcclelland1995complement** which posits that intelligent agents require two learning systems: a fast-learning hippocampus for episodic details and a slow-learning neocortex for statistical generalization. This service implements:

- **Three-Timescale Memory Hierarchy** (Hippocampus $\rightarrow$ SWR $\rightarrow$ Neocortex) following CLS architecture (**mcclelland1995complementary**)
- **Sharp-Wave Ripple Simulation** for prioritized experience replay (**buzsaki2015hippocampal**; **schaul2015prioritized**)
- **Schema Formation** via feature-range matching with UMAP-based visualization (**mcinnes2018umap**; **alignedumap2023**)
- **Recall-Gated Consolidation** ensuring only successful patterns are promoted (**frank2006working**)
- **Signal Persistence and Analytics** via PostgreSQL repositories

The Backward service runs on a cold path during off-market hours, performing computationally intensive operations to distill daily experiences into long-term knowledge, effectively replicating the biological process of memory consolidation during sleep (**buzsaki2015hippocampal**).

1 Memory Hierarchy: Three-Timescale Architecture

The three-tier memory architecture directly implements the Complementary Learning Systems theory (**mcclelland1995complementary**), preventing catastrophic forgetting while enabling rapid learning of new market patterns.

1.1 Short-Term Memory (Hippocampus)

The hippocampal buffer stores episodic memories of individual trading events, enabling fast learning without interfering with consolidated knowledge (**mcclelland1995complementary**).

1.1.1 Episodic Buffer

Stores raw experiences during trading, mirroring the role of biological hippocampus in episodic memory formation:

$$\mathcal{D}_{\text{hippo}} = \{(s_t, a_t, r_t, s_{t+1}, \mathbf{c}_t, \mathbf{e}_t)\}_{t=1}^{T} \tag{37}$$

where $\mathbf{c}_t$ contains contextual metadata (volatility, spreads, volume) and $\mathbf{e}_t$ contains emotional tags (fear level, confidence, surprise) that bias consolidation priority.

1.1.2 Pattern Separation

Uses random projections to ensure diverse encoding:

$$\mathbf{h}_t = \tanh(\mathbf{W}_{\text{rand}} \cdot [s_t; a_t; \mathbf{c}_t]) \tag{38}$$

1.1.3 Spatial Mapping

Experiences are organized into a spatial map that preserves topological relationships between market states, analogous to hippocampal place cells. This enables efficient retrieval of contextually similar experiences during replay.

1.2 Medium-Term Consolidation (SWR Simulator)

Consolidation occurs during Sharp-Wave Ripples (SWRs)—high-frequency oscillations that replay compressed sequences of neural activity (**buzsaki2015hippocampal**). JANUS mimics this using Prioritized Experience Replay (**schaul2015prioritized**), modified to incorporate surprise, emotion, and logical violations.

1.2.1 Replay Prioritization

During "sleep" (post-market hours), experiences are replayed in priority order. Biological research shows that replay is biased towards salient and novel events (**kar2023selection**), which JANUS replicates through composite priority scoring. Compute TD-error based priority:

$$p_i = |\delta_i| + \epsilon \tag{39}$$

where $\delta_i = r_i + \gamma \max_{a'} Q(s_{i+1}, a') - Q(s_i, a_i)$ and $\epsilon = 10^{-6}$ ensures numerical stability.

1.2.2 Sampling Probability

$$P(i) = \frac{p_i^\alpha}{\sum_j p_j^\alpha} \tag{40}$$

where $\alpha \in [0, 1]$ controls prioritization strength (default $\alpha = 0.6$).

1.2.3 Importance Sampling Correction

$$w_i = \left(\frac{1}{N \cdot P(i)} \right)^{\beta} \tag{41}$$

where β is annealed from $0.4 \to 1.0$ during training via a configurable increment parameter, fully correcting bias at convergence.

1.2.4 Efficient Sampling via Sum Tree

The replay buffer uses a sum tree data structure for $\mathcal{O}(\log n)$ sampling, enabling efficient prioritized replay even with large buffer sizes.

1.2.5 Sleep-Phase Consolidation

The SWR simulator progresses through biologically inspired phases:

$$\text{Phase} \in \{\text{Awake} \to \text{Light} \to \text{Deep} \to \text{Integration} \to \text{Transition}\} \tag{42}$$

Each phase adjusts replay parameters (compression ratio, replay rate, consolidation threshold), mirroring the empirical finding that different sleep stages serve distinct memory functions.

1.3 Long-Term Memory (Neocortex)

1.3.1 Schema Representation

Schemas represent learned market regime prototypes. The production implementation uses feature-range matching with weighted multi-criteria scoring rather than pure centroid-based clustering:

$$\text{match}(\mathbf{x}, \mathcal{S}_k) = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} w_f \cdot \mathbb{1}[x_f \in \text{range}_f^{(k)}] \tag{43}$$

where $\mathcal{F}$ is the feature set, w_f is the feature weight, and $\text{range}_f^{(k)}$ is the acceptable range for feature f in schema k. This approach is deterministic and interpretable, providing explicit decision boundaries for each regime.

