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

Author

Jordan Smith

Date

February 18, 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 (LeCun, Bengio, and Hinton, 2015) toward a new paradigm of Neuro-Symbolic integration (Garcez and Lamb, 2024). 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 (Fama and French, 1993)
- **Quant 3.0 (2010s-Present):** Deep learning hegemony with LSTMs, Transformers (Vaswani et al., 2017), 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 (Kahneman, 2011; Evans, 2008). The system is bifurcated into two distinct but interacting services:

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 (McClelland, McNaughton, and O'Reilly, 1995).

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

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) (Wang and Oates, 2015) with LSTM and Video Vision Transformer (ViViT) (Arnab et al., 2021) temporal modeling
- **Symbolic Reasoning** via Logic Tensor Networks (LTN) (Badreddine et al., 2022) 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 (Collins and Frank, 2014)
- **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` $\rightarrow$ `hypothalamus` $\rightarrow$ `amygdala` $\rightarrow$ `gating` $\rightarrow$ `correlation` $\rightarrow$ `execution`) that routes signals through brain regions in real time. FPGA acceleration (Marino et al., 2023; Vemeko, 2023) 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 Wang and Oates (2015), 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 (Wang and Oates, 2015). Recent research (Author, 2025) 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 (Wang and Oates, 2015).

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 (Arnab et al., 2021). Unlike standard Vision Transformers (Dosovitskiy et al., 2020) 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:

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

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 $z_t \in \{1, \ldots, K\}$ 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 (Badreddine et al., 2022). 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 (Garcez and Lamb, 2024).

3.1 Mathematical Foundation

The central innovation of LTNs (Badreddine et al., 2022) is the ability to make Boolean logic differentiable using Real Logic, specifically Łukasiewicz t-norms (Author, 2024b).

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 Badreddine et al. (2022) and Author (2024b), 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:

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

For inference/evaluation, standard Łukasiewicz logic is used:

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

3.2.2 Disjunction (OR)

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

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:

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

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

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

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

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 (Internal Revenue Service, 2024)—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 Almgren and Chriss (2001), 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 (Author, 2023c), 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 (Author, 2018).

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, with Chronos foundation model forecasting (Ansari et al., 2024b; Ansari et al., 2024a)
- **Volume:** Volume profile analysis, volume-weighted metrics, and participation rates
- **Sentiment:** BERT/FinBERT embeddings (Hugging Face, 2024) from news feeds (NewsAPI, CryptoPanic) and social sentiment signals, with Qdrant-backed similarity search for regime-aware aggregation

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 Vaswani et al. (2017), 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 (Halassa and Kastner, 2017):

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

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 (Collins and Frank, 2014; Foster, Morris, and Dayan, 2013). 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 (Author, 2020):

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

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 (Collins and Frank, 2014):

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

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

5.2 Action Selection

The final action is determined by the competition between pathways:

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

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 (Almgren and Chriss, 2001; Markwick, 2023; Author, 2024c), the model predicts slippage and volatility. To detect predatory environments, JANUS employs the VPIN (Volume-Synchronized Probability of Informed Trading) metric (Easley, Lopez de Prado, and O'Hara, 2011; Easley, De Prado, and O'Hara, 2012), 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 (McClelland, McNaughton, and O'Reilly, 1995), 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 → SWR → Neocortex) following CLS architecture (McClelland, McNaughton, and O'Reilly, 1995)
- **Sharp-Wave Ripple Simulation** for prioritized experience replay (Buzsáki, 2015; Schaul et al., 2015)
- **Schema Formation** via feature-range matching with UMAP-based visualization (McInnes, Healy, and Melville, 2018; Author, 2023a)
- **Recall-Gated Consolidation** ensuring only successful patterns are promoted (Frank, Loughry, and O'Reilly, 2006)
- **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 (Buzsáki, 2015).

1 Memory Hierarchy: Three-Timescale Architecture

The three-tier memory architecture directly implements the Complementary Learning Systems theory (McClelland, McNaughton, and O'Reilly, 1995), 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 (McClelland, McNaughton, and O'Reilly, 1995).

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 (Buzsáki, 2015). JANUS mimics this using Prioritized Experience Replay (Schaul et al., 2015), 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 (Kar et al., 2023), 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 $0.4 \rightarrow 1.0$ 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:

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

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:

$$\mathbf{z}_k \leftarrow \mathbf{z}_k + \eta \cdot \mathbb{1}[\text{recall\_success}] \cdot (\mathbf{h}_{\text{new}} - \mathbf{z}_k) \tag{46}$$

2 UMAP Visualization: Cognitive Dashboard

To visualize and manage schema structures, JANUS employs Uniform Manifold Approximation and Projection (UMAP) (McInnes, Healy, and Melville, 2018), 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 (Author, 2023a), 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:

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

The parametric UMAP implementation uses a multi-layer encoder network trained on the fuzzy simplicial set graph, with configurable distance metrics (Euclidean, Cosine,

Manhattan), smooth k-nearest-neighbor bandwidth estimation, and a Qdrant bridge for persistent storage of projected embeddings. A drift detection module computes embedding-space displacement between reference and live data, classifying severity (Low/Moderate/High/Critical) for regime monitoring. Regime cluster analysis provides per-cluster statistics (centroid, intra-cluster distances, standard deviation) and inter-cluster separation metrics.