The system supports seven regime schemas (Bull, Bear, Ranging, Crisis, Recovery, Bubble, Deflation), each with a Markov transition matrix for regime prediction:

$$P(\mathcal{S}_{t+1} = j \mid \mathcal{S}_t = i) = \mathbf{T}_{ij} \tag{44}$$

For embedding-based operations (similarity search, schema formation from new experiences), centroid-based representations remain available:

$$\mathbf{z}_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} \mathbf{h}_i \tag{45}$$

1.3.2 Recall-Gated Consolidation

Only update schemas from successfully recalled experiences:

$$\mathbf{z}_k \leftarrow \mathbf{z}_k + \eta \cdot \nVdash[\text{recall\_success}] \cdot (\mathbf{h}_{\text{new}} - \mathbf{z}_k) \tag{46}$$

2 UMAP Visualization: Cognitive Dashboard

To visualize and manage schema structures, JANUS employs Uniform Manifold Approximation and Projection (UMAP) (**mcinnes2018umap**), which preserves both local and global structure better than alternatives like t-SNE.

2.1 AlignedUMAP for Schema Formation

Track how internal representations evolve over time using AlignedUMAP (**alignedumap2023**), which aligns manifolds across different time steps to monitor representational drift. Maintains consistent embeddings across sleep cycles.

2.1.1 Objective Function

The full UMAP loss includes both attraction and repulsion terms:

$$\mathcal{L}_{\text{UMAP}} = \sum_{i \neq j} \left[w_{ij} \log(q_{ij}) + (1 - w_{ij}) \log(1 - q_{ij}) \right] \tag{47}$$

where $q_{ij} = (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$.

Note: In practice, the repulsion term $(1 - w_{ij})$ is approximated via *negative sampling* to achieve $\mathcal{O}(N)$ complexity. For each positive edge, we sample $k = 5$ random negative pairs.

2.2 Parametric UMAP for Real-Time Monitoring

Train a neural network to project new experiences:

$$\mathbf{y}_{\text{new}} = f_\theta(\mathbf{h}_{\text{new}}) \tag{48}$$

Note: The parametric neural projection is a planned enhancement; current monitoring uses standard UMAP projections recomputed during consolidation phases.

3 Persistence Layer

3.1 Primary Storage: PostgreSQL and Redis

The production system uses PostgreSQL for ACID-compliant storage of trading records and Redis for low-latency operational state:

- **Signal Repository:** Persists generated trading signals with full metadata
- **Portfolio Repository:** Tracks positions, P&L, and capital allocation
- **Performance Repository:** Stores performance analytics for backward analysis
- **Redis:** Walk-forward optimizer parameters, kill switch state, hot-reloadable configuration

PostgreSQL provides the ACID guarantees essential for financial records, while Redis enables sub-millisecond reads for time-critical operational state.

3.2 Vector Storage: Qdrant

For schema similarity search, JANUS integrates Qdrant (**qdrant2024**), a high-performance vector similarity search engine. Schemas are stored with L2-normalized centroid vectors for cosine similarity search:

$$\mathcal{N}_k = \arg\max_k \text{cosine}(\mathbf{h}_t, \mathbf{z}_k) \tag{49}$$

The vector database layer complements the relational storage, serving the specific use case of nearest-neighbor retrieval for regime pattern matching.

Part IV

Neuromorphic Architecture

Abstract

This document maps the computational components of Project JANUS to specific brain regions, ensuring biological plausibility and leveraging neuroscience insights for system design. The neuromorphic approach is grounded in cognitive neuroscience (**buzsaki2015hippocampal**; **frank2006working**; **collins2014opponent**) and provides:

- **Modular Design** with clear functional boundaries mirroring brain organization
- **Biological Validation** of architectural decisions based on empirical neuroscience (**mcclelland1995complementary**)
- **Emergent Intelligence** through brain-inspired interactions and allostatic regulation (**sterling2012allostasis**)

1 Neuromorphic Design Philosophy

1.1 Why Brain-Inspired Architecture?

The brain efficiently solves problems similar to trading (**daw2006cortical**), demonstrating capabilities that map directly to trading challenges:

- Pattern recognition under uncertainty (visual cortex and hippocampus (**buzsaki2015hippocam**
- Fast decision-making with delayed rewards (basal ganglia (**collins2014opponent**))
- Continual learning without catastrophic forgetting (complementary learning systems (**mcclelland1995complementary**))
- Multi-timescale memory consolidation (hippocampal-neocortical transfer (**buzsaki2015hippoca**
- Homeostatic regulation under varying conditions (hypothalamic control (**sterling2012allostasis**

1.2 Neuroscience-to-Trading Mapping

This mapping is grounded in empirical neuroscience and cognitive modeling (**frank2006working**; **collins2014opponent**; **foster2013hierarchical**). JANUS implements ten brain regions, each serving a distinct functional role:

Brain Region	Biological Function	Trading Function
Visual Cortex	Pattern recognition	GAF/ViViT chart analysis (**wang2015imaging**; **arnab2021vivit**)
Hippocampus	Episodic memory (**buzsaki2015hippocampal**)	Experience replay buffer (**schaul2015prioritized**)
Prefrontal Cortex	Logic and planning (**frank2006working**)	LTN constraint checking (**badreddine2022logic**)
Basal Ganglia	Action selection (**collins2014opponent**)	Buy/sell/hold decisions
Cerebellum	Motor prediction	Market impact forecasting (**almgren2001optimal**)
Amygdala	Threat detection (**amygdala2019fear**)	Risk circuit breakers
Thalamus	Attentional gating (**halassa2014thalamic**)	Modality fusion and signal routing
Hypothalamus	Homeostatic regulation (**sterling2012allostasis**)	Position sizing and risk appetite
Cortex	Strategic planning	Regime schemas and hierarchical RL
Integration	Inter-region coordination	Service bridges and data pipeline

2 Brain Region Architectures

The following sections detail how each brain region's computational principles are implemented in JANUS.

2.1 Visual Cortex: Pattern Recognition

The visual cortex processes market data as images through the DiffGAF pipeline (Section 1), with submodules for GAF encoding (GASF/GADF), vision model inference (DiffGAF-LSTM and ViViT), data ingestion and buffering, and UMAP-based visualization of learned representations.

2.2 Cortex: Strategic Planning & Long-term Memory

The neocortex implements slow, statistical learning of market schemas (**mcclelland1995complemen**

2.2.1 Trading Implementation

Component: Neocortical Schema Network with Hierarchical RL Manager

- Schema prototypes stored with feature-range matching and confidence scoring

- Seven market regime templates (Bull, Bear, Ranging, Crisis, Recovery, Bubble, Deflation) with Markov transition matrices
- Declarative memory and long-term knowledge base for persistent market insights
- Hierarchical RL manager for multi-level strategic planning
- Slow consolidation during sleep cycles

2.3 Hippocampus: Episodic Memory & Experience Replay

The hippocampus provides fast learning and episodic memory storage, with consolidation via Sharp-Wave Ripples (**buzsaki2015hippocampal**; **kar2023selection**).

2.3.1 Trading Implementation

Component: Episodic Buffer + SWR Replay

- Fixed-size circular buffer storing recent trades
- Sparse encoding via random projections
- Emotional tagging for consolidation priority
- Trade episode and market event recording
- Spatial mapping of experience relationships
- Prioritized replay with sum-tree sampling during training
- Multi-phase sleep consolidation (Awake $\rightarrow$ Light $\rightarrow$ Deep $\rightarrow$ Integration $\rightarrow$ Transition)

2.4 Thalamus: Attentional Gating & Modality Fusion

The Thalamus functions as the “gatekeeper” of perception in JANUS, regulating the flow of visual and numerical data into the decision engine (**thalamus2018attention**; **halassa2014thalamic**).

2.4.1 Trading Implementation

Component: Multi-Head Cross-Attention with Modality-Specific Fusion

- Cross-attention with causal masking and residual connections
- Saliency computation and attentional focus control

- Gating mechanism for signal suppression and amplification
- Specialized fusion engines: order book, price, volume, and sentiment
- External data source integration (news, weather, celestial/space weather)
- Signal routing to downstream brain regions

The Thalamic Reticular Nucleus provides attentional gating, enabling JANUS to focus computational resources on the most informative market data streams. Wilson-Cowan mean-field models (**wilson1972excitatory**; **wilson2024bidirectional**) for oscillatory attention dynamics are planned for a future release.

2.5 Hypothalamus: Homeostatic Regulation

The Hypothalamus implements allostatic regulation (**sterling2012allostasis**), maintaining the system's internal balance across varying market conditions.

2.5.1 Trading Implementation

Component: Adaptive Position Sizing and Risk Appetite Control

- **Position Sizing:** Dynamically adjusts position sizes based on regime, portfolio heat, and recent performance using Kelly Criterion-inspired scaling
- **Homeostasis:** Monitors portfolio-level vital signs (exposure, drawdown, correlation) and applies corrective adjustments to maintain target ranges
- **Energy Management:** Tracks "metabolic" state of the trading system—capital utilization, margin usage, and recovery capacity—and modulates aggression accordingly
- **Risk Appetite:** Integrates regime signals from the ensemble detector with internal portfolio state to produce a single risk appetite scalar that modulates all downstream position sizing

The Hypothalamus sits at the second stage of the brain wiring pipeline, translating raw regime detection into calibrated risk parameters before signals reach the Amygdala and downstream modules.

2.6 Basal Ganglia: Action Selection & Reinforcement Learning

The basal ganglia implements Opponent Actor Learning (OpAL) (**collins2014opponent**), balancing Go (Direct) and No-Go (Indirect) pathways modulated by dopamine (**dopamine2020rewar**
See Section 5 for the complete mathematical formulation.

2.7 Prefrontal Cortex: Logic, Planning & Compliance

The prefrontal cortex provides working memory gating and logical reasoning capabilities (**frank2006working**).