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 (Qdrant, 2024), 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 (Buzsáki, 2015; Frank, Loughry, and O'Reilly, 2006; Collins and Frank, 2014) and provides:

- **Modular Design** with clear functional boundaries mirroring brain organization
- **Biological Validation** of architectural decisions based on empirical neuroscience (McClelland, McNaughton, and O'Reilly, 1995)
- **Emergent Intelligence** through brain-inspired interactions and allostatic regulation (Sterling, 2012)

1 Neuromorphic Design Philosophy

1.1 Why Brain-Inspired Architecture?

The brain efficiently solves problems similar to trading (Daw et al., 2006), demonstrating capabilities that map directly to trading challenges:

- Pattern recognition under uncertainty (visual cortex and hippocampus (Buzsáki, 2015))
- Fast decision-making with delayed rewards (basal ganglia (Collins and Frank, 2014))
- Continual learning without catastrophic forgetting (complementary learning systems (McClelland, McNaughton, and O'Reilly, 1995))
- Multi-timescale memory consolidation (hippocampus to neocortex (Buzsáki, 2015))
- Homeostatic regulation under varying conditions (hypothalamic control (Sterling, 2012))

1.2 Neuroscience-to-Trading Mapping

This mapping is grounded in empirical neuroscience and cognitive modeling (Frank, Loughry, and O'Reilly, 2006; Collins and Frank, 2014; Foster, Morris, and Dayan, 2013). 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 (Wang and Oates, 2015; Arnab et al., 2021)
Hippocampus	Episodic memory (Buzsáki, 2015)	Experience replay buffer (Schaul et al., 2015)
Prefrontal Cortex	Logic and planning (Frank, Loughry, and O'Reilly, 2006)	LTN constraint checking (Badreddine et al., 2022)
Basal Ganglia	Action selection (Collins and Frank, 2014)	Buy/sell/hold decisions
Cerebellum	Motor prediction	Market impact forecasting (Almgren and Chriss, 2001)
Amygdala	Threat detection (Author, 2019)	Risk circuit breakers
Thalamus	Attentional gating (Halassa and Kastner, 2017)	Modality fusion and signal routing
Hypothalamus	Homeostatic regulation (Sterling, 2012)	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, parametric UMAP (McInnes, Healy, and Melville, 2018) with neural encoder projection, drift detection (Low/Moderate/High/Critical severity classification), and Qdrant-backed regime cluster persistence for real-time representation monitoring.

2.2 Cortex: Strategic Planning & Long-term Memory

The neocortex implements slow, statistical learning of market schemas (McClelland, McNaughton, and O'Reilly, 1995).

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 (Buzsáki, 2015; Kar et al., 2023).

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)
- **Diffusion-Based Synthetic Data:** Regime-conditional DDPM (Author, 2024e) generates synthetic market sequences for training data augmentation, with configurable noise schedules (linear, cosine, quadratic), Min-SNR loss weighting, EMA model averaging, and automated quality assessment comparing synthetic vs. real feature distributions and autocorrelation structure

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 (Author, 2018; Halassa and Kastner, 2017).

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
- **Chronos Time Series Forecasting:** ONNX-based inference pipeline (Ansari et al., 2024b; Ansari et al., 2024a) with quantile-based tokenization, configurable presets (T5-Tiny/Small/Base), and confidence-interval forecast output
- **BERT Sentiment Analysis:** FinBERT/DistilBERT sentiment embeddings (Hugging Face, 2024) running natively in Rust via Candle, producing `[CLS]` embeddings stored in Qdrant for regime-aware sentiment aggregation

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 (Wilson and Cowan, 1972; Author, 2024a) implement oscillatory attention dynamics, modeling coupled excitatory-inhibitory neural populations via ODEs with Hilbert-transform-based amplitude/phase estimation, bifurcation analysis, neuromodulator-driven regime switching, and fixed-point stability classification.

2.5 Hypothalamus: Homeostatic Regulation

The Hypothalamus implements allostatic regulation (Sterling, 2012), 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) (Collins and Frank, 2014), balancing Go (Direct) and No-Go (Indirect) pathways modulated by dopamine (Author, 2020). 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 (Sterling, 2012) 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. 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 (Frank, Loughry, and O'Reilly, 2006).

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 with goal decomposition, subgoal generation, plan synthesis, and contingency planning
- **Quantum-Inspired Portfolio Optimization** (Author, 2024d): QAOA (Quantum Approximate Optimization Algorithm) simulator for combinatorial asset selection via QUBO formulation, VQE (Variational Quantum Eigensolver) for continuous weight optimization using parameterized circuits, and simulated quantum annealing for escaping local minima in non-convex portfolio landscapes — complementing classical Mean-Variance, Risk Parity, and Black-Litterman optimizers

2.8 Amygdala: Fear, Threat Detection & Circuit Breakers

The amygdala provides rapid threat detection and fear learning (Author, 2019), with connections to substantia nigra enabling fear extinction (Author, 2016; Monfils et al., 2009).

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 (Easley, Lopez de Prado, and O'Hara, 2011; Easley, De Prado, and O'Hara, 2012) 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 (Monfils et al., 2009; Author, 2016), 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.

2.9 Cerebellum: Motor Control & Execution

The cerebellum provides forward models for motor prediction, adapted here for market impact forecasting (Almgren and Chriss, 2001).

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
- **LOB Simulator** (Fu, Pakkanen, and Cont, 2024): Full limit order book simulator with price-time priority matching engine, support for Limit, Market, IOC, FOK, Post-Only, Iceberg, Stop, and Stop-Limit order types, self-trade prevention, maker/taker fee computation, L2/L3 snapshots, VWAP and weighted mid-price calculations, queue position estimation, configurable tick/lot sizing, and Almgren-Chriss market impact modeling — enabling parallel synthetic training data generation for reinforcement learning

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 (Author, 2023b), with zero-cost abstractions essential for nanosecond-critical high-frequency trading environments. This section includes:

- **ML Framework Strategy** leveraging Candle (Hugging Face, 2024) for end-to-end Rust-native ML
- **High-Performance Services** with async Tokio runtime (Tokio Contributors, 2024)
- **Rust-Native Training & Inference Pipeline** — full ML lifecycle in Rust
- **Neuromorphic Module** — complete brain-region implementations in pure Rust
- **Trading Strategies** — nine regime-aware strategies with gating and affinity
- **Deployment Architecture** (Docker Compose + Kubernetes)