2.7.1 Trading Implementation

Component: Logic Tensor Network + Conscience Module

- Łukasiewicz fuzzy logic with linguistic hedges
- Predicate grounding and constraint satisfaction
- Wash sale rule enforcement with full 30-day window tracking
- Position limits and risk limit constraints
- Proprietary firm rule compliance (daily loss limits, trailing drawdown, consistency)
- Strategic planning and goal management

2.8 Amygdala: Fear, Threat Detection & Circuit Breakers

The amygdala provides rapid threat detection and fear learning (**amygdala2019fear**), with connections to substantia nigra enabling fear extinction (**substantia2016connections**; **monfils2009extinction**).

2.8.1 Trading Implementation

Component: Multi-Layer Threat Detection and Safety System

Anomaly Detection uses multiple complementary methods: Z-score deviation, isolation forest scoring, moving average deviation, percentile outliers, and multivariate scoring. Threats are classified into severity levels (None $\rightarrow$ Low $\rightarrow$ Medium $\rightarrow$ High $\rightarrow$ Critical).

Mahalanobis Distance:

$$D_M(\mathbf{s}_t) = \sqrt{(\mathbf{s}_t - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{s}_t - \boldsymbol{\mu})} \tag{50}$$

where $\boldsymbol{\mu}$ is the historical mean state and $\boldsymbol{\Sigma}$ is the covariance matrix.

Circuit Breaker Condition:

$$\text{Trigger} = \begin{cases} 1 & \text{if } D_M(\mathbf{s}_t) > \tau_{\text{danger}} \\ 0 & \text{otherwise} \end{cases} \tag{51}$$

where τ_{danger} is calibrated to a false-positive rate (e.g., $\tau = 5$ for $p < 0.001$).

Additional Threat Signals:

- Sudden volatility spike: $\sigma_t > 3 \cdot \sigma_{\text{baseline}}$
- Drawdown threshold: cumulative loss $> L_{\max}$
- Liquidity crisis: bid-ask spread $> 10\times$ normal
- Regime shift detection
- Correlation breakdown across monitored pairs
- Black swan event detection
- Flash crash detection via VPIN

VPIN Flow Toxicity: The VPIN calculator (**easley2011vpin**; **easley2012flow**) uses volume bucket aggregation, bulk volume classification, and rolling window computation to produce a toxicity score. High and critical thresholds trigger graduated responses from position reduction to full kill switch activation.

Production Circuit Breakers:

- **Kill Switch:** Dual-layer design with in-process `AtomicBool` for zero-latency local halt and Redis-backed distributed coordination across services
- **Position Freeze:** Prevents new position entry while allowing exits
- **Safe Mode:** Reduces system to minimal-risk operation
- **Cancel All:** Emergency cancellation of all pending orders

Fear Extinction: The fear learning system implements extinction mechanisms (**monfils2009extinction**), allowing the system to "unlearn" fear when threats have passed. This prevents permanent paralysis after traumatic market events while maintaining protective responses for genuine systemic risks.

2.9 Cerebellum: Motor Control & Execution

The cerebellum provides forward models for motor prediction, adapted here for market impact forecasting (**almgren2001optimal**).

2.9.1 Trading Implementation

Component: Forward Model for Market Impact with Error Correction

- **Almgren-Chriss Model:** Full optimal execution with permanent/temporary impact coefficients, risk aversion parameter, and optimal trajectory calculation
- **VPIN Integration:** Volume-Synchronized Probability of Informed Trading for flow toxicity detection
- **Forward Models:** Adverse selection detection, Smith predictor for latency compensation, order latency estimation, and fill probability prediction
- **Error Correction:** PID controller, feedback loops, and adaptive correction for continuous execution quality improvement

Price movement from order execution:

$$\Delta p = f_{\text{cerebellum}}(\text{order\_size}, \text{liquidity}, \text{volatility}) \tag{52}$$

2.10 Integration: Inter-Region Coordination

The Integration module provides the “white matter” connecting brain regions, handling service bridges, cross-region coordination, and the data pipeline that routes information between the Forward, Backward, Execution, Data, and CNS services.

Part V

Rust Implementation

Abstract

This document provides production-ready Rust implementation specifications for Project JANUS. The choice of Rust is strategic, prioritizing memory safety and concurrency (**tradfi2023hft**), with zero-cost abstractions essential for nanosecond-critical high-frequency trading environments. This section includes:

- **ML Framework Strategy** leveraging Candle (**candle2024**) and tch-rs (**mazare2024tch**) for end-to-end Rust-native ML
- **High-Performance Services** with async Tokio runtime (**tokio2024**)
- **Rust-Native Training & Inference Pipeline** — full ML lifecycle in Rust
- **Deployment Architecture** (Docker Compose + Kubernetes)