1 Architectural Overview

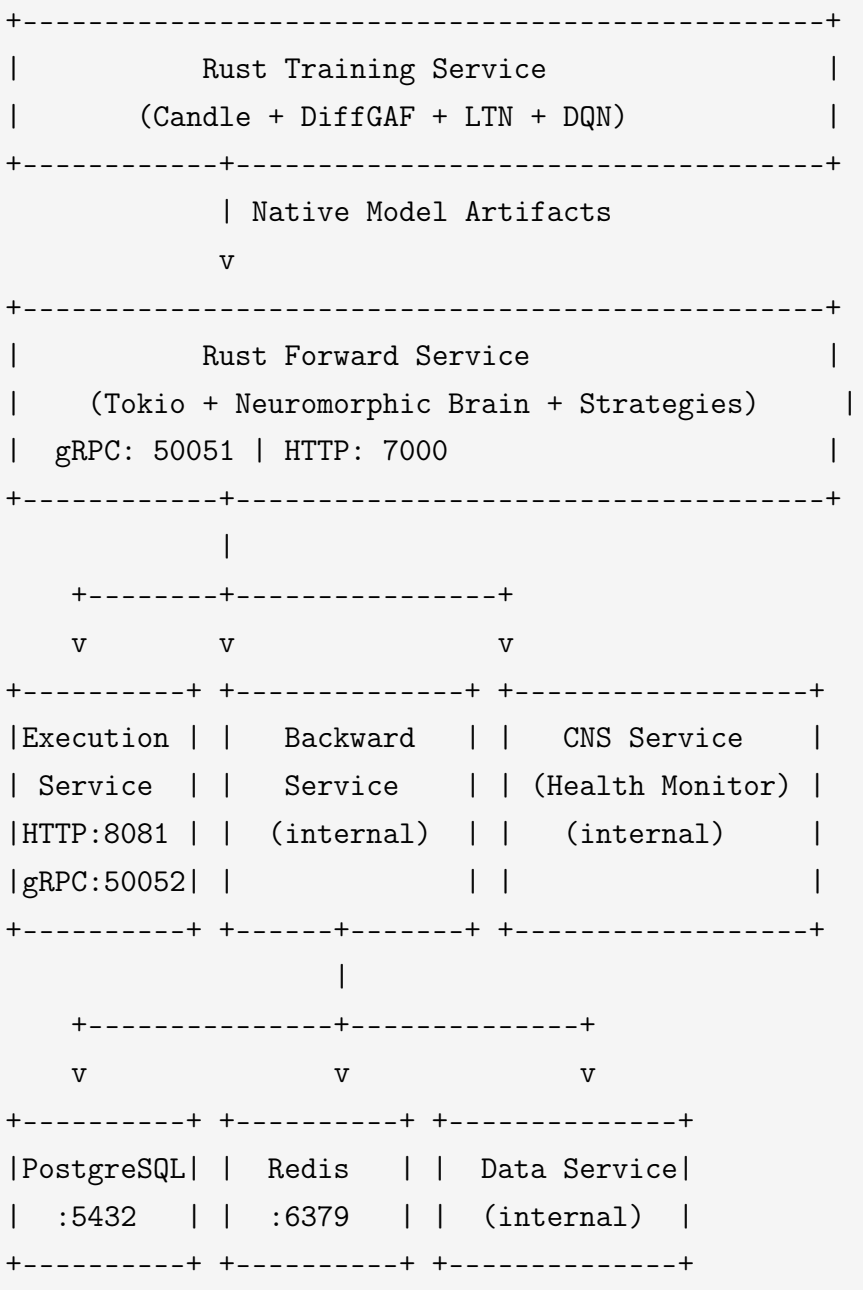
1.1 The Rust-Only Philosophy

The exclusive use of Rust across the entire stack—training, inference, and production services—is justified by requirements in high-frequency trading systems (Author, 2023b):

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 (Tokio Contributors, 2024) for handling high-frequency websocket feeds
4. **Ecosystem:** End-to-end ML training and inference via Candle (Hugging Face, 2024)
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)



2 Machine Learning Framework Strategy

2.1 Framework Stack

The following Rust-native frameworks comprise the JANUS ML stack (Hugging Face, 2024):

Framework	Pros	Cons	Use Case
Candle (Hugging Face, 2024)	Pure Rust, HuggingFace integration, minimal deps, autograd	Younger ecosystem	Primary training & inference (DiffGAF, ViViT, LTN, DQN)
ndarray	Zero-dependency numerical arrays, mature ecosystem	No autograd	Traditional fuzzy logic, GAF encoding, feature engineering
Polars (Polars Contributors, 2024)	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 with autograd support
- Custom differentiable kernels for DiffGAF and LTN operations
- Double DQN with online/target networks for reinforcement learning
- Native model serialization using `safetensors` format
- GPU acceleration via wgpu (Candle) and CUDA (optional)
- Zero cross-language overhead—no FFI bridges, no serialization boundaries
- Training infrastructure: AdamW/SGD optimizers, warmup+cosine LR scheduling, prioritized replay buffers

3 Forward Service: Rust Implementation

3.1 Performance Requirements

FPGA acceleration using AMD Alveo U55C cards (AMD, 2023; Vemeko, 2023; Marino et al., 2023) 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 (Wang and Oates, 2015). For training data generation, JANUS employs a Rust-native GPU-accelerated limit order book simulator, inspired by JAX-LOB (Fu, Pakkanen, and Cont, 2024), that enables parallel simulation of thousands of order books, solving the data scarcity problem inherent in traditional trading systems.

Algorithm 1 GAF Computation

Input: Time series $X = \{x_1, \ldots, x_W\}$, window size W
Output: Gramian matrix $\mathbf{G} \in \mathbb{R}^{W \times W}$

$\tilde{X} \leftarrow$ Normalize(X) to $[-1, 1]$
$\phi_i \leftarrow \arccos(\tilde{x}_i)$ for $i = 1, \ldots, W$
for $i = 1$ to W **do**
    for $j = 1$ to W **do**
        $\mathbf{G}_{ij} \leftarrow \cos(\phi_i + \phi_j)$
    end for
end for
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

Algorithm 2 Prioritized Experience Sampling

Input: Buffer $\mathcal{B}$, batch size B
Output: Sampled batch $\{e_{i_1}, \ldots, e_{i_B}\}$

Compute probabilities: $P(i) = \frac{p_i^\alpha}{\sum_j p_j^\alpha}$
for $k = 1$ to B **do**
    Sample index $i_k \sim \text{Categorical}(P)$
    Add e_{i_k} to batch
end for
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.