1 Architectural Overview

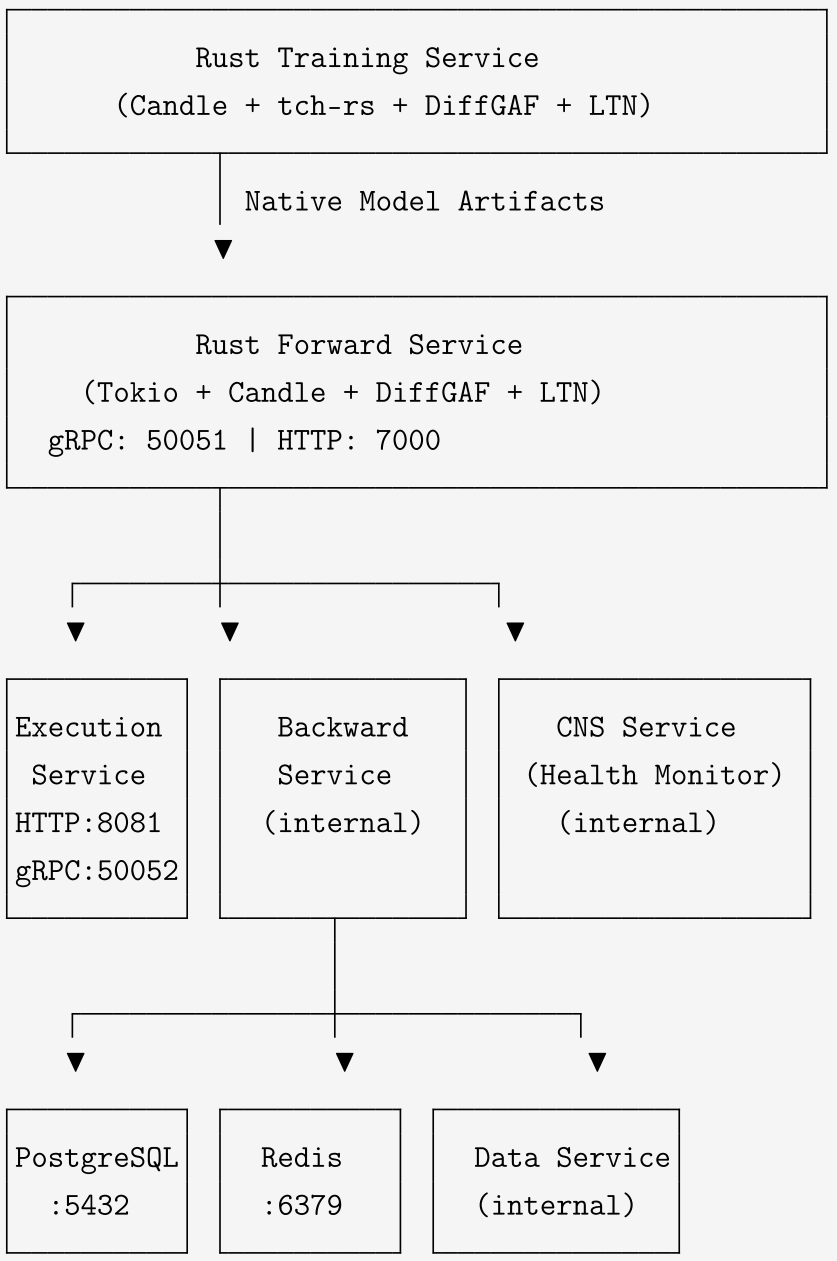
1.1 The Rust-Only Philosophy

The exclusive use of Rust across the entire stack—training, inference, and production services—is justified by requirements in high-frequency trading systems (**tradfi2023hft**):

1. **Performance:** Zero-cost abstractions, no GC pauses—critical for sub-microsecond latency
2. **Safety:** Memory safety without runtime overhead, preventing undefined behavior
3. **Concurrency:** Fearless async/await with Tokio (**tokio2024**) for handling high-frequency websocket feeds
4. **Ecosystem:** End-to-end ML training and inference via Candle (**candle2024**) and tch-rs (**mazare2024tch**)
5. **Unified Stack:** Single language for the entire pipeline eliminates cross-language serialization overhead, deployment complexity, and FFI boundary risks

1.2 Component Diagram

System Architecture (Pure Rust)


Rust Training Service
(Candle + tch-rs + DiffGAF + LTN)
Native Model Artifacts
Rust Forward Service
(Tokio + Candle + DiffGAF + LTN)
gRPC: 50051 | HTTP: 7000
Execution Service HTTP:8081 gRPC:50052
Backward Service (internal)
CNS Service (Health Monitor) (internal)
PostgreSQL :5432
Redis :6379
Data Service (internal)

2 Machine Learning Framework Strategy

2.1 Framework Stack

The following Rust-native frameworks comprise the JANUS ML stack (**mazare2024tch**; **candle2024**):

Framework	Pros	Cons	Use Case
Candle (**candle2024**)	Pure Rust, HuggingFace integration, minimal deps	Younger ecosystem	Primary training & inference
tch-rs (**mazare2024tch**)	Mature PyTorch backend, full GPU support	C++ libtorch dependency	GPU-intensive training & inference
Polars (**polars2024**)	High-speed DataFrames	N/A	Data manipulation

2.2 Rust-Native ML Architecture

The system operates as a **fully Rust-native** ML pipeline with no external language dependencies:

- End-to-end training and inference in Rust via Candle and tch-rs
- Custom differentiable kernels for DiffGAF and LTN operations
- Native model serialization using `safetensors` format
- GPU acceleration via CUDA (tch-rs) and wgpu (Candle)
- Zero cross-language overhead—no FFI bridges, no serialization boundaries

3 Forward Service: Rust Implementation

3.1 Performance Requirements

FPGA acceleration using AMD Alveo U55C cards (**amd2023alveo**; **vemeko2023fpga**; **marino2023mevit**) is planned as future work. Current targets:

- Latency: p99 < 10ms (target: < 1μs with FPGA)
- Throughput: 10,000 req/s
- Memory: < 2GB RSS

3.2 Core Data Structures

The system maintains several key data structures for real-time processing:

Market State Representation:

$$\mathcal{S}_t = (\tau_t, \mathbf{f}_t, \mathcal{O}_t, \mathbf{c}_t) \tag{53}$$

where:

- $\tau_t \in \mathbb{Z}^+$ is the timestamp
- $\mathbf{f}_t \in \mathbb{R}^d$ is the feature vector
- $\mathcal{O}_t = (\mathcal{B}_t, \mathcal{A}_t)$ is the order book with bids $\mathcal{B}_t$ and asks $\mathcal{A}_t$
- $\mathbf{c}_t$ contains contextual metadata (volatility, spreads, volume)

Order Book Structure:

$$\mathcal{B}_t = \{(p_i, q_i) : p_i \in \mathbb{R}^+, q_i \in \mathbb{R}^+\}_{i=1}^{N_{\text{bid}}} \tag{54}$$

$$\mathcal{A}_t = \{(p_j, q_j) : p_j \in \mathbb{R}^+, q_j \in \mathbb{R}^+\}_{j=1}^{N_{\text{ask}}} \tag{55}$$

3.3 GAF Transformation Algorithm

The GAF transformation converts time series to 2D images via the following algorithm (**wang2015imaging**). For training data generation, JANUS employs a Rust-native GPU-accelerated limit order book simulator, inspired by JAX-LOB (**fu2024jaxlob**), that enables parallel simulation of thousands of order books, solving the data scarcity problem inherent in traditional trading systems.

Algorithm: GAF Computation

```
1:  **Input:** Time series $X = \{x_1, \dots, x_W\}$, window size $W$
2:  **Output:** Gramian matrix $\mathbf{G} \in \mathbb{R}^{W \times W}$
3:
4:  $\tilde{X} \leftarrow$ Normalize$(X)$ to $[-1, 1]$
5:  $\phi_i \leftarrow \arccos(\tilde{x}_i)$ for $i = 1, \dots, W$
6:  **for** $i = 1$ to $W$ **do**
7:      **for** $j = 1$ to $W$ **do**
8:          $\mathbf{G}_{ij} \leftarrow \cos(\phi_i + \phi_j)$
9:      **end for**
10: **end for**
11: **return** $\mathbf{G}$ reshaped to $[1, W, W]$ tensor
```

Computational Complexity: $\mathcal{O}(W^2)$ for matrix construction, where W is the window size.

3.4 LTN Constraint Evaluation

Each constraint is represented as a weighted predicate function:

Constraint Structure:

$$\mathcal{C}_k = (P_k, w_k) \tag{56}$$

where $P_k : \mathcal{S} \to [0, 1]$ is a predicate and $w_k \in \mathbb{R}^+$ is the weight.

Evaluation Function:

$$\text{Eval}(\mathcal{C}_k, \mathcal{S}_t) = w_k \cdot P_k(\mathcal{S}_t) \tag{57}$$

T-norm Operations (already defined in Part 2):

$$a \wedge_{\mathcal{L}} b = \max(0, a + b - 1) \quad \text{(Conjunction)} \tag{58}$$

$$a \Rightarrow_{\mathcal{L}} b = \min(1, 1 - a + b) \quad \text{(Implication)} \tag{59}$$

Total Constraint Satisfaction:

$$\mathcal{L}_{\text{constraint}} = 1 - \frac{1}{K} \sum_{k=1}^{K} \text{Eval}(\mathcal{C}_k, \mathcal{S}_t) \tag{60}$$

3.5 Async Service Architecture

The service follows an event-driven architecture with the following characteristics:

Request Processing Pipeline:

1. **Initialization:** Load native Rust model $\mathcal{M}_{\text{ViViT}}$ and LTN engine $\mathcal{E}_{\text{LTN}}$
2. **Connection Handling:** Bind gRPC listener on port 50051 and HTTP on port 7000
3. **Concurrent Processing:** For each incoming request:
   - Spawn asynchronous task with model clone
   - Process request independently (non-blocking)
   - Return prediction and constraint satisfaction scores

Concurrency Model:

$$\text{Throughput} = \frac{N_{\text{workers}} \times 1000}{T_{\text{avg}}} \tag{61}$$

where N_{workers} is the thread pool size and T_{avg} is average processing time in ms.

Performance Characteristics: Non-blocking I/O via async/await, zero-copy model sharing across tasks, and bounded memory through connection limiting.

4 Backward Service: Batch Processing

4.1 Prioritized Experience Replay

The replay buffer maintains experiences with importance-based sampling.

Buffer State:

$$\mathcal{B} = \{(e_i, p_i)\}_{i=1}^{N} \tag{62}$$

where e_i is an experience and $p_i \in \mathbb{R}^+$ is its priority.