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

Algorithm 3 Schema Update

Input: Experiences $\mathcal{E} = \{e_1, \ldots, e_N\}$, number of clusters K
Output: Updated schema database

Extract embeddings: $\mathbf{h}_i = \text{Embed}(e_i)$ for $i = 1, \ldots, N$
Cluster: $\mathcal{C} = \{C_1, \ldots, C_K\} \leftarrow \text{K-means}(\{\mathbf{h}_i\}, K)$
for $k = 1$ to K **do**
    Compute centroid: $\mathbf{z}_k = \frac{1}{|C_k|} \sum_{i \in C_k} \mathbf{h}_i$
    Compute statistics:
        $n_k = |C_k|$
        $\bar{r}_k = \frac{1}{|C_k|} \sum_{i \in C_k} r_i$ (average reward)
    Upsert schema k with vector $\mathbf{z}_k$ and metadata $(n_k, \bar{r}_k)$
end for

- 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 through a unified adapter interface (`exchanges` crate), with exchange-specific adapters handling authentication, rate limiting, and order format translation:

- **Kraken:** Full REST and WebSocket adapter with order normalization (primary exchange)
- **Bybit:** Dedicated client crate (`bybit-client`) with unified order types
- **Coinbase:** REST adapter with authentication and rate limiting
- **OKX:** REST adapter with order format translation
- **Binance:** Legacy connector maintained for backward compatibility

- **Kucoin:** Legacy connector maintained for backward compatibility

Each adapter implements a common `ExchangeAdapter` trait, enabling transparent exchange selection at runtime. The `normalizer` module translates exchange-specific order responses into canonical JANUS types.

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)
- Circuit breaker per exchange with configurable failure thresholds and recovery timeouts
- Rate limiter with token bucket and sliding window algorithms

5.3 Execution Algorithms

The system provides production execution algorithms:

- **TWAP:** Time-Weighted Average Price execution with configurable slice intervals
- **VWAP:** Volume-Weighted Average Price execution with volume profile estimation
- **Iceberg:** Hidden large orders by exposing small visible tip orders to minimize market impact
- **Execution Analytics:** Per-venue latency tracking, fill rate monitoring, and slippage analysis

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 a five-phase preflight sequence, each with configurable criticality levels (Critical, Required, Optional):

1. **Infrastructure Phase:** Database connectivity, Redis availability, Qdrant health, shared memory paths
2. **Sensory Phase:** Exchange API authentication, WebSocket feed health, market data freshness
3. **Regulatory Phase:** Kill switch state verification, compliance rule loading, wash sale detector initialization
4. **Strategy Phase:** Model file availability, version consistency, strategy affinity configuration
5. **Executive Phase:** Memory and CPU resource adequacy, neuromorphic brain region initialization

Critical failures abort the boot sequence; required failures block trading but allow monitoring; optional failures are logged but permit full operation. The `PreFlightRunner` supports both sequential and parallel-within-phase execution modes, producing a comprehensive `BootReport` with Discord-formatted notifications.

6.2 Runtime Monitoring

During operation, the CNS Service provides:

- **Watchdog Monitoring:** Heartbeat-based component liveness detection with configurable degraded/dead thresholds, per-component criticality levels (Critical, Important, NonEssential), and automatic kill switch triggering when critical components die
- **Boot Reports:** Comprehensive system state summary with pass/fail/skip counts per phase
- **Prometheus Metrics:** 15+ custom metrics exposed for Grafana dashboards (6+ pre-built dashboards)
- **Alert Integration:** Slack, PagerDuty, and generic webhook notifications with severity-based routing
- **Distributed Tracing:** Jaeger integration for cross-service latency analysis
- **Circuit Breakers:** Per-component circuit breakers with Closed $\rightarrow$ Open $\rightarrow$ HalfOpen state machine, configurable failure windows, and recovery timeouts
- **Reflex Actions:** Automated responses including component restart, throttling, graceful shutdown, and safe command execution with allowlist validation

- **Neuromorphic Brain Coordinator:** Topological initialization ordering of all 10 brain regions based on dependency graphs, per-region health scoring, and global brain activation/deactivation

7 Trading Strategies

The `strategies` crate implements nine regime-aware trading strategies, each designed for specific market conditions as identified by the ensemble regime detector:

7.1 Trend-Following Strategies

- **EMA Flip:** 8/21 EMA crossover with ATR-based stops, trading pullbacks to the fast EMA in the trend direction
- **EMA Ribbon Scalper:** 8/13/21 EMA ribbon with volume confirmation for higher-quality pullback entries
- **Trend Pullback:** Fibonacci retracement entries within established trends, confirmed by RSI divergence and candlestick patterns (pin bars, engulfing)
- **Momentum Surge:** Detects sudden price surges with volume spikes, entering on the first pullback within the surge
- **Multi-Timeframe Trend:** EMA 50/200 crossover with ADX strength and higher-timeframe alignment

7.2 Mean-Reversion Strategies

- **Mean Reversion:** Bollinger Bands with RSI confirmation and ATR-based stops
- **Bollinger Squeeze Breakout:** Detects low-volatility squeeze periods and generates breakout signals when price escapes the bands
- **VWAP Scalper:** Mean reversion scalping around the Volume-Weighted Average Price with standard deviation bands
- **Opening Range Breakout:** Trades breakouts above or below the first N candles of a session with volume confirmation

7.3 Strategy Gating and Affinity

The `StrategyGate` module controls which strategies execute based on:

- **Regime Compatibility:** Each asset can define preferred strategies per regime (e.g., Trending → EMA Flip, MeanReverting → Bollinger Squeeze)
- **Affinity Scoring:** Strategy-asset affinity tracker with performance-weighted scoring
- **Allowlists/Denylists:** Per-asset strategy filtering via TOML configuration
- **Untested Strategy Policy:** Configurable flag to allow or block strategies without historical performance data

8 Neuromorphic Module: Brain-Region Implementations

The `neuromorphic` crate is the largest module in the JANUS codebase, implementing all ten brain regions as production Rust code. Each region is a self-contained submodule with its own configuration, state management, and comprehensive test suite.