Hyperparameters:

- $\alpha \in [0, 1]$: Priority exponent (0 = uniform, 1 = full prioritization)
- $\beta \in [0, 1]$: Importance sampling correction
- C: Buffer capacity

Sampling Algorithm:

```
1: Input: Buffer $\mathcal{B}$, batch size $B$
2: Output: Sampled batch $\{e_{i_1}, \ldots, e_{i_B}\}$
3:
4: Compute probabilities: $P(i) = \frac{p_i^\alpha}{\sum_j p_j^\alpha}$
5: for $k = 1$ to $B$ do
6:     Sample index $i_k \sim$ Categorical$(P)$
7:     Add $e_{i_k}$ to batch
8: end for
9: return batch
```

Importance Weights:

$$w_i = \left(\frac{1}{N \cdot P(i)}\right)^{\beta} \tag{63}$$

These weights correct for the non-uniform sampling distribution.

4.2 Schema Consolidation Algorithm

Schemas are formed by clustering experience embeddings and storing centroids.

Algorithm: Schema Update

```
1: Input: Experiences $\mathcal{E} = \{e_1, \ldots, e_N\}$, number of clusters $K$
2: Output: Updated schema database
3:
4: Extract embeddings: $\mathbf{h}_i =$ Embed$(e_i)$ for $i = 1, \ldots, N$
5: Cluster: $\mathcal{C} = \{C_1, \ldots, C_K\} \leftarrow$ K-means$(\{\mathbf{h}_i\}, K)$
6: for $k = 1$ to $K$ do
7:     Compute centroid: $\mathbf{z}_k = \frac{1}{|C_k|} \sum_{i \in C_k} \mathbf{h}_i$
8:     Compute statistics:
9:         $n_k = |C_k|$
10:        $\bar{r}_k = \frac{1}{|C_k|} \sum_{i \in C_k} r_i$ (average reward)
11:    Upsert schema $k$ with vector $\mathbf{z}_k$ and metadata $(n_k, \bar{r}_k)$
```

```
12: end for
```

Note: In production, the primary schema classification uses deterministic feature-range matching for interpretability. The K-means clustering algorithm above is used for schema *formation* from accumulated experiences during deep consolidation phases.

Schema Metadata: Each schema k stores:

- Centroid vector $\mathbf{z}_k \in \mathbb{R}^d$
- Member count n_k
- Average reward $\bar{r}_k$
- Volatility σ_k (standard deviation of returns)

K-means Objective:

$$\min_{\mathcal{C}} \sum_{k=1}^{K} \sum_{i \in C_k} \|\mathbf{h}_i - \mathbf{z}_k\|^2 \tag{64}$$

5 Execution Service

The Execution Service is a dedicated microservice responsible for multi-exchange order routing, providing a clean separation between decision-making (Forward Service) and order management.

5.1 Multi-Exchange Support

The service supports multiple cryptocurrency exchanges (Kraken, Bybit, Binance) through a unified interface, with exchange-specific adapters handling authentication, rate limiting, and order format translation.

5.2 Service Interface

- HTTP API on port 8081 for order management
- gRPC on port 50052 for low-latency inter-service communication
- Standardized order lifecycle (place, modify, cancel, query)

6 CNS Service: System Health Monitoring

The Central Nervous System (CNS) Service provides system-wide health monitoring and operational safety, analogous to the autonomic nervous system's regulation of vital functions.

6.1 Preflight Validation

Before entering live or paper trading, the CNS Service executes 14+ preflight checks including:

- Database connectivity and schema integrity
- Exchange API authentication and rate limit status
- Model file availability and version consistency
- Memory and CPU resource adequacy
- Kill switch state verification
- Market data feed health

6.2 Runtime Monitoring

During operation, the CNS Service provides:

- **Watchdog Monitoring:** Heartbeat-based service liveness detection
- **Boot Reports:** Comprehensive system state summary on startup
- **Prometheus Metrics:** 15+ custom metrics exposed for Grafana dashboards
- **Alert Integration:** Discord and Slack notifications for critical events
- **Distributed Tracing:** Jaeger integration for cross-service latency analysis

7 Deployment Architecture

7.1 Service Orchestration

The system deploys as five core services plus supporting infrastructure:

Service Topology:

1. **Forward Service:**
   - Ports: gRPC 50051, HTTP 7000
   - Dependencies: Native model artifacts, Redis, PostgreSQL
   - Resource limits: 2GB memory, 2 CPU cores
2. **Backward Service:**

- Internal service (no external ports)
- Dependencies: PostgreSQL, Redis
- Scheduling: Triggered during market close

3. **Execution Service:**
    - Ports: HTTP 8081, gRPC 50052
    - Dependencies: Exchange API credentials, Redis (kill switch)
    - Multi-exchange order routing
4. **Data Service:**
    - Internal service for centralized market data management
    - Dependencies: Exchange WebSocket feeds, QuestDB (time series)
5. **CNS Service:**
    - Internal service for health monitoring and preflight validation
    - Dependencies: All other services (monitoring target)

Infrastructure:

- **PostgreSQL** (port 5432): Primary relational storage for signals, portfolios, and performance
- **Redis** (port 6379): Operational state, kill switch coordination, hot-reloadable config
- **QuestDB** (ports 9000/9009): High-performance time series storage for market data
- **Prometheus** (port 9090): Metrics collection
- **Grafana** (port 3000): Visualization dashboards (5+ dashboards)
- **Alertmanager** (port 9093): Alert routing
- **Jaeger** (port 16686): Distributed tracing

Service Communication:

$$\text{Forward} \xrightarrow[\text{gRPC}]{\text{signals}} \text{Execution} \xrightarrow[\text{HTTP}]{\text{orders}} \text{Exchanges} \tag{65}$$

$$\text{Forward} \xrightarrow[\text{SQL}]{\text{experiences}} \text{PostgreSQL} \xrightarrow[\text{nightly}]{\text{batch}} \text{Backward} \xrightarrow[\text{SQL}]{\text{schemas}} \text{PostgreSQL} \tag{66}$$

Volume Management: Model artifacts are stored on shared read-only volumes; PostgreSQL and QuestDB use persistent volumes with automated backups; Redis data is ephemeral with configurable persistence; experience buffers rotate daily.

Part VI

Advanced Neuroscience Integration

1 Thalamic Oscillations and Attentional Gating

The Thalamus functions as the "gatekeeper" of perception in JANUS, regulating the flow of visual and numerical data into the decision engine. Wilson-Cowan mean-field models (**wilson1972excitatory**; **wilson2024bidirectional**) are planned for simulating the oscillatory dynamics of neural populations to generate "attention masks" that suppress noise and amplify signal.

Currently, the Thalamus module implements attentional gating through multi-head cross-attention with saliency computation and gating mechanisms (Section 2.4). The Thalamic Reticular Nucleus provides attentional gating (**thalamus2018attention**; **halassa2014thala**
enabling JANUS to focus computational resources on the most informative market data streams.

2 Dopaminergic Modulation and Market Regimes

The balance between Direct (Go) and Indirect (No-Go) pathways is modulated by a simulated dopamine signal (**dopamine2020reward**). High dopamine (bull market/high confidence) amplifies the Direct pathway; low dopamine (bear market/uncertainty) amplifies the Indirect pathway. This creates dynamic risk tolerance that adapts to market volatility, implementing allostatic regulation (**sterling2012allostasis**) rather than simple homeostatic feedback.

The Hypothalamus module (Section 2.5) translates regime detection into calibrated risk parameters that modulate dopaminergic signaling throughout the decision pipeline.

3 Fear Extinction and Adaptive Risk Management

The Amygdala module implements fear extinction mechanisms (**monfils2009extinction**; **substantia2016connections**), allowing the system to "unlearn" fear when threats have passed. This prevents the agent from becoming permanently paralyzed by a single traumatic market event (e.g., flash crash), while maintaining protective circuit breakers for genuine systemic risks.

Conclusion

Project JANUS represents a paradigm shift in quantitative trading: from opaque black boxes to transparent, brain-inspired systems that combine the best of deep learning and symbolic reasoning.

Key Innovations

1. **Neuromorphic Architecture:** Ten biologically plausible brain regions with modular design
2. **Neuro-Symbolic Fusion:** LTNs with linguistic hedges bridge neural networks and logical constraints
3. **Multi-Timescale Memory:** Three-tier hierarchy with sleep-phase consolidation mirrors hippocampal-neocortical transfer
4. **Ensemble Regime Detection:** HMM, statistical, and technical methods fused for robust market state identification
5. **Production-Ready Safety:** Dual-layer kill switch, multi-method anomaly detection, and CNS health monitoring
6. **Pure Rust Stack:** End-to-end training and inference in Rust with high-performance, safe, and maintainable five-service architecture

Future Work

- Wilson-Cowan thalamic oscillation models for attention dynamics
- Chronos (**ansari2024chronos**; **ansari2024chronos2**) integration for foundation model time series forecasting (Rust-native)
- BERT text embeddings for structured sentiment analysis via Candle (**candle2024**)
- Parametric UMAP for real-time schema monitoring
- Quantum computing integration for portfolio optimization (**quantum2024portfolio**)
- Continual learning without catastrophic forgetting using CLS principles (**mcclelland1995compl**
- Multi-agent systems for distributed trading using feudal hierarchies (**vezhnevets2017feudal**; **feudal2019multi**)
- Regulatory AI for automated compliance leveraging LTN constraint satisfaction (**badreddine2022logic**)

- Hardware acceleration using FPGAs (**marino2023mevit**; **vemeko2023fpga**; **amd2023alveo**) and neuromorphic chips
- Enhanced market microstructure analysis using VPIN and flow toxicity metrics (**easley2011vpin**; **easley2012flow**)
- Generative diffusion models for synthetic market data generation (**diffusion2024lob**)
- Rust-native GPU-accelerated LOB simulator for scalable training (**fu2024jaxlob**)
- Custom GPU kernels via wgpu for Candle-based training acceleration
- Full Qdrant integration as primary vector database for schema storage

Repository & Contact

GitHub: `https://github.com/nuniesmith/technical_papers`

For implementation code, updates, and discussions, visit the repository.

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