Neuromorphic Crate Structure

```
neuromorphic/
+-- visual_cortex/    # GAF, ViViT, preprocessing, UMAP viz
+-- cortex/           # Schemas, knowledge base, planning
|   +-- memory/       # 7 regime schemas + Markov transitions
+-- hippocampus/      # Episodic buffer, SWR, consolidation
|   +-- swr/          # Ripple detection, compressed replay
+-- thalamus/         # Attention, fusion, gating, routing
|   +-- fusion/       # Orderbook, price, volume, sentiment
+-- hypothalamus/     # Position sizing, homeostasis, energy
|   +-- position_sizing/  # Drawdown scaling, Kelly criterion
+-- basal_ganglia/    # Direct/Indirect pathways, OpAL
|   +-- praxeological/  # Go/No-Go signals, confidence
+-- amygdala/         # Threat detection, VPIN, kill switch
|   +-- vpin/         # Calculator, flash crash, toxicity
+-- cerebellum/       # Forward models, error correction
|   +-- error_correction/  # PID controller, feedback loops
|   +-- forward_models/    # Smith predictor, fill prob
+-- prefrontal/       # LTN, fuzzy logic, conscience, goals
|   +-- ltn/          # Hedges: very, somewhat, extremely
+-- integration/      # Coordinator, message bus, bridges
|   +-- engine/       # Cognitive core, orchestrator
+-- distributed/      # Multi-node coordination
```

8.1 Key Implementation Details

Basal Ganglia — Opponent Actor Learning (OpAL): The `praxeological` module implements the full Go/No-Go architecture with dopamine-modulated action selection. The `GoSignal` evaluates action value against adaptive thresholds with dopamine sensitivity, urgency boosting, and facilitation bias. The `NoGoSignal` evaluates 12 inhibition reasons (risk threshold, position limit, loss limit, high volatility, low liquidity, cooling off, correlation risk, drawdown protection, time restriction, external halt, learned pattern, and custom) with learned inhibition patterns and adaptive thresholds. The actor-critic framework includes Generalized Advantage Estimation (GAE) and TD(λ) learning for stable policy updates, with winner-take-all selection and habit caching for frequently-encountered states.

Cerebellum — Forward Models and Error Correction: The `almgren_chriss` module implements optimal execution trajectories with permanent/temporary impact coefficients. The `pid_controller` provides a full PID implementation with anti-windup,

dead band, derivative filtering, cascaded PID, and Ziegler-Nichols auto-tuning. The `forward_models` submodule includes adverse selection detection, Smith predictor for latency compensation, order latency estimation, and fill probability prediction.

Amygdala — VPIN and Kill Switch: The `vpin` module implements volume-synchronized probability of informed trading with bulk volume classification, rolling bucket computation, and configurable high/critical thresholds. The `kill_switch` provides a four-scope design (per-strategy, per-instrument, per-service, global) with emergency actions (cancel all orders, close all positions, disable trading, send alerts). The fear network integrates reinforcement learning (FNI-RL) to adapt threat responses based on historical outcomes.

Cortex — Schema Formation: The `schemas` module implements all seven regime schemas (Bull, Bear, Ranging, Crisis, Recovery, Bubble, Deflation) with 10 market features (trailing return, realised volatility, average correlation, max drawdown, vol rate of change, momentum signal, mean reversion signal, credit spread, yield curve slope, relative volume), weighted feature-range matching, a full Markov transition matrix with stationary distribution computation, and EMA-smoothed confidence tracking. The cortex also implements a hierarchical RL manager with feudal goal-setting and subgoal generation for multi-level strategic planning.

Prefrontal — Fuzzy Logic with Linguistic Hedges: The `ltn` submodule implements the complete Łukasiewicz fuzzy logic system with five linguistic hedges: `very` (x^2, concentration), `somewhat` ($\sqrt{x}$, dilation), `slightly` ($\sqrt{x} - x$), `extremely` (x^3), and `more_or_less` ($\sqrt{x}$, synonym for dilation). A full expression evaluator supports nested hedge application.

Thalamus — Multimodal Fusion: Four specialized fusion engines process distinct data streams: `orderbook_fusion` (multi-venue book consolidation with weighted mid-price, imbalance detection, and EMA smoothing), `price_fusion` (multi-timeframe price action), `volume_fusion` (volume profile analysis), and `sentiment_fusion` (news and social sentiment signals). The `gating` submodule implements Wilson-Cowan oscillatory dynamics (Wilson and Cowan, 1972; Author, 2024a) with Hilbert-transform-based amplitude/phase estimation, bifurcation analysis, and neuromodulator-driven regime switching, alongside sensory gates with relevance scoring and threshold-based filtering. The `sources` submodule integrates Chronos time series forecasting (Ansari et al., 2024b; Ansari et al., 2024a) via ONNX inference with quantile tokenization and confidence intervals, BERT/FinBERT sentiment embeddings (Hugging Face, 2024) running natively via Candle with Qdrant-backed storage, plus news feeds, weather data, and celestial/space weather for commodity correlation analysis.

Hippocampus — Sharp-Wave Ripple Simulation: The `swr` submodule implements 11 ripple types (large profit, large loss, novel pattern, market anomaly, strategy breakthrough, risk event, regime change, correlation breakdown, volatility spike, liquid-

ity event, periodic) with priority-weighted detection. The `compressed_replay` module provides compressed experience sequences, and `consolidation_sync` coordinates the transfer from hippocampal to neocortical storage. The hippocampus also houses a feudal RL worker agent with a skill library and tactical policy for executing subgoals issued by the cortex's strategic planner. A regime-conditional DDPM diffusion model (Author, 2024e) generates synthetic market data for training data augmentation, with configurable noise schedules (linear, cosine, quadratic), Min-SNR loss weighting, and automated quality assessment comparing synthetic vs. real feature distributions.

Distributed Training Infrastructure: The `distributed` submodule provides multi-GPU and multi-node training coordination with AllReduce, Parameter Server, and Ring AllReduce gradient synchronization strategies. Distributed data loading supports multiple sharding strategies (Contiguous, RoundRobin, Random, Stratified). The infrastructure includes NCCL GPU-to-GPU communication, gRPC-based inter-node coordination, and distributed checkpointing with cloud storage backends.

GPU Compute Infrastructure: The `gpu` submodule provides custom wgpu compute kernels for hardware-accelerated numerical operations: MatMul, Softmax, LayerNorm, Attention, GELU, Reduce, Embedding Lookup, and Pairwise Distance. The module includes a device manager, buffer pool, kernel registry, and a Candle tensor bridge for CPU↔GPU data transfer. All GPU functionality is gated behind a `gpu` feature flag, with graceful CPU fallback when unavailable.

Prefrontal — Quantum-Inspired Portfolio Optimization: The `planning` submodule includes quantum-inspired portfolio optimizers (Author, 2024d): a QAOA (Quantum Approximate Optimization Algorithm) simulator for combinatorial asset selection via QUBO formulation, a VQE (Variational Quantum Eigensolver) portfolio optimizer for continuous weight optimization using parameterized circuits, and simulated quantum annealing for escaping local minima in non-convex portfolio landscapes. These complement classical Mean-Variance, Risk Parity, and Black-Litterman optimizers.

Cerebellum — LOB Simulator: The `lob_simulator` module provides a full limit order book simulator (Fu, Pakkanen, and Cont, 2024) with price-time priority matching, support for Limit, Market, IOC, FOK, Post-Only, Iceberg, Stop, and Stop-Limit order types, self-trade prevention, maker/taker fee computation, L2/L3 snapshots, VWAP and weighted mid-price calculations, Almgren-Chriss market impact estimation, order event history, and configurable tick/lot sizing. The simulator enables parallel training data generation, solving the data scarcity problem for reinforcement learning.

8.2 Supporting Crates

Beyond the core neuromorphic and service crates, the workspace includes:

- **vision:** DiffGAF (learnable normalization, polar encoding, Gramian layers) and

ViViT (factorized encoder with spatial/temporal attention) connected via a fully wired `VisionPipeline` that chains the two end-to-end as a single differentiable Candle module with automatic frame splitting, dual-GAF mode, and optional projection head. Also includes backtest simulation, portfolio optimization (Black-Litterman, Risk Parity, Mean-Variance), production monitoring with circuit breakers, live inference with latency profiling, and parametric UMAP (McInnes, Healy, and Melville, 2018) with drift detection and Qdrant-backed cluster persistence

- **training:** End-to-end training loop with AdamW/SGD optimizers, learning rate schedulers (warmup+cosine, step, exponential), prioritized experience replay with SWR sampling, gradient clipping, checkpointing with metadata, early stopping, and pluggable callbacks
- **ml:** LSTM and MLP models via Candle, Double DQN with online/target networks and soft updates, feature engineering (price, volume, technical indicators, normalizers), and dataset management
- **logic:** Dual-mode LTN system — non-differentiable (ndarray) for inference-time constraint checking and differentiable (Candle) for gradient-based training — with three t-norm families (Łukasiewicz, Product, Gödel), learnable/threshold/similarity predicates, and comprehensive rule composition (ForAll, Exists, Implies, Iff, AndN, OrN)
- **ltn:** Domain-specific neuro-symbolic engine with 10 market axioms encoded as logical rules (e.g., Trending + Positive Divergence $\rightarrow$ Long, Low Confidence $\rightarrow$ Neutral), integrating DSP features through fuzzy predicates with a hybrid supervised-semantic loss function
- **regime:** Ensemble regime detection combining HMM (Gaussian emissions, Baum-Welch, online parameter updates), indicator-based (ADX, Bollinger Bands, ATR, EMA, RSI), and ensemble fusion with agreement tracking and strategy routing
- **dsp:** Digital signal processing with FRAMA (Fractal Adaptive Moving Average) and Sevcik fractal dimension estimation producing an 8D feature vector (divergence, alpha, fractal dimension, Hurst exponent, regime, sign, deviation, confidence) at sub-microsecond latency per tick
- **lob:** Standalone limit order book crate with matching engine, fill probability models, L2 replay, latency simulation, and Almgren-Chriss market impact estimation
- **compliance:** Dedicated `WashSaleDetector` with full 30-day lookback/lookforward window, partial wash sale handling, cost basis tracking, and position-aware de-

tection; plus `ComplianceSheriff` enforcing proprietary firm trading rules (daily loss limits, maximum loss thresholds, mandatory stop-losses)

- **optimizer:** Hyperparameter optimization with configurable samplers, constraints, backtesting integration, and result publishing
- **data-quality:** Market data validation and anomaly detection
- **gap-detection:** Multi-layer time series gap detection (sequence ID, heartbeat, statistical, volume-aware) with PostgreSQL persistence
- **indicators:** Technical indicator library (ADX, ATR, Bollinger Bands, EMA, RSI, MACD) with incremental O(1) per-tick computation
- **rate-limiter:** Token bucket and sliding window rate limiters with async circuit breaker, exchange-specific algorithms
- **questdb-writer:** High-performance time series ingestion to QuestDB via ILP, batched writes targeting >100K inserts/sec
- **memory:** Three-tier memory hierarchy with production Qdrant vector database client (circuit breaker, exponential backoff retries, TLS, mock fallback), predefined collections for market regimes, episodic memory, sentiment embeddings, and schema prototypes, plus prioritized experience replay buffer

9 Deployment Architecture

9.1 Service Orchestration

The system deploys as eight 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)
6. **API Service:**
    - HTTP API gateway for external access
    - Dependencies: Forward Service, Backward Service
7. **Registry Service:**
    - Internal service discovery and registration
    - Dependencies: Redis
8. **Optimizer Service:**
    - Hyperparameter optimization with backtesting integration
    - Metrics port: 9092

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
- **Qdrant**: Vector similarity search engine for schema pattern matching
- **Prometheus** (port 9090): Metrics collection with 1700+ lines of alert rules

- **Grafana** (port 3000): Visualization dashboards (6+ dashboards including strategy, regime, CNS, brain region monitors)
- **Alertmanager** (port 9093): Alert routing with Discord integration
- **Jaeger** (port 16686): Distributed tracing
- **Loki + Promtail**: Centralized log aggregation with label-based organization
- **Authelia**: OpenID Connect / SAML authentication gateway
- **Nginx**: Reverse proxy with SSL termination

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.

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, distributed training infrastructure, and Wilson-Cowan oscillatory dynamics
2. **Neuro-Symbolic Fusion:** Dual-mode LTN system with domain-specific axioms and linguistic hedges bridging neural networks and logical constraints
3. **Multi-Timescale Memory:** Three-tier hierarchy with sleep-phase consolidation mirrors hippocampal-neocortical transfer, implementing Complementary Learning Systems theory
4. **Ensemble Regime Detection:** HMM, statistical, and technical methods fused for robust market state identification with strategy routing
5. **Production-Ready Safety:** Four-scope kill switch, multi-method anomaly detection, FNI-RL fear network, and CNS health monitoring with five-phase preflight
6. **Pure Rust Stack:** End-to-end training and inference in Rust with high-performance, safe, and maintainable eight-service architecture
7. **Fractal Signal Processing:** Sub-microsecond DSP pipeline producing 8D feature vectors via FRAMA and Sevcik fractal dimension for regime-aware signal generation
8. **Synthetic Data Generation:** Regime-conditional DDPM diffusion models for synthetic market data with quality assessment, augmenting sparse regime training data
9. **Quantum-Inspired Optimization:** QAOA simulation, VQE portfolio optimization, and simulated quantum annealing for non-convex portfolio selection
10. **GPU Compute Infrastructure:** Custom wgpu kernels (MatMul, Softmax, LayerNorm, Attention, GELU) with Candle tensor bridge for hardware-accelerated inference

Future Work

The following items remain as planned enhancements beyond the current implementation:

- Hardware acceleration using FPGAs (Marino et al., 2023; Vemeko, 2023; AMD, 2023) and neuromorphic chips for nanosecond-level latency in high-frequency trading applications

Already implemented since initial specification: VPIN flow toxicity calculator with flash crash detection, full PID controller with Ziegler-Nichols auto-tuning, comprehensive wash sale detection in dedicated compliance crate with proprietary firm rule enforcement, nine regime-aware trading strategies with gating and affinity scoring, four-scope kill switch (per-strategy/instrument/service/global), FNI-RL fear network with reinforcement learning adaptation, neuromorphic distributed training infrastructure (multi-GPU/multi-node with AllReduce, Parameter Server, and Ring AllReduce), feudal RL hierarchy with cortex manager and hippocampus worker agents, domain-specific LTN axiom system with 10 market rules and hybrid supervised-semantic loss, complete Markov transition matrix with stationary distribution for regime prediction, 8D DSP feature vector pipeline with sub-microsecond latency, end-to-end VisionPipeline orchestration chaining DiffGAF $\rightarrow$ ViViT for fully differentiable time-series-to-embedding inference (with dual-GAF, intermediate visualisation, and optional projection head support), Complementary Learning Systems theory implementation across three-tier memory hierarchy, Wilson-Cowan thalamic oscillation models (Wilson and Cowan, 1972; Author, 2024a) with Hilbert-transform amplitude/phase estimation and bifurcation analysis, Chronos time series forecasting (Ansari et al., 2024b; Ansari et al., 2024a) via ONNX inference with quantile tokenization and confidence intervals, BERT/FinBERT sentiment embeddings (Hugging Face, 2024) running natively in Rust via Candle with Qdrant-backed storage, parametric UMAP (McInnes, Healy, and Melville, 2018) for real-time schema monitoring with drift detection and regime cluster analysis, quantum-inspired portfolio optimization (Author, 2024d) including QAOA simulation, VQE portfolio optimization, and simulated quantum annealing, generative diffusion models (DDPM) (Author, 2024e) for regime-conditional synthetic market data with quality assessment, Rust-native GPU-accelerated limit order book simulator (Fu, Pakkanen, and Cont, 2024) with full matching engine, iceberg/stop/FOK/IOC order types, self-trade prevention, and Almgren-Chriss market impact modeling, custom GPU compute kernels via wgpu (MatMul, Softmax, LayerNorm, Attention, GELU, Reduce, Embedding, Pairwise Distance) with Candle tensor bridge, and full Qdrant production client with circuit breaker, exponential backoff retries, TLS support, and graceful mock fallback.

Repository & Contact

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

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

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

References

Almgren, Robert and Neil Chriss (2001). “Optimal execution of portfolio transactions”. In: *Journal of Risk* 3, pp. 5–40.

AMD (2023). *AMD Alveo U55C data center accelerator card*. URL: https://www.amd.com/en/products/accelerators/alveo/u55c.html.

Ansari, Abdul Fatir et al. (2024a). “Introducing Chronos-2: Large language model time series forecasting”. In: *Amazon Science Blog*. URL: https://www.amazon.science/blog/introducing-chronos-2.

Ansari, Abdul Fatir, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Syndar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. (2024b). “Chronos: Learning the language of time series”. In: *arXiv preprint arXiv:2403.07815*.

Arnab, Anurag, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid (2021). “ViViT: A video vision transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6836–6846.

Author, Various (2016). “Substantia nigra-amygdala connections underlie fear extinction”. In: *Nature*.

— (2018). “Thalamic reticular nucleus activation reflects attentional gating”. In: *Journal of Neuroscience*.

— (2019). “Fear-neuro-inspired reinforcement learning for safe autonomous driving”. In: *IEEE Transactions on Neural Networks and Learning Systems*.

— (2020). “Dopamine and serotonin differentially associated with reward and punishment processes”. In: *Nature Neuroscience*.

— (2023a). *AlignedUMAP for temporal manifold alignment*. URL: https://umap-learn.readthedocs.io/en/latest/aligned_umap_basic_usage.html.

— (2023b). *Building the software that powers high-frequency trading in TradFi*. URL: https://www.tradfi.com/hft-software.

— (2023c). “Gated cross-attention for multimodal fusion”. In: *Computer Vision and Pattern Recognition*.

— (2024a). “Bidirectionally regulating gamma oscillations in Wilson-Cowan model”. In: *Neural Computation*.

— (2024b). “Deep differentiable logic gate networks based on fuzzy Łukasiewicz T-norm”. In: *Neural Networks*.

— (2024c). *Deep dive into IS: The Almgren-Chriss framework*. URL: https://www.internalscience.com.

— (2024d). “Dynamic portfolio optimization with real datasets using quantum processors and quantum-inspired tensor networks”. In: *Science Advances*.

Author, Various (2024e). “Generative diffusion models for financial limit order book simulation”. In: *arXiv preprint*.

— (2025). “Fusion of recurrence plots and Gramian angular fields for time series classification”. In: *Pattern Recognition*. Recent validation of GAF encodings for temporal structures.

Badreddine, Samy, Artur d’Avila Garcez, Luciano Serafini, and Michael Spranger (2022). “Logic tensor networks”. In: *Artificial Intelligence* 303, p. 103649.

Buzsáki, György (2015). “Hippocampal sharp wave-ripple: A cognitive biomarker for episodic memory and planning”. In: *Hippocampus* 25.10, pp. 1073–1188.

Collins, Anne GE and Michael J Frank (2014). “Opponent actor learning (OpAL): Modeling interactive effects of striatal dopamine on reinforcement learning and choice incentive”. In: *Psychological Review* 121.3, p. 337.

Daw, Nathaniel D, John P O’Doherty, Peter Dayan, Ben Seymour, and Raymond J Dolan (2006). “Cortical substrates for exploratory decisions in humans”. In: *Nature* 441.7095, pp. 876–879.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. (2020). “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929*.

Easley, David, Marcos Lopez De Prado, and Maureen O’Hara (2012). “Flow toxicity and liquidity in a high-frequency world”. In: *The Review of Financial Studies* 25.5, pp. 1457–1493.

Easley, David, Marcos M Lopez de Prado, and Maureen O’Hara (2011). “The microstructure of the "flash crash": flow toxicity, liquidity crashes, and the probability of informed trading”. In: *The Journal of Portfolio Management* 37.2, pp. 118–128.

Evans, Jonathan St BT (2008). “Dual-processing accounts of reasoning, judgment, and social cognition”. In: *Annual Review of Psychology* 59, pp. 255–278.

Fama, Eugene F and Kenneth R French (1993). “Common risk factors in the returns on stocks and bonds”. In: *Journal of Financial Economics* 33.1, pp. 3–56.

Foster, David J, Richard GM Morris, and Peter Dayan (2013). “Mechanisms of hierarchical reinforcement learning in corticostriatal circuits”. In: *Nature Neuroscience* 16, pp. 383–391.

Frank, Michael J, Bryan Loughry, and Randall C O’Reilly (2006). “Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia”. In: *Neural Computation* 18.2, pp. 283–328.

Fu, Sascha, Mikko S Pakkanen, and Rama Cont (2024). “JAX-LOB: A GPU-accelerated limit order book simulator to unlock large scale reinforcement learning”. In: *arXiv preprint arXiv:2408.08806*.

Garcez, Artur d'Avila and Luís C Lamb (2024). "Mapping the neuro-symbolic AI landscape". In: *AI Magazine*.

Halassa, Michael M and Sabine Kastner (2017). "Thalamic functions in distributed cognitive control". In: *Nature Neuroscience* 20.12, pp. 1669–1679.

Hugging Face (2024). *Candle: Minimalist ML framework for Rust*. URL: https://github.com/huggingface/candle.

Internal Revenue Service (2024). *Wash sale rule basics for active traders*. URL: https://www.irs.gov/taxtopics/tc409.

Kahneman, Daniel (2011). *Thinking, fast and slow*. Farrar, Straus and Giroux.

Kar, Kanishka et al. (2023). "Selection of experience for memory by hippocampal sharp wave ripples". In: *Science* 380, pp. 1171–1175.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444.

Marino, Kevin et al. (2023). "ME-ViT: A single-load memory-efficient FPGA accelerator for vision transformers". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

Markwick, David (2023). *Solving the Almgren-Chriss model*. URL: https://www.actuaries.digital/2023/01/19/solving-the-almgren-chriss-model/.

McClelland, James L, Bruce L McNaughton, and Randall C O'Reilly (1995). "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory". In: *Psychological Review* 102.3, p. 419.

McInnes, Leland, John Healy, and James Melville (2018). "UMAP: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426*.

Monfils, Marie-H, Kiriana K Cowansage, Eric Klann, and Joseph E LeDoux (2009). "Extinction-reconsolidation boundaries: key to persistent attenuation of fear memories". In: *Science* 324.5929, pp. 951–955.

Polars Contributors (2024). *Polars: Lightning-fast DataFrame library*. URL: https://www.pola.rs/.

Qdrant (2024). *Qdrant: Vector similarity search engine*. URL: https://qdrant.tech/.

Schaul, Tom, John Quan, Ioannis Antonoglou, and David Silver (2015). "Prioritized experience replay". In: *arXiv preprint arXiv:1511.05952*.

Sterling, Peter (2012). "Allostasis: a model of predictive regulation". In: *Physiology & Behavior* 106.1, pp. 5–15.

Tokio Contributors (2024). *Tokio: Asynchronous runtime for Rust*. URL: https://tokio.rs/.

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

Vemeko (2023). *How to use FPGAs for HFT acceleration*. URL: https://www.vemeko.com/fpga-hft.

Wang, Zhiguang and Tim Oates (2015). "Imaging time-series to improve classification and imputation". In: *arXiv preprint arXiv:1506.00327*. URL: https://arxiv.org/abs/1506.00327.

Wilson, Hugh R and Jack D Cowan (1972). "Excitatory and inhibitory interactions in localized populations of model neurons". In: *Biophysical Journal* 12.1, pp. 1–24